The PassKit API

PassKit

DOCUMENTATION

Introduction

The API tools serve as the client interface to the PassKit web service. Use these tools to create and update mobile wallet passes, and more.

If you cannot find a method to achieve your objective please visit PassKit support and choose 'API and Customisation' as the help topic.

API End Points

API ENDPOINT

https://api.passkit.com/

SUMMARY OF RESOURCE URL PATTERNS

- /v1/authenticate
- /v1/template/{templateName}/fieldnames
- /v1/template/{templateName}/passes
- /v1/template/list
- /v1/template/update/{templateName}
- /v1/template/{templateName}/reset
- /v1/pass/issue/template/{templateName}
- /v1/pass/update/template/{templateName}/serial/{serialNumber}
- /v1/pass/update/passid/{passId}
- /v1/pass/invalidate/template/{templateName}/serial/{serialNumber}
- /v1/pass/invalidate/passid/{passId}
- /v1/pass/get/template/{templateName}/serial/{serialNumber}
- /v1/pass/get/passid/{passId}
- /v1/pass/issue/batch/template/{templateName}
- /v1/image/add/{imageType}
- /v1/image/{imageID}

PASSKIT REST API

Introduction

The API tools serve as the client interface to the PassKit web service. Use these tools to create and update mobile wallet passes, and more.

If you cannot find a method to achieve your objective please visit PassKit support and choose 'API and Customisation' as the help topic.

Accessing the PassKit API

Our API is accessed with an API Key and Secret.

Request your Keys from the Pass Designer, under My Account, in the Details tab. Your API Key is a 20 or 32 character string. Your API Secret is longer and may contain periods and forward slashes. For example:

```
API Key (Base16):
ff0b04afe47feacd09a850d9a1dd91d0
API Key (Base62): B9gaEoDarsV7e1EgutQg
API Secret: kFTlvlfrjU/
djar.V3tO0uyvoF0svLGVhM7ccGN.ek80GdqcJNcju
```

These keys allow access to templates and passes on your account. They should be treated the same as your username and password, and should never be written down or shared.

API AUTHENTICATION

The PassKit API supports both HTTP Digest and HMAC authentication. HTTP Digest and HMAC are compatible with all modern web browsers, allowing you to POST and GET using simple HTML forms.

Our HTTP Digest realm is rotated every 2 minutes.

When using one of our SDK wrapper classes (for PHP or C#), you don't have to worry about writing the authentication bit, since the wrapper already takes care of it.

HMAC

In order for the HMAC authentication to work properly, set an Authorization header with the following:

1. The Authorization header string needs to start with 'PKAuth '
2. Next it needs to have the API key
3. Next it needs to have the hash of the data being sent
4. Next it needs to have a timestamp

The hash needs to contain: `requestMethod`**\n** `requestUri` **(minus / at the end)\n** `data` **\n**`timestamp`. Then use the php `hash_hmac` function to hash the data-string with the API secret and a timestamp.

Note that `data` is a string of all the (post/get) data being sent in the format of: [key=value&].

Example of the hashing algorithm being implemented in PHP (copy this logic in any language – make sure to sort the POST array first):

```
$method = "GET";
$dataString = '';
$apiKey = 'My API Key';
$apiSecret = 'My API Secret';
// If data isset - sort it, and build the
string for HMAC hashing
if($post != null) {
    $method = "POST";
```

Continued on next page.

PASSKIT API

API Authentication

```
    uksort($post, 'strcmp');
    $post = array();

    // Loop through every data entry - only
do this for non image method
    foreach($post as $key => $value) {
        $dataString .= rawurlencode($key) .
'=' . rawurlencode($value) . "&";
    }

    $dataString = substr($dataString, 0, -1);
}

// String with data to hash
$to_hash = $method . "\n" . trim($this-
>apiVersion . "/" . $path, '/') . "\n" .
$dataString . "\n" . time();
// Do the actual hash
$hash = base64_encode(hash_hmac('sha256',
$to_hash, $apiSecret, true));

// Set Authorization string
$authorizationString = "PKAuth ".$apiKey." ".
$hash." ".time();
$headers[0] = "Authorization: ".
$authorizationString;
```

Continued from previous page.

DIGEST

Our HTTP Digest realm is rotated every 2 minutes.

PHP Example of Digest implementation in CURL:

```
$apiKey = 'My API Key';
$apiSecret = 'My API Secret';

curl_setopt ($ch, CURLOPT_HTTPAUTH,
CURLAUTH_DIGEST);
curl_setopt ($ch, CURLOPT_USERPWD,
$apiKey.':'.$apiSecret);
```

PASSKIT API

API Authentication

To do basic testing and debugging, use cURL, a command-line tool available on Macs, Windows, Linux.

Some examples for CURL

```
# GET
curl --digest -u {API Key}:{API Secret} \
    "https://api.passkit.com/{version}/{method
URL}"

# POST
curl --digest -u {API Key}:{API Secret} --data
"{method parameters}" \
    -F "{POST Fieldname}=@{path and filename};
{MIME Type}"  \
    -X POST "https://api.passkit.com/
{version}/{method URL}"

# GET template fieldnames for “My Great Pass”
curl --digest -u {API Key}:{API Secret} \
    "https://api.passkit.com/v1/template/My
%20Great%20Pass/fieldnames"

# POST upload a thumbnail image (an
unauthenticated call)
curl -F "image=@/Users/Percy/Pictures/
Percy.jpg;type=image/jpeg" \
    -X POST "https://api.passkit.com/v1/image/
add/thumbnail"

# POST Pass Issue for “My Great Pass” template
and
# retrieve the results in XML
curl --digest -u {API Key}:{API Secret} --data
"Issued To=Percy PassKit&Balance=20" \
    "https://api.passkit.com/v1/pass/issue/
template/My%20Great%20Pass/?format=xml"

# POST Issue a Batch of Passes for “My Great
Pass”
curl --digest -u {API Key}:{API Secret} -d
@batch.json \
    "https://api.passkit.com/v1/pass/issue/
batch/template/My%20Great%20Pass"

# PUT Pass Update the balance and reward tier
of “My Great Pass”,
# serial number 1234 and push an update
curl --digest -u {API Key}:{API Secret} -d
"Balance=29.88&Reward Tier=Gold" -X PUT \
    "https://api.passkit.com/v1/pass/update/
template/My%20Great%20Pass/serial/1234/push"
```

PASSKIT SDK

PHP SDK

All the PHP implementation examples on this documentation page use the our PHP SDK. We advise you to use our wrapper class if you are developing in PHP. This class is developed, tested and maintained by the PassKit team, so it will ensure that your website/ application communicates with our API in the best way possible.

The latest version of the PassKit SDK for PHP can be downloaded at our Google Code Repository.

C# SDK

All the C# implementation examples on this documentation page use the C# SDK. We advise you to use our wrapper class if you are doing development in C#. This class is developed, tested and maintained by the PassKit team, so it will ensure that your website/ application communicates with our API in the best way possible.

The latest version of the PassKit SDK for C# can be downloaded at our Google Code Repository.

Other Languages

To consume our API via other languages we recommend that you use Mashape. We have made our API available through their system, and they support all major languages: Java, Node, PHP, Pyhton, Objective-C, Ruby, .NET.

The PassKit API at Mashape

AUTHENTICATE

Description

URI	https://api.passkit.com/v1/authenticate /? format=xml (optional)
Verb	GET
Auth	Yes

Authenticates access to the API. `true` on successful authentication or an error message on failure.

Request

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret}
"https://api.passkit.com/v1/authenticate"
```

Response

success (string)
`true` on success (will not be present on error).

JSON Response

```
{
   "success" : true,
}
```

XML Response

```
<PassKit_API_Response>
    <success>1</success>
</PassKit_API_Response>
```

GET TEMPLATE FIELD NAMES

Description

URI	https://api.passkit.com/v1/template/{templateName}/fieldnames /full (optional) /?format=xml (optional)
Verb	GET
Auth	Yes

Return a template's dynamic field names, which are used by the Issue Pass and Update Pass methods. In addition to all dynamic field names, variables such as barcode content, serial number, and thumbnail image are returned.

To also return static field names, append `/full` to the URI.

URI PARAMETERS

templateName (string)
The template name.

Request

cURL Syntax

```
# Normal Request
curl --digest -u {APIKey}:{APISecret} \
    "https://api.passkit.com/v1/template/
{templateName}/fieldnames"

# With full parameter:
curl --digest -u {APIKey}:{APISecret} \
    "https://api.passkit.com/v1/template/
{templateName}/fieldnames/full"
```

TEMPLATE METHODS

Get Template Field Names

Response

RESPONSE PARAMETERS

{fieldName} (string)
The name of each dynamic 'value' field, together with the data type: one of `text`, `number`, `date`, `datetime` or `currency`.

{fieldName}changeMessage (string)
A field's Change Message, if set.

{fieldName}label (string)
The ID of a dynamic 'label' field with a value of text.

thumbnailImage (string)
A template's thumbnail image, if it applicable for this template.

serialNumber (string)
If the template requires a serial to be provided at pass creation, returns a value of text.

recoveryEmail (string)
All passes accept a recovery email address. This parameter will always be returned with a value of text. An invalid email address will return an error.

barcodeContent (string)
If the barcode content can be changed, returns a value of text.

barcodeAltContent (string)
If the template allows for the barcode alternative content to be changed, returns a value of text.

EXAMPLES

JSON Response

Continued on next page.

```
{
  "My Pass Template" : {
    "Check In"                 : "date",
    "Check In_changeMessage"   : "text",
    "Check Out"                : "date",
    "Check Out_changeMessage"  : "text",
    "No Rooms"                 : "number",
    "No Guests"                : "number",
    "Reservation Num"          : "text",
    "Status"                   : "text",
```

TEMPLATE METHODS

Get Template Field Names

```
    "Status_changeMessage:"    : "text",
    "Guest Name"               : "text",
    "Hotel Name"               : "text",
    "Hotel Address"            : "text",
    "Hotel Telephone"          : "text",
    "Nightly Rate"             : "currency",
    "recoveryEmail"            : "text"
  }
}
```

Continued from previous page.

```
<PassKit_API_Response>
  <My·Pass·Template>
    <Check·In>date</Check·In>
    <Check·In_changeMessage>text</
Check·In_changeMessage>
    <Check·Out>date</Check·Out>
    <Check·Out_changeMessage>text</
Check·Out_changeMessage>
    <No.·Rooms>number</No.·Rooms>
    <No.·Guests>number</No.·Guests>
    <Reservation·Num>text</Reservation·Num>
    <Status>text</Status>
    <Status_changeMessage>text</
Status_changeMessage>
    <Guest·Name>text</Guest·Name>
    <Hotel·Name>text</Hotel·Name>
    <Hotel·Address>text</Hotel·Address>
    <Hotel·Telephone>text</Hotel·Telephone>
    <Nightly·Rate>currency</Nightly·Rate>
    <recoveryEmail>text</recoveryEmail>
  </My·Pass·Template>
</PassKit_API_Response>
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

TEMPLATE METHODS

Get Template Field Names

Implementation Examples

PHP Implementation Example

```php
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My template name";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

// Defaults to /full parameter
$result = $pk-
>getTemplateFieldNames($templateName);

// Without /full parameter
$result = $pk-
>getTemplateFieldNames($templateName, false);

// Do something with result
print_r($result);
```

C# Implementation Example

Continued on next page.

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
```

TEMPLATE METHODS

Get Template Field Names

```
        private string apiAccount =
"apiAccount";
        private string apiSecret =
"apiSecret";

        protected void Page_Load(object
sender, EventArgs e)
        {
            string template_name = "My
template name";

            // Initialize new instance of
PassKit API wrapper
            PassKit pk = new
PassKit(apiAccount, apiSecret);

            PassKitResponse result =
pk.GetTemplateFieldNames(template_name);

            // Do something with result
        }
    }
}
```

Continued from previous page.

GET PASSES FOR TEMPLATE

Description

URI	https://api.passkit.com/v1/template/{templateName}/passes /?format=xml (optional)
Verb	GET
Auth	Yes

Returns basic template information and all issued passes (pass meta and field data) for the template.

URI PARAMETERS

templateName (string)
The template name.

Request

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret}
"https://api.passkit.com/v1/template/
{templateName}/passes"
```

Get Passes For Template

Response

TEMPLATE PARAMETERS

success (string)
`true` on success (will not be present on error).

templateName (string)
The template name.

templateLastUpdated (date)
The date when the template was last updated.

totalPasses (int)
The total number of issued passes.

passRecords (array)
An array of all the pass records.

PASS META-DATA PARAMETERS

passStatus (string)
The status of the pass.

installIp (string)
The IP address of the device when the pass was installed.

installIPCountry (string)
The country that the device was in when the pass was installed.

installIPCity (string)
The city that the device was in when the pass installed.

recoveryURL (string)
The recovery URL of the pass.

issueDate (date)
The issue date of the pass.

Continued on next page.

lastDataChange (date)
The time when data was last changed on the pass.

lastPushRefresh (date)
The time when data was pushed last to the pass.

Get Passes For Template

PASS META-DATA PARAMETERS

Continued from previous page.

lastManualRefresh (date)
The last time the user did a manual refresh.

passExpires (date)
The expiry date of the pass.

passbookSerial (string)
The PassBook serial of the pass.

uniqueID (string)
The unique pass-id of the the pass.

serialNumber (string)
The serial number of the pass.

shareID (string)
The share-id of the pass.

PASS FIELD-DATA PARAMETERS

{fieldName} (string)
The name of each dynamic 'value' field, together with the data type: one of `text`, `number`, `date`, `datetime` or `currency`.

barcodeContent (string)
If the barcode content can be changed, returns a value of text.

EXAMPLES

JSON Response

```
{
  "success": true,
  "templateName": "My template",
  "templateLastUpdated":
"2013-02-08T01:02:55+00:00",
  "totalPasses": 2,
  "passRecords": {
    "pass_1": {
      "passMeta": {
        "passStatus": "Active",
        "installIP": "123.123.123.123",
        "installIPCountry": "HK",
        "installIPCity": "Central District",
        "recoveryURL": "http://
recoveryUrl.com",
```

Continued on next page.

TEMPLATE METHODS

Get Passes For Template

Continued from previous page.


```
        "issueDate":
"2013-02-10T08:19:30+00:00",
        "lastDataChange":
"2013-03-02T04:56:47+00:00",
        "lastPushRefresh":
"2013-02-10T14:45:08+00:00",
        "lastManualRefresh":
"2013-03-02T04:57:09+00:00",
        "passExpires":
"2013-03-02T04:57:09+00:00",
        "passbookSerial": "uniqueSerial",
        "uniqueID": "uniqueId",
        "serialNumber": "serialNumber",
        "shareID": "shareid"
      },
      "passData": {
        "Student·name": "Test student",
        "Balance": "8",
        "Issue·date": "2013-02-10",
        "Expiry·date": "2014-02-10",
        "barcodeContent": "content"
      }
    },
    "pass_2": {
      "passMeta": {
        "passStatus": "Active",
        "installIP": "123.123.123.123",
        "installIPCountry": "HK",
        "installIPCity": "Central District",
        "recoveryURL": "http://
recoveryUrl.com",
        "issueDate":
"2013-02-10T08:19:30+00:00",
        "lastDataChange":
"2013-03-02T04:56:47+00:00",
        "lastPushRefresh":
"2013-02-10T14:45:08+00:00",
        "lastManualRefresh":
"2013-03-02T04:57:09+00:00",
        "passExpires":
"2013-03-02T04:57:09+00:00",
        "passbookSerial": "uniqueSerial",
        "uniqueID": "uniqueId",
        "serialNumber": "serialNumber",
        "shareID": "shareid"
      },
```


Continued on next page.

TEMPLATE METHODS

Get Passes For Template

Continued from previous page.

```
"passData": {
        "Student·name": "Test student 2",
        "Balance": "8",
        "Issue·date": "2013-02-10",
        "Expiry·date": "2014-02-10",
        "barcodeContent": "content"
      }
    }
  }
}
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

```
<PassKit_API_Response>
  <success>1</success>
  <templateName>My template</templateName>
  <templateLastUpdated>2013-02-08T01:02:55+00
:00</templateLastUpdated>
  <totalPasses>2</totalPasses>
  <passRecords>
    <pass_1>
      <passMeta>
        <passStatus>Active</passStatus>
        <installIP>123.123.123.123</
installIP>
        <installIPCountry>HK</
installIPCountry>
        <installIPCity>Central District</
installIPCity>
        <recoveryURL>http://recoveryUrl.com</
recoveryURL>
        <issueDate>2013-02-10T08:19:30+00:00<
/issueDate>
        <lastDataChange>2013-03-02T04:56:47+0
0:00</lastDataChange>
        <lastPushRefresh>2013-02-10T14:45:08+
00:00</lastPushRefresh>
        <lastManualRefresh>2013-03-02T04:57:0
9+00:00</lastManualRefresh>
        <passExpires>2013-03-02T04:57:09+00:0
0</passExpires>
        <passbookSerial>uniqueSerial</
passbookSerial>
        <uniqueID>uniqueId</uniqueID>
        <serialNumber>serialNumber</
serialNumber>
```

Continued on next page.

TEMPLATE METHODS

Get Passes For Template

Continued from previous page.

```
        <shareID>shareid</shareID>
      </passMeta>
      <passData>
        <Student.name>Test student</
Student.name>
        <Balance>8</Balance>
        <Issue.date>2013-02-10</Issue.date>
        <Expiry.date>2014-02-10</Expiry.date>
        <barcodeContent>content</
barcodeContent>
      </passData>
    </pass_1>
    <pass_2>
      <passMeta>
        <passStatus>Active</passStatus>
        <installIP>123.123.123.123</
installIP>
        <installIPCountry>HK</
installIPCountry>
        <installIPCity>Central District</
installIPCity>
        <recoveryURL>http://recoveryUrl.com</
recoveryURL>
        <issueDate>2013-02-10T08:19:30+00:00<
/issueDate>
        <lastDataChange>2013-03-02T04:56:47+0
0:00</lastDataChange>
        <lastPushRefresh>2013-02-10T14:45:08+
00:00</lastPushRefresh>
        <lastManualRefresh>2013-03-02T04:57:0
9+00:00</lastManualRefresh>
        <passExpires>2013-03-02T04:57:09+00:0
0</passExpires>
        <passbookSerial>uniqueSerial</
passbookSerial>
        <uniqueID>uniqueId</uniqueID>
        <serialNumber>serialNumber</
serialNumber>
        <shareID>shareid</shareID>
      </passMeta>
      <passData>
        <Student.name>Test student 2</
Student.name>
        <Balance>8</Balance>
        <Issue.date>2013-02-10</Issue.date>
```

Continued on next page.

TEMPLATE METHODS

Get Passes For Template

```
<Expiry·date>2014-02-10</Expiry·date>
        <barcodeContent>content</
barcodeContent>
      </passData>
    </pass_2>
  </passRecords>
</PassKit_API_Response>
```

Continued from previous page.

Implementation Examples

C# Implementation Example

Continued on next page.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        private string apiAccount =
"apiAccount";
        private string apiSecret =
"apiSecret";

        protected void Page_Load(object
sender, EventArgs e)
        {
            string template_name = "My
template name";

            // Initialize new instance of
PassKit API wrapper
```

Note: The PHP and C# examples make use of our SDK classes. The SDK classes have been written to make access to the PassKit API even easier. The SDK's can be downloaded at our Google Code Repository.

TEMPLATE METHODS

Get Passes For Template

```
            PassKit pk = new
PassKit(apiAccount, apiSecret);

            PassKitResponse result =
pk.GetPasses(template_name);
            // Do something with result
        }
    }
}
```

Continued from previous page.

PHP Implementation Example

```
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My template name";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

$result = $pk-
>getTemplatePasses($templateName);

// Do something with result
print_r($result);
```

LIST TEMPLATES

Description

URI	https://api.passkit.com/v1/template/list /?format=xml (optional)
Verb	GET
Auth	Yes

List this account's templates.

Request

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret}
"https://api.passkit.com/v1/template/list"
```

Response

RESPONSE PARAMETERS

success (string)
`true` on success (will not be present on error).

templates (array)
This account's templates.

TEMPLATE METHODS

List Templates

EXAMPLES

```
{
   "success" : true,
   "templates" : [
      "My template 1",
      "My template 2",
      "My template 3"
   ]
}
```

JSON Response

```
<PassKit_API_Response>
  <success>1</success>
  <templates>
    <item>My template 1</item>
    <item>My template 2</item>
    <item>My template 3</item>
  </templates>
</PassKit_API_Response>
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

Implementation Examples

```
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My template name";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

$result = $pk->listTemplates();

// Do something with result
print_r($result);
```

PHP Implementation Example

TEMPLATE METHODS

List Templates

C# Implementation Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        private string apiAccount =
"apiAccount";
        private string apiSecret =
"apiSecret";

        protected void Page_Load(object
sender, EventArgs e)
        {
            // Initialize new instance of
PassKit API wrapper
            PassKit pk = new
PassKit(apiAccount, apiSecret);

            PassKitResponse result =
pk.GetTemplates();

            // Do something with result
        }
    }
}
```

Note: The PHP and C# examples make use of our SDK classes. The SDK classes have been written to make access to the PassKit API even easier. The SDK's can be downloaded at our Google Code Repository.

UPDATE TEMPLATE

Description

URI	https://api.passkit.com/v1/template/update/{templateName} /push (optional) /? format=xml (optional)
Verb	GET,POST,PUT
Auth	Yes

Update a template's default values. To update existing passes, append /push to the URI.

URI PARAMETERS

templateName (string)
The template to be updated.

Request

REQUEST PARAMETERS

{fieldName} (mixed *optional*)
The default value for a dynamic field within the template. The submitted data type must match the data type of the field, as specified in the template. Date values must be in ISO 8601 format.

cURL Syntax

```
# GET
curl --digest -u {APIKey}:{APISecret} \
    "https://api.passkit.com/v1/template/
update/{templateName}/push?{parameter}
={value}&..."

# POST
curl --digest -u {APIKey}:{APISecret} -d
"{parameter}={value};{parameter}={value};..."
\
    "https://api.passkit.com/v1/template/
update/{templateName}/push"

# PUT
curl --digest -u {APIKey}:{APISecret} -d
@{filename} -X PUT \
    "https://api.passkit.com/v1/template/
update/{templateName}/push"
```

TEMPLATE METHODS

Update Template

Response

RESPONSE PARAMETERS

success (string)
`true` on success (will not be present on error).

devices (array)
The deviceIDs that will receive this update via push.

EXAMPLES

JSON Response

```
{
   "success" : true,
   "devices" : {
      "device_1" :
"a7fc7ff20ef26c95808ab820e2aa797e0c590df760fe
bd974051287779ec69f5"
   }
}
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

```
<PassKit_API_Response>
  <success>1</success>
  <devices>
    <device_1>a7fc7ff20ef26c95808ab820e2aa797
e0c590df760febd974051287779ec69f5</device_1>
  </devices>
</PassKit_API_Response>
```

RESET TEMPLATE

Description

URI	https://api.passkit.com/v1/template/{templateName}/reset /push (optional) /?format=xml (optional)
Verb	GET,POST,PUT
Auth	Yes

Reset pass records to default values and remove all data-fields from each pass record. This only affects values that the user cannot edit.

Each pass can have an unlimited number of fields that start with "data-". These fields do not show up on the pass, but are attached to the pass record and thus returned with every request for pass data.

To reset existing passes, append /push to the URI.

URI PARAMETERS

templateName (string)
The template to be reset.

Request

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret}
"https://api.passkit.com/v1/template/
{templateName}/reset/push"
```

Reset Template

Response

RESPONSE PARAMETERS

success (string)
`true` on success (will not be present on error).

devices (array)
The deviceIDs that will receive this update via push.

EXAMPLES

JSON Response

```
{
   "success" : true,
   "devices" : {
      "device_1" :
"a7fc7ff20ef26c95808ab820e2aa797e0c590df760fe
bd974051287779ec69f5"
   }
}
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

```
<PassKit_API_Response>
  <success>1</success>
  <devices>
    <device_1>a7fc7ff20ef26c95808ab820e2aa797
e0c590df760febd974051287779ec69f5</device_1>
  </devices>
</PassKit_API_Response>
```

Implementation Examples

PHP Implementation Example

Continued on next page.

```
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
```

TEMPLATE METHODS

Reset Template

```php
$templateName = "My template name";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

$result = $pk->resetTemplate($templateName);

// Do something with result
print_r($result);
```

Continued from previous page.

C# Implementation Example

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        private string apiAccount =
"apiAccount";
        private string apiSecret =
"apiSecret";

        protected void Page_Load(object
sender, EventArgs e)
        {
            string template_name = "My
template name";

            // Initialize new instance of
PassKit API wrapper
            PassKit pk = new
PassKit(apiAccount, apiSecret);
```

Note: The PHP and C# examples make use of our SDK classes. The SDK classes have been written to make access to the PassKit API even easier. The SDK's can be downloaded at our Google Code Repository.

TEMPLATE METHODS

Reset Template

```
            // Reset all passes for the
template
            PassKitResponse result =
pk.ResetTemplate(template_name);
            // Do something with result
        }
    }
}
```

Continued from previous page.

ISSUE PASS

Description

URI	https://api.passkit.com/v1/pass/issue/template/{templateName} /?format=xml (optional)
Verb	GET,POST
Auth	Yes

Issues passes with parameters returned by Get Template Field Names, plus a relevant date, locations, and beacons. Requests are issued as a GET or POST request using key/value pairs, or by PUTing a JSON or XML file. A successful submission returns the serial number and URL of the issued pass.

URI PARAMETERS

templateName (string)
The name of the template from which the pass will be created.

Request

GENERIC PARAMETERS

{fieldName} (mixed) *optional*
The value for a dynamic field within the template. The submitted data type must match the data type of the field, as specified in the template. Date values must be in ISO 8601 format. If not provided, the pass will return the default value for the field or an empty string or zero value for number and currency fields if no default is present.

Continued on next page.

{fieldName}_changeMessage (string) *optional*
The notification message displayed when the field value changes. Include %@ in the string to include the new value in the notification.

{fieldName}_label (string) *optional*
The value of a dynamic label field. Label values are always treated as text.

Issue Pass

GENERIC PARAMETERS

Continued from previous page.

serialNumber (string) *optional*
For templates that require a serial number to be provided at pass creation. The serial provided must be unique within the template.

recoveryEmail (string) *optional*
The email address used to reinstall a pass. An invalid email address will return an error.

barcodeContent (string) *optional*
For templates that allow the barcode encoded content to be changed. Using %@ in the string will insert the pass serial number into the encoded content.

barcodeAltContent (string) *optional*
For templates that allow the barcode alternative content to be changed. Using %@ in the string will insert the pass serial number into the message.

relevantDate (string) *optional*
An ISO 8601 formatted date for pass types that support a Relevant Date.

expirationDate (ISO 8601 date) *optional (iOS 7)*
The expiration date of the pass.

voided (bool) optional (iOS 7)
If set to `true`, this pass is voided. Useful for single-use coupons.

COLOUR PARAMETERS

labelColor / labelColour (string) *optional*
The label color. Accepts a 6 digit HEX color code.

foregroundColor / foregroundColour (string) *optional*
The foreground color. Accepts a 6 digit HEX color code.

backgroundColor / backgroundColour (string) *optional*
The background color. Accepts a 6 digit HEX color code.

groupingIdentifier (string) *optional (iOS 7)*
Identifier for grouping event tickets and boarding passes with the same style, pass type identifier, and grouping identifiers. For example, multiple passes for a big event can be grouped together.

Issue Pass

IMAGE PARAMETERS

The images on each pass are set by the image parameters. Each parameter accepts an imageID returned by the Upload Image method. ImageIDs can be checked with the Get Image Data method.

thumbnailImage (imageID) *optional*
For templates that accept a thumbnail image.

stripImage (imageID) *optional*
For templates that accept a strip image.

logoImage (imageID) *optional*
For templates that accept a logo image.

footerImage (imageID) *optional*
For templates that accept a footer image.

backgroundImage (imageID) *optional*
For templates that accept a background image.

iconImage (imageID) *optional*
For templates that accept an icon image.

INSTALL PASSTHRU PARAMETERS

The latitude and longitude of the device when the pass was installed. Defaults to the server making the API call.

installLatitude (float) *optional*
The latitude of the location.

installLongitude (float) *optional*
The longitude of the location.

locationDeclined (bool) *optional*
`true` if user declined to give their location when installing a pass.

installIP (string) *optional*
The IP address of the device.

PASS METHODS

Issue Pass

APP PARAMETERS

The app that is associated with a pass.

associatedStoreIdentifiers (comma separated ints) *optional*
A list of iTunes Store item identifiers (Adam IDs) for the associated apps. This list will contain all the compatible apps for this device. The app will open or the App Store will be opened to this app's page.

appLaunchURL (string) *optional*
A URL passed to the associated app when launching.

userInfo (string of JSON) optional (iOS 7)
Custom information for an app. This is not displayed; rather, this can be used to store information about a customer's preferences, e.g., enjoys slow walks on the beach and margaritas.

LOCATION PARAMETERS

A Pass is allowed to store up to 10 locations. The location should be relevant to the Pass. With location(s) stored, as the device holding the pass nears this location(s) the relevant message will appear on the lock screen and allow for convenient access to the Pass. For each location, `address`, `latitude`,`longitude` and `locationAlert` are all required.

location{number}address (string) *required*
An address or other meaningful text to identify the location.

location{number}latitude (float) *required*
The latitude of the location.

location{number}longitude (float) *required*
The longitude of the location.

location{number}locationAlert (string) *required*
The message to be displayed when the device is close to the location.

Issue Pass

BEACON PARAMETERS

A Pass is allowed to store up to 10 iBeacon identities. The Beacon should be relevant to the Pass. With iBeacons(s) stored, as the device holding the pass nears this iBeacon(s) the relevant message will appear on the lock screen and allow for convenient access to the Pass. For each beacon, *name* is required.

beacon{number} (bool) *required (iOS 7)*
`true` for beacon in use.

beacon{number}name (string) *required (iOS 7)*
A UUID or other meaningful text to identify the beacon.

beacon{number}major **(**16-bit unsigned integer**)** *optional (iOS 7)*
The major identifier of a beacon.

beacon{number}minor **(**16-bit unsigned integer**)** *optional (iOS 7)*
The minor identifier of a beacon.

beacon{number}beaconAlert (string) *required (iOS 7)*
The message to be displayed when the device is close to the beacon.

CHOP PASS PARAMETERS

The following parameters only apply to the paid Chop Pass Template and is not a part of the original set of Pass Designer templates. To purchase Chop Pass, please visit this page.

For regular chops:

chopIncrement (int)
Increments the chops by the number provided. Also increments the total chops. If the total of existing chops plus increment value is greater than the total number of chops, then the card is rolled over. If rolled over, increments the chop rewards count.

chopValue (int)
Sets the chops to the number provided. Also increments the total chops and assumes that the card has been rolled over if the new value is less than the existing value. If rolled over, increments the chop rewards count.

Continued on next page.

Issue Pass

CHOP PASS PARAMETERS

Continued from previous page.

For random chops:

chopToggle (int)
Toggles the chop for the number provided – if adding a chop, total chop count in incremented, if removing a chop then the count is decremented.

chopToggle (int)
Toggles the chop for the number provided – if adding a chop, total chop count in incremented, if removing a chop then the count is decremented.

chopString (string)
Takes a sting of O or X of length equal to the total number of chops (E.g. a string 10 chop card with chops 4 and 7 stamped would be OOOXOOXOOO). If all chops are 'X' then chop rewards count is incremented.

For all chop types:

chopTotal (int)
Sets the chop total to the value provided.

chopRewards (int)
Sets the rewards number to the value provided.

PASS METHODS

Issue Pass

EXAMPLES

JSON Request

```
{
  "Check In": "2012-10-06T00:00:00+08:00",
  "Check Out": "2012-10-08T00:00:00+08:00",
  "No. Rooms": "1",
  "No. Guests": "2",
  "Reservation No.": "8864337",
  "Guest Name": "Daniel Allen",
  "Hotel Name": "JW Marriott Hotel Hong
Kong",
  "Hotel Address": "Pacific Place, 88
Queensway, Hong Kong",
  "Hotel Telephone": "+852 2810 8366",
  "Nightly Rate": "HK$ 3,315",
  "relevantDate":
"2012-10-06T00:00:00+08:00",
  "location1address": "Pacific Place, 88
Queensway, Hong Kong",
  "location1latitude": "22.27772",
  "location1longitude": "114.16481",
  "location1locationAlert": "Enjoy your
stay",
  "recoveryEmail": "daniel.allen@example.com"
}
```

PASS METHODS

Issue Pass

```
<?xml version="1.0" encoding="UTF-8"?>
<PassKit_API_Request>
  <Check·In>2012-10-06T00:00:00+08:00</
Check·In>
  <Check·Out>2012-10-08T00:00:00+08:00</
Check·Out>
  <No.·Rooms>1</No.·Rooms>
  <No.·Guests>2</No.·Guests>
  <Reservation·No.>8864337</Reservation·No.>
  <Guest·Name>Daniel Allen</Guest·Name>
  <Hotel·Name>JW Marriott Hotel Hong Kong</
Hotel·Name>
  <Hotel·Address>Pacific Place, 88 Queensway,
Hong Kong</Hotel·Address>
  <Hotel·Telephone>+852 2810 8366</
Hotel·Telephone>
  <Nightly·Rate>HK$ 3,315</Nightly·Rate>
  <relevantDate>2012-10-06T00:00:00+08:00</
relevantDate>
  <location1address>Pacific Place, 88
Queensway, Hong Kong</location1address>
  <location1latitude>22.27772</
location1latitude>
  <location1longitude>114.16481</
location1longitude>
  <location1locationAlert>Enjoy your stay</
location1locationAlert>
  <recoveryEmail>daniel.allen@example.com</
recoveryEmail>
</PassKit_API_Request>
```

XML Request

Note that for XML requests, spaces in tag names must be replaced by a middle dot (·), and tags starting with numbers (and other invalid characters) are prefixed with two underscores (__).

```
# GET
curl --digest -u {APIKey}:{APISecret} \
    "https://api.passkit.com/v1/pass/issue/
template/{templateName}/\
    "?{parameter}={value}&{parameter}={value}"

# POST (Key/Value)
curl --digest -u {APIKey}:{APISecret} \
    -d "{parameter}={value};{parameter}
={value}"
    "https://api.passkit.com/v1/pass/issue/
template/{templateName}"

### POST (File)
curl --digest -u {APIKey}:{APISecret} -F
@{filename} -X POST \
    "https://api.passkit.com/v1/pass/issue/
template/{templateName}"
```

cURL Syntax

Issue Pass

Response

RESPONSE PARAMETERS

success (string)
`true` on success (will not be present on error).

serial (string)
The serial number of the issued pass.

url (string)
The URL for downloading the issued pass.

passbookSerial (string)
The PassBook serial of the pass.

shareID (string)
The share-id of the pass.

uniqueID (string)
The unique pass-id of the the pass

EXAMPLES

JSON Response

```
{
  "success": true,
  "serial": "8618513507140783",
  "url": "https://r.pass.is/9aCKbhwAjRrS",
  "passbookSerial": "14QKnRzuAKpjng9jjMf0i",
  "shareID": "4H3NCpqtCKD3kgG4fSE0M5",
  "uniqueID": "9aCKbhwAjRrS"
}
```

XML Response

```
<PassKit_API_Response>
<success>1</success>
<serial>7142697882935140</
serial><url>https://r.pass.is/ednDGymatydV</
url>
<passbookSerial>W7ecEMnwYJjMjWLWfExD</
passbookSerial>
<shareID>7dLsbGMSRrEOG8qZhTaWtK</shareID>
<uniqueID>ednDGymatydV</uniqueID>
</PassKit_API_Response>
```

PASS METHODS

Issue Pass

Implementation Examples

PHP Implementation Example

```php
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My template name";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

// Set data
$data["Field1"] = "Field 1 Value";
$data["Field2"] = "Field 2 Value";

$result = $pk->issuePass($templateName,
$data);

// Do something with result
print_r($result);
```

C# Implementation Example

Continued on next page.

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        string template_name = "My template";
```

PASS METHODS

Issue Pass

```
        // Initialize new instance of PassKit
API wrapper
        PassKit pk = new PassKit(apiAccount,
apiSecret);

        Dictionary<string, string> data = new
Dictionary<string, string>();
        data["field 1"] = "Field 1 data";
        data["field 2"] = "Field 2 data";
        data["field 3"] = "Field 3 data";

        PassKitResponse result =
pk.IssuePass(template_name, data);

        // Do something with result
    }
}
```

Continued from previous page.

Note

The PHP and C# examples make use of our helper classes. The helper classes have been written to make access to the PassKit API even easier. The examples can be downloaded at our Google Code Repository.

UPDATE PASS

Description

URI (by Template & Serial)	https://api.passkit.com/v1/pass/update/template/{templateName}/serial/{serialNumber} /push (optional) /? format=xml (optional)a
URI (by Pass ID)	https://api.passkit.com/v1/pass/update/passid/{passId} /push (optional) /? format=xml (optional)
Verb	GET,POST,PUT
Auth	Yes

Updates passes with parameters returned by Get Template Field Names, plus a relevant date, locations, and beacons.

Requests are issued as a GET or POST request using key/value pairs, or by PUTing a JSON or XML file. Returns the deviceID of each updated pass and the number of passes updated. For example, if there were 4 issued passes and 2 had notifications active, 2 deviceIDs and 4 (passes updated) would be returned.

To update existing passes, append /push to the URI.

URI PARAMETERS

Update by template & serial:

templateName (string)
The name of the template for the pass to be updated.

serialNumber (string)
The serial number of the pass to be updated.

Update by Unique Pass ID:

passId (string)
The unique id of the pass to be updated.

Update Pass

Request

GENERIC PARAMETERS

{fieldName} (mixed) *optional*
The value for a dynamic field within the template. The submitted data type must match the data type of the field, as specified in the template. Date values must be in ISO 8601 format. If not provided, the pass will return the default value for the field or an empty string or zero value for number and currency fields if no default is present.

{fieldName}_changeMessage (string) *optional*
The text of the notification message that will be displayed when the field value changes. Include %@ in the string to include the new value in the notification.

{fieldName}_label (string) *optional*
The value of a dynamic label field. Label values are always treated as text.

recoveryEmail (string) *optional*
The email address used to reinstall a pass. An invalid email address will return an error.

barcodeContent (string) *optional*
For templates that allow the barcode encoded content to be changed. Using %@ in the string will insert the pass serial number into the encoded content.

barcodeAltContent (string) *optional*
For templates that allow the barcode alternative content to be changed. Using %@ in the string will insert the pass serial number into the message.

relevantDate (string) *optional*
An ISO 8601 formatted date for pass types that support a Relevant Date.

expirationDate (ISO 8601 date) *optional (iOS 7)*
The expiration date of the pass.

voided (bool) optional (iOS 7)
If set to `true`, this pass is voided. Useful for single-use coupons.

PASS METHODS

Update Pass

COLOUR PARAMETERS

labelColor / labelColour (string) *optional*
The label color. Accepts a 6 digit HEX color code.

foregroundColor / foregroundColour (string) *optional*
The foreground color. Accepts a 6 digit HEX color code.

backgroundColor / backgroundColour (string) *optional*
The background color. Accepts a 6 digit HEX color code.

groupingIdentifier (string) *optional (iOS 7)*
Identifier for grouping event tickets and boarding passes with the same style, pass type identifier, and grouping identifiers. For example, multiple passes for a big event can be grouped together.

IMAGE PARAMETERS

The images on each pass are set by the image parameters. Each parameter accepts an imageID returned by the Upload Image method. ImageIDs can be checked with the Get Image Data method.

thumbnailImage (imageID) *optional*
For templates that accept a thumbnail image.

stripImage (imageID) *optional*
For templates that accept a strip image.

logoImage (imageID) *optional*
For templates that accept a logo image.

footerImage (imageID) *optional*
For templates that accept a footer image.

backgroundImage (imageID) *optional*
For templates that accept a background image.

iconImage (imageID) *optional*
For templates that accept an icon image.

PASS METHODS

Update Pass

APP PARAMETERS

The app that is associated with a pass.

associatedStoreIdentifiers (comma separated ints) *optional*
A list of iTunes Store item identifiers (Adam IDs) for the associated apps. This list will contain all the compatible apps for this device. The app will open or the App Store will be opened to this app's page.

appLaunchURL (string) *optional*
A URL passed to the associated app when launching.

userInfo (string of JSON) optional (iOS 7)
Custom information for an app. This is not displayed; rather, this can be used to store information about a customer's preferences, e.g., enjoys slow walks on the beach and margaritas.

LOCATION PARAMETERS

A Pass is allowed to store up to 10 locations. The location should be relevant to the Pass. With location(s) stored, as the device holding the pass nears this location(s) the relevant message will appear on the lock screen and allow for convenient access to the Pass. For each location, *address*, *latitude*,*longitude* and *locationAlert* are all required.

location{number}address (string) *required*
An address or other meaningful text to identify the location.

location{number}latitude (float) *required*
The latitude of the location.

location{number}longitude (float) *required*
The longitude of the location.

location{number}locationAlert (string) *required*
The message to be displayed when the device is close to the location.

Update Pass

BEACON PARAMETERS

A Pass is allowed to store up to 10 iBeacon identities. The Beacon should be relevant to the Pass. With iBeacons(s) stored, as the device holding the pass nears this iBeacon(s) the relevant message will appear on the lock screen and allow for convenient access to the Pass. For each beacon, *number*, *name* and *beaconAlert* are all required.

beacon{number} (bool) *required (iOS 7)*
Set to true for beacon in use.

beacon{number}name (string) *required (iOS 7)*
A UUID or other meaningful text to identify the beacon.

beacon{number}major **(**16-bit unsigned integer**)** *optional (iOS 7)*
The major identifier of a beacon.

beacon{number}minor **(**16-bit unsigned integer**)** *optional (iOS 7)*
The minor identifier of a beacon.

beacon{number}beaconAlert (string) *required (iOS 7)*
The message to be displayed when the device is close to the beacon.

CHOP PASS PARAMETERS

The following parameters only apply to the paid Chop Pass Template and is not a part of the original set of Pass Designer templates. To purchase Chop Pass, please visit this page.

For regular chops:

chopIncrement (int)
Increments the chops by the number provided. Also increments the total chops. If the total of existing chops plus increment value is greater than the total number of chops, then the card is rolled over. If rolled over, increments the chop rewards count.

Continued on next page.

chopValue (int)
Sets the chops to the number provided. Also increments the total chops and assumes that the card has been rolled over if the new value is less than the existing value. If rolled over, increments the chop rewards count.

Update Pass

CHOP PASS PARAMETERS

Continued from previous page.

For random chops:

chopToggle (int)
Toggles the chop for the number provided – if adding a chop, total chop count in incremented, if removing a chop then the count is decremented.

chopToggle (int)
Toggles the chop for the number provided – if adding a chop, total chop count in incremented, if removing a chop then the count is decremented.

chopString (string)
Takes a sting of O or X of length equal to the total number of chops (E.g. a string 10 chop card with chops 4 and 7 stamped would be OOOXOOXOOO). If all chops are 'X' then chop rewards count is incremented.

For all chop types:

chopTotal (int)
Sets the chop total to the value provided.

chopRewards (int)
Sets the rewards number to the value provided.

PASS METHODS

Update Pass

EXAMPLES

JSON Request

```
{
   "Check In": "2012-10-07T00:00:00+08:00",
   "Check In_changeMessage" : "Your check in
date has changed to %@",
   "Check Out": "2012-10-08T00:00:00+08:00",
   "No. Rooms": "1",
   "No. Guests": "2",
   "Reservation No.": "8864337",
   "Guest Name": "Daniel Allen",
   "Hotel Name": "JW Marriott Hotel Hong
Kong",
   "Hotel Address": "Pacific Place, 88
Queensway, Hong Kong",
   "Hotel Telephone": "+852 2810 8366",
   "Nightly Rate": "HK$ 3,315",
   "relevantDate":
"2012-10-06T00:00:00+08:00",
   "location1address": "Pacific Place, 88
Queensway, Hong Kong",
   "location1latitude": "22.27772",
   "location1longitude": "114.16481",
   "location1locationAlert": "Enjoy your
stay"
}
```

XML Request

Note that for XML requests, spaces in tag names must be replaced by a middle dot (·), and tags starting with numbers (and other invalid characters) are prefixed with two underscores (__).

```
<!--?xml version="1.0" encoding="UTF-8"?-->
<PassKit_API_Request>
    <Check·In>2012-10-06T00:00:00+08:00
    <Check·In_changeMessage>Your check in
date has changed to %@
    <Check·Out>2012-10-08T00:00:00+08:00
    <No.·Rooms>1
    <No.·Guests>2
    <Reservation·No.>8864337
    <Guest·Name>Daniel Allen
    <Hotel·Name>JW Marriott Hotel Hong Kong
    <Hotel·Address>Pacific Place, 88
Queensway, Hong Kong
    <Hotel·Telephone>+852 2810 8366
    <Nightly·Rate>HK$ 3,315
    2012-10-06T00:00:00+08:00
    Pacific Place, 88 Queensway, Hong Kong
    22.27772
    114.16481
    Enjoy your stay
```

PASS METHODS

Update Pass

cURL Syntax

```
# GET by Template & Serial
curl --digest -u {APIKey}:{APISecret} \
    "https://api.passkit.com/v1/pass/update/
template/{templateName}/"
    "serial/{serialNumber}/push?{parameter}
={value}&{parameter}={value}"

# GET by Pass ID
curl --digest -u {APIKey}:{APISecret} \
    "https://api.passkit.com/v1/pass/update/
passId/{passId}/push"
    "?{parameter}={value}&{parameter}
={value}"

# POST by Template & Serial
curl --digest -u {APIKey}:{APISecret} -d
"{parameter}={value};{parameter}={value}" \
    "https://api.passkit.com/v1/pass/update/
template/{templateName}/serial/
{serialNumber}/push"

# POST by Pass ID
curl --digest -u {APIKey}:{APISecret} -d
"{parameter}={value};{parameter}={value}" \
    "https://api.passkit.com/v1/pass/update/
passId/{passId}/push"

# PUT by Template & Serial
curl --digest -u {APIKey}:{APISecret} -d
@{filename} -X PUT \
    "https://api.passkit.com/v1/pass/update/
template/{templateName}/serial/
{serialNumber}/push"

# PUT by Pass ID
curl --digest -u {APIKey}:{APISecret} -d
@{filename} -X PUT \
    "https://api.passkit.com/v1/pass/update/
passId/{passId}/push"
```

Update Pass

Response

RESPONSE PARAMETERS

success (bool)
True on success (will not be present on error).

error (string)
Will be returned on error (will not be present on success). Will contain the error message.

device_ids (array)
The deviceIDs that will receive this update via push.

passes (integer)
The number of issued passes updated in the database.

EXAMPLES

JSON Response

```
{
    "success": true,
    "device_ids": {
    "device_1":
"a7fc7ff20ef26c95808ab820e2aa797e0c590df760fe
bd974051287779ec69f5"
    },
    "passes": 2
}
```

XML Response

```
<PassKit_API_Response>
        <success>1</success>
        <device_ids>
                <device_1>a7fc7ff20ef26c95808
ab820e2aa797e0c590df760febd974051287779ec69f5
</device_1>
        </device_ids>
        <passes>2</passes>
</PassKit_API_Response>
```

PASS METHODS

Update Pass

Implementation Examples

```php
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My template name";
$serialNumber = "123412341234";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

// Set data
$data["Field1"] = "Field 1 Value";
$data["Field2"] = "Field 2 Value";

// With /push - automatically pushed update
to the device
$result = $pk-
>updatePassByTemplateSerial($templateName,
$serialNumber, $data);

// Without /push - no automatic push
$result = $pk-
>updatePassByTemplateSerial($templateName,
$serialNumber, $data, false);

// Do something with result
print_r($result);
```

PHP Implementation Example

Template & Serial Method

Note

The PHP and C# examples make use of our helper classes. The helper classes have been written to make access to the PassKit API even easier. The examples can be downloaded at our Google Code Repository.

```php
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$passId = "4hf871623gh4";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

// Set data
```

PHP Implementation Example

Unique Pass ID Method

PASS METHODS

Update Pass

```
$data["Field1"] = "Field 1 Value";
$data["Field2"] = "Field 2 Value";

// With /push - automatically pushed update
to the device
$result = $pk->updatePassById($passId,
$data);

// Without /push - no automatic push
$result = $pk->updatePassById($passId, $data,
false);

// Do something with result
print_r($result);
```

Continued from previous page.

C# Implementation Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        string template_name = "My template";
        string pass_serial = "Pass serial";
        string unique_pass_id = "Unique Pass
ID";

        // Initialize new instance of PassKit
API wrapper
        PassKit pk = new PassKit(apiAccount,
apiSecret);
```

Continued on next page.

PASS METHODS

Update Pass

```
        Dictionary<string, string> data = new
Dictionary<string, string>();
        data["field 1"] = "Field 1 data";
        data["field 2"] = "Field 2 data";
        data["field 3"] = "Field 3 data";

        // Example 1: update pass via
Template + Serial
        PassKitResponse result =
pk.UpdatePass(template_name, pass_serial,
data, true);
        // Do something with result

        // Example 2: update pass via Pass ID
        PassKitResponse result2 =
pk.UpdatePass(unique_pass_id, data, true);
        // Do something with result
    }
}
```

Continued from previous page.

INVALIDATE PASS

Description

URI (by Template & Serial)	https://api.passkit.com/v1/pass/invalidate/template/{templateName}/serial/{serialNumber} /?format=xml (optional)
URI (by Pass ID)	https://api.passkit.com/v1/pass/invalidate/passid/{passId} /?format=xml (optional)
Verb	GET,POST,PUT
Auth	Yes

Invalidates passes with parameters returned by Get Template Field Names, plus a relevant date, locations, and beacons. Requests are issued as a GET or POST request using key/value pairs, or by PUTing a JSON or XML file.

Returns the deviceID of each invalidated pass and the number of passes invalidated. For example, if there were 4 issued passes and 2 had notifications active, 2 deviceIDs and 4 (passes invalidated) would be returned.

Invalidating a pass first updates its content, then removes it from circulation, locking further updates and manual refreshes. For additional security, there is a request parameter that will remove the barcode from the pass to prevent it from being presented. Note that passes cannot be changed if a device has disabled notifications: invalid passes can visually appear valid.

Invalidated passes remain in the database for analytics purposes but once invalidated, they cannot be reactivated. Relevance alerts (locations, beacons, and relevant date) will continue to function, unless deactivated with removeLocations, removeRelevantDate, or turning off all beacons. Alternatively, for marketing or client engagement purposes, this can be used to specify new dates, locations, beacons, and alert messages to engage the pass holder.

Invalidate Pass

URI PARAMETERS

Update by template & serial:

templateName (string)
The name of the template for the pass to be updated.

serialNumber (string)
The serial number of the pass to be updated.

Update by Unique Pass ID:

passId (string)
The unique id of the pass to be updated.

Request

GENERIC PARAMETERS

{fieldName} (mixed) *optional*
The value for a dynamic field within the template. The submitted data type must match the data type of the field, as specified in the template. Date values must be in ISO 8601 format. If not provided, the pass will return the default value for the field or an empty string or zero value for number and currency fields if no default is present.

Continued on next page.

{fieldName}_changeMessage (string) *optional*
The text of the notification message that will be displayed when the field value changes. Include %@ in the string to include the new value in the notification.

{fieldName}_label (string) *optional*
The value of a dynamic label field. Label values are always treated as text.

recoveryEmail (string) *optional*
The email address used to reinstall a pass. An invalid email address will return an error.

removeBarcode (boolean) *optional*
If set to true, the barcode will be removed from the face of the pass.

Invalidate Pass

GENERIC PARAMETERS

Continued from previous page.

barcodeContent (string) *optional*
For templates that allow the barcode encoded content to be changed. Using %@ in the string will insert the pass serial number into the encoded content.

barcodeAltContent (string) *optional*
For templates that allow the barcode alternative content to be changed. Using %@ in the string will insert the pass serial number into the message.

removeRelevantDate (boolean) *optional*
If set to true, the relevantDate will be removed from the pass. The pass holder will no longer receive time based alerts.

relevantDate (string) *optional*
An ISO 8601 formatted date for pass types that support a Relevant Date.

expirationDate (ISO 8601 date) *optional (iOS 7)*
The expiration date of the pass.

voided (bool) optional (iOS 7)
If set to `true`, this pass is voided. Useful for single-use coupons.

COLOUR PARAMETERS

labelColor / labelColour (string) *optional*
The label color. Accepts a 6 digit HEX color code.

foregroundColor / foregroundColour (string) *optional*
The foreground color. Accepts a 6 digit HEX color code.

backgroundColor / backgroundColour (string) *optional*
The background color. Accepts a 6 digit HEX color code.

groupingIdentifier (string) *optional (iOS 7)*
Identifier for grouping event tickets and boarding passes with the same style, pass type identifier, and grouping identifiers. For example, multiple passes for a big event can be grouped together.

Invalidate Pass

IMAGE PARAMETERS

The images on each pass are set by the image parameters. Each parameter accepts an imageID returned by the Upload Image method. ImageIDs can be checked with the Get Image Data method.

thumbnailImage (imageID) *optional*
For templates that accept a thumbnail image.

stripImage (imageID) *optional*
For templates that accept a strip image.

logoImage (imageID) *optional*
For templates that accept a logo image.

footerImage (imageID) *optional*
For templates that accept a footer image.

backgroundImage (imageID) *optional*
For templates that accept a background image.

iconImage (imageID) *optional*
For templates that accept an icon image.

APP PARAMETERS

The app that is associated with a pass.

associatedStoreIdentifiers (comma separated ints) *optional*
A list of iTunes Store item identifiers (Adam IDs) for the associated apps. This list will contain all the compatible apps for this device. The app will open or the App Store will be opened to this app's page.

appLaunchURL (string) *optional*
A URL passed to the associated app when launching.

userInfo (string of JSON) optional (iOS 7)
Custom information for an app. This is not displayed; rather, this can be used to store information about a customer's preferences, e.g., enjoys slow walks on the beach and margaritas.

Invalidate Pass

LOCATION PARAMETERS

A Pass is allowed to store up to 10 locations. The location should be relevant to the Pass. With location(s) stored, as the device holding the pass nears this location(s) the relevant message will appear on the lock screen and allow for convenient access to the Pass. For each location, `address`, `latitude`,`longitude` and `locationAlert` are all required.

removeLocations (boolean) *optional*
If set to true, all locations will be removed from the pass. The pass holder will no longer receive location based alerts.

location{number}address (string) *required*
An address or other meaningful text to identify the location.

location{number}latitude (float) *required*
The latitude of the location.

location{number}longitude (float) *required*
The longitude of the location.

location{number}locationAlert (string) *required*
The message to be displayed when the device is close to the location.

BEACON PARAMETERS

A Pass is allowed to store up to 10 iBeacon identities. The Beacon should be relevant to the Pass. With iBeacons(s) stored, as the device holding the pass nears this iBeacon(s) the relevant message will appear on the lock screen and allow for convenient access to the Pass. For each beacon, `name` is required.

beacon{number} (bool) *required (iOS 7)*
`true` for beacons in use.

Continued on next page.

beacon{number}name (string) *required (iOS 7)*
A UUID or other meaningful text to identify the beacon.

beacon{number}major (16-bit unsigned integer) *optional (iOS 7)*
The major identifier of a beacon.

beacon{number}minor (16-bit unsigned integer) *optional (iOS 7)*
The minor identifier of a beacon.

Invalidate Pass

Continued from previous page.

beacon{number}beaconAlert (string) *required (iOS 7)*
The message to be displayed when the device is close to the beacon.

CHOP PASS PARAMETERS

The following parameters only apply to the paid Chop Pass Template and is not a part of the original set of Pass Designer templates. To purchase Chop Pass, please visit this page.

For regular chops:

chopIncrement (int)
Increments the chops by the number provided. Also increments the total chops. If the total of existing chops plus increment value is greater than the total number of chops, then the card is rolled over. If rolled over, increments the chop rewards count.

chopValue (int)
Sets the chops to the number provided. Also increments the total chops and assumes that the card has been rolled over if the new value is less than the existing value. If rolled over, increments the chop rewards count.

For random chops:
chopToggle (int)
Toggles the chop for the number provided – if adding a chop, total chop count in incremented, if removing a chop then the count is decremented.

chopToggle (int)
Toggles the chop for the number provided – if adding a chop, total chop count in incremented, if removing a chop then the count is decremented.

chopString (string)
Takes a sting of O or X of length equal to the total number of chops (E.g. a string 10 chop card with chops 4 and 7 stamped would be OOOXOOXOOO). If all chops are 'X' then chop rewards count is incremented.

For all chop types:

chopTotal (int)
Sets the chop total to the value provided.

chopRewards (int)
Sets the rewards number to the value provided.

Invalidate Pass

EXAMPLES

JSON Request

```
{
   "Check In": "2012-10-07T00:00:00+08:00",
   "Check In_changeMessage" : "Your check in
date has changed to %@",
   "Check Out": "2012-10-08T00:00:00+08:00",
   "No. Rooms": "1",
   "No. Guests": "2",
   "Reservation No.": "8864337",
   "Guest Name": "Daniel Allen",
   "Hotel Name": "JW Marriott Hotel Hong
Kong",
   "Hotel Address": "Pacific Place, 88
Queensway, Hong Kong",
   "Hotel Telephone": "+852 2810 8366",
   "Nightly Rate": "HK$ 3,315",
   "removeBarcode": true,
   "removeRelevantDate": true,
   "removeLocations": true
}
```

XML Request

Note that for XML requests, spaces in tag names must be replaced by a middle dot (·), and tags starting with numbers (and other invalid characters) are prefixed with two underscores (__).

```
<?xml version="1.0" encoding="UTF-8"?>
<PassKit_API_Request>
  <Check·In>2012-10-06T00:00:00+08:00</
Check·In>
  <Check·In_changeMessage>Your check in date
has changed to %@</Check·In_changeMessage>
  <Check·Out>2012-10-08T00:00:00+08:00</
Check·Out>
  <No.·Rooms>1</No.·Rooms>
  <No.·Guests>2</No.·Guests>
  <Reservation·No.>8864337</Reservation·No.>
  <Guest·Name>Daniel Allen</Guest·Name>
  <Hotel·Name>JW Marriott Hotel Hong Kong</
Hotel·Name>
<Hotel·Address>Pacific Place, 88 Queensway,
Hong Kong</Hotel·Address>
  <Hotel·Telephone>+852 2810 8366</
Hotel·Telephone>
  <Nightly·Rate>HK$ 3,315</Nightly·Rate>
  <removeBarcode>1</removeBarcode>
  <removeRelevantDate>1</removeRelevantDate>
  <removeLocations>1</removeLocations>
</PassKit_API_Request>
```

PASS METHODS

Invalidate Pass

cURL Syntax

```
# GET by Template & Serial
curl --digest -u {APIKey}:{APISecret} \
    "https://api.passkit.com/v1/pass/
invalidate/template/{templateName}/"
    "serial/{serialNumber}/push?{parameter}
={value}&{parameter}={value}"

# GET by Pass ID
curl --digest -u {APIKey}:{APISecret} \
    "https://api.passkit.com/v1/pass/
invalidate/passid/{passId}/"
    "push?{parameter}={value}&{parameter}
={value}"

# POST by Template & Serial
curl --digest -u {APIKey}:{APISecret} -d
"{parameter}={value};{parameter}={value}" \
    "https://api.passkit.com/v1/pass/
invalidate/template/"
    "{templateName}/serial/{serialNumber}/
push"

# POST by Pass ID
curl --digest -u {APIKey}:{APISecret} -d
"{parameter}={value};{parameter}={value}" \
    "https://api.passkit.com/v1/pass/
invalidate/passid/{passId}/push"

# PUT by Template & Serial
curl --digest -u {APIKey}:{APISecret} -d
@{filename} -X PUT \
    "https://api.passkit.com/v1/pass/
invalidate/template/"
    "{templateName}/serial/{serialNumber}/
push"

# PUT by Pass ID
curl --digest -u {APIKey}:{APISecret} -d
@{filename} -X PUT \
    "https://api.passkit.com/v1/pass/
invalidate/passid/{passId}/push"
```

Invalidate Pass

Request

RESPONSE PARAMETERS

success (string)
`true` on success (will not be present on error).

device_ids (array)
The deviceIDs that will receive this update via push.

passes (integer)
The number of issued passes updated in the database.

EXAMPLES

JSON Response

```
{
   "passes" : 2,
   "success" : true,
   "device_ids" : {
      "device_1" :
"a7fc7ff20ef26c95808ab820e2aa797e0c590df760fe
bd974051287779ec69f5"
   }
}
```

XML Response

```
<PassKit_API_Response>
  <success>1</success>
  <device_ids>
    <device_1>a7fc7ff20ef26c95808ab820e2aa797
e0c590df760febd974051287779ec69f5</device_1>
  </device_ids>
  <passes>2</passes>
</PassKit_API_Response>
```

Invalidate Pass

Implementation Examples

```php
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My template name";
$serialNumber = "123412341234";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

// Set data
$data["Field1"] = "Field 1 Value";
$data["Field2"] = "Field 2 Value";

// Make sure to take of barcode, locations &
relevant date (if required)
$data["removeBarcode"] = 1;
$data["removeLocations"] = 1;
$data["removeRelevantDate"] = 1;

// With /push - automatically pushed update
to the device
$result = $pk-
>invalidatePassByTemplateSerial($templateName
, $serialNumber, $data);

// Without /push - no automatic push
$result = $pk-
>invalidatePassByTemplateSerial($templateName
, $serialNumber, $data, false);

// Do something with result
print_r($result);
```

PHP Implementation Example

Template & Serial Method

Note

The PHP and C# examples make use of our helper classes. The helper classes have been written to make access to the PassKit API even easier. The examples can be downloaded at our Google Code Repository.

C# Implementation Example

Continued on next page.

PASS METHODS

Invalidate Pass

```php
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$passId = "4hf871623gh4";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

// Set data
$data["Field1"] = "Field 1 Value";
$data["Field2"] = "Field 2 Value";

// Make sure to take of barcode, locations &
relevant date (if required)
$data["removeBarcode"] = 1;
$data["removeLocations"] = 1;
$data["removeRelevantDate"] = 1;

// With /push - automatically pushed update
to the device
$result = $pk-
>invalidatePassByPassId($passId, $data);

// Without /push - no automatic push
$result = $pk-
>invalidatePassByPassId($passId, $data,
false);

// Do something with result
print_r($result);
```

PHP Implementation Example

Unique Pass ID Method

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
```

C# Implementation Example

Continued on next page.

PASS METHODS

Invalidate Pass

```
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        string template_name = "My template";
        string pass_serial = "Pass serial";
        string unique_pass_id = "Unique Pass
ID";

        // Initialize new instance of PassKit
API wrapper
        PassKit pk = new PassKit(apiAccount,
apiSecret);

        Dictionary<string, string> data = new
Dictionary<string, string>();
        data["field 1"] = "Field 1 data";
        data["field 2"] = "Field 2 data";
        data["field 3"] = "Field 3 data";
        // remove barcode, relevant date &
locations
        data["removeBarcode"] = "1";
        data["removeLocations"] = "1";
        data["removeRelevantDate"] = "1";

        // Example 1: invalidate via Template
+ Serial
        PassKitResponse result =
pk.InvalidatePass(template_name, pass_serial,
data);
        // Do something with result

        // Example 2: invalidate via Pass ID
        PassKitResponse result2 =
pk.InvalidatePass(unique_pass_id, data);
        // Do something with result
    }
}
```

C# Implementation Example

Continued from previous page.

GET PASS DETAILS (BY TEMPLATE & SERIAL)

Description

URI (by Template & Serial)	https://api.passkit.com/v1/pass/get/template/{templateName}/serial/{serialNumber} /?format=xml (optional)
Verb	GET
Auth	Yes

Gets the pass details based on the template name and serial of the pass. Returns an extensive result of pass meta and field data.

URI PARAMETERS

templateName (string)
The name of the template that the pass is created from.

serialNumber (string)
The serial number of the pass.

Request

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret}
"https://api.passkit.com/v1/pass/get/
template/{templateName}/serial/
{serialNumber}"
```

Response

A successful request should receive the following response:

PASS METHODS

Get Pass Details (by Template & Serial)

JSON Response


```
{
   "templateLastUpdated" :
"2013-02-08T01:02:55+00:00",
   "uniqueID" : "uniqueID",
   "success" : true,
   "totalPasses" : 1,
   "templateName" : "Lesson package",
   "passRecords" : {
      "pass_1" : {
         "passMeta" : {
            "registrationIPCity" : "Central
District",
            "installIPCity" : "Central
District",
            "installIPCountry" : "HK",
            "passbookSerial" :
"passbookSerial",
            "registrationIP" :
"123.123.123.123",
            "lastDataChange" :
"2013-03-02T04:56:47+00:00",
            "passStatus" : "Active",
            "installIP" : "123.123.123.123",
            "deviceIsCurrent" : true,
            "recoveryURL" : "recoveryUrl",
            "registeredDate" :
"2013-03-02T04:56:44+00:00",
            "lastPushRefresh" :
"2013-02-10T14:45:08+00:00",
            "issueDate" :
"2013-02-10T08:19:30+00:00",
            "lastManualRefresh" :
"2013-03-02T04:57:09+00:00",
            "registrationIPCountry" : "HK"
         },
         "passData" : {
            "barcodeContent" :
"barcodeContent",
            "Expiry date" : "2014-02-10",
            "Issue date" : "2013-02-10",
            "Balance" : "8",
            "Student name" : "Test student"
         }
      }
   },
   "serialNumber" : "serialNumber"
}
```

PASS METHODS

Get Pass Details (by Template & Serial)

```
<PassKit_API_Response>
    <success>1</success>
    <serialNumber>serialNumber</serialNumber>
    <templateName>Lesson package</
templateName>
    <uniqueID>uniqueId</uniqueID>
    <templateLastUpdated>2013-02-08T01:02:55+
00:00</templateLastUpdated>
    <totalPasses>1</totalPasses>
    <passRecords>
        <pass_1>
            <passMeta>
                <passStatus>Active</
passStatus>
                <installIP>123.123.123.123</
installIP>
                <registrationIP>123.123.123.1
23</registrationIP>
                <installIPCountry>HK</
installIPCountry>
                <installIPCity>Central
District</installIPCity>
                <registrationIPCountry>HK</
registrationIPCountry>
                <registrationIPCity>Central
District</registrationIPCity>
                <recoveryURL>recoveryUrl</
recoveryURL>
                <issueDate>2013-02-10T08:19:3
0+00:00</issueDate>
                <registeredDate>2013-03-02T04
:56:44+00:00</registeredDate>
                <lastDataChange>2013-03-02T04
:56:47+00:00</lastDataChange>
                <lastPushRefresh>2013-02-10T1
4:45:08+00:00</lastPushRefresh>
                <lastManualRefresh>2013-03-02
T04:57:09+00:00</lastManualRefresh>
                <deviceIsCurrent>1</
deviceIsCurrent>
                <passbookSerial>passbookSeria
l</passbookSerial>
            </passMeta>
            <passData>
                <Student·name>Test student</
Student·name>
                <Balance>8</Balance>
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

Continued on next page.

Get Pass Details (by Template & Serial)

```
<Issue·date>2013-02-10</Issue·date>
                <Expiry·date>2014-02-10</
Expiry·date>
                <barcodeContent>barcodeConten
t</barcodeContent>
            </passData>
        </pass_1>
    </passRecords>
</PassKit_API_Response>
```

Continued from previous page.

Implementation Examples

PHP Implementation Example

```php
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My Template";
$serialNumber = "12341234";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

$result = $pk-
>getPassDetailsByTemplateSerial($templateName
, $serialNumber);

// Do something with result
print_r($result);
```

PASS METHODS

Get Pass Details (by Template & Serial)

C# Implementation Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        string template_name = "My template";
        string pass_serial = "Pass serial";
        string unique_pass_id = "Unique Pass
ID";

        // Initialize new instance of PassKit
API wrapper
        PassKit pk = new PassKit(apiAccount,
apiSecret);

        PassKitResponse result =
pk.GetPassDetails(template_name,
pass_serial);
        // Do something with result
    }
}
```

GET PASS DETAILS (BY PASSID)

Description

URI	https://api.passkit.com/v1/pass/get/passid/{passId} /?format=xml (optional)
Verb	GET
Auth	Yes

Gets the pass details based on the Pass ID. The function returns a result of pass meta and field data.

URI PARAMETERS

passID (string)
The unique ID of the pass.

Request

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret}
https://api.passkit.com/v1/pass/get/passid/{passId} /?
format=xml (optional)
```

Response

A successful request should receive the following response:

JSON Response

```
{
   "templateLastUpdated" :
"2013-02-08T01:02:55+00:00",
   "uniqueID" : "uniqueID",
   "success" : true,
   "totalPasses" : 1,
```

Continued on next page.

PASS METHODS

Get Pass Details (by PassID)

Continued from previous page.


```
   "templateName" : "Lesson package",
   "passRecords" : {
      "pass_1" : {
         "passMeta" : {
            "registrationIPCity" : "Central
District",
            "installIPCity" : "Central
District",
            "installIPCountry" : "HK",
            "passbookSerial" :
"passbookSerial",
            "registrationIP" :
"123.123.123.123",
            "lastDataChange" :
"2013-03-02T04:56:47+00:00",
            "passStatus" : "Active",
            "installIP" : "123.123.123.123",
            "deviceIsCurrent" : true,
            "recoveryURL" : "recoveryUrl",
            "registeredDate" :
"2013-03-02T04:56:44+00:00",
            "lastPushRefresh" :
"2013-02-10T14:45:08+00:00",
            "issueDate" :
"2013-02-10T08:19:30+00:00",
            "lastManualRefresh" :
"2013-03-02T04:57:09+00:00",
            "registrationIPCountry" : "HK"
         },
         "passData" : {
            "barcodeContent" :
"barcodeContent",
            "Expiry date" : "2014-02-10",
            "Issue date" : "2013-02-10",
            "Balance" : "8",
            "Student name" : "Test student"
         }
      }
   },
   "serialNumber" : "serialNumber"
}
```

PASS METHODS

Get Pass Details (by Pass Fields)

```
<PassKit_API_Response>
    <success>1</success>
    <serialNumber>serialNumber</serialNumber>
    <templateName>Lesson package</
templateName>
    <uniqueID>uniqueId</uniqueID>
    <templateLastUpdated>2013-02-08T01:02:55+
00:00</templateLastUpdated>
    <totalPasses>1</totalPasses>
    <passRecords>
        <pass_1>
            <passMeta>
                <passStatus>Active</
passStatus>
                <installIP>123.123.123.123</
installIP>
                <registrationIP>123.123.123.1
23</registrationIP>
                <installIPCountry>HK</
installIPCountry>
                <installIPCity>Central
District</installIPCity>
                <registrationIPCountry>HK</
registrationIPCountry>
                <registrationIPCity>Central
District</registrationIPCity>
                <recoveryURL>recoveryUrl</
recoveryURL>
                <issueDate>2013-02-10T08:19:3
0+00:00</issueDate>
                <registeredDate>2013-03-02T04
:56:44+00:00</registeredDate>
                <lastDataChange>2013-03-02T04
:56:47+00:00</lastDataChange>
                <lastPushRefresh>2013-02-10T1
4:45:08+00:00</lastPushRefresh>
                <lastManualRefresh>2013-03-02
T04:57:09+00:00</lastManualRefresh>
                <deviceIsCurrent>1</
deviceIsCurrent>
                <passbookSerial>passbookSeria
l</passbookSerial>
            </passMeta>
            <passData>
                <Student·name>Test student</
Student·name>
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

Get Pass Details (by Pass Fields)

```
<Balance>8</Balance>
                <Issue·date>2013-02-10</
Issue·date>
                <Expiry·date>2014-02-10</
Expiry·date>
                <barcodeContent>barcodeConten
t</barcodeContent>
            </passData>
        </pass_1>
    </passRecords>
</PassKit_API_Response>
```

Continued from previous page.

Implementation Examples

PHP Implementation Example

```
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$passId = "4hf871623gh4";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

$result = $pk-
>getPassDetailsByPassId($passId);

// Do something with result
print_r($result);
```

PASS METHODS

Get Pass Details (by PassID)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        string template_name = "My template";
        string pass_serial = "Pass serial";
        string unique_pass_id = "Unique Pass
ID";

        // Initialize new instance of PassKit
API wrapper
        PassKit pk = new PassKit(apiAccount,
apiSecret);

        PassKitResponse result =
pk.GetPassDetails(template_name,
pass_serial);
        // Do something with result
    }
}
```

C# Implementation Example

Note

The PHP and C# examples make use of our helper classes. The helper classes have been written to make access to the PassKit API even easier. The examples can be downloaded at our Google Code Repository.

GET PASS DETAILS (BY PASS FIELDS)

Description

URI	http://api.passkit.com/v1/pass/get/template/{template name}/select/{field name}/{value}/{fieldname2}/{value2}/etc.
Verb	GET
Auth	Yes

Gets passes in template where the fields match the values specified. The function returns a result of pass meta and field data.

URI PARAMETERS

templateName (string)
The name of the template that the pass is created from.

Request

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret}
https://api.passkit.com/v1/pass/get/template/
{template name}/select/{field name}/{value}/
{fieldname2}/{value2}/etc.
```

Response

A successful request should receive the following response:

JSON Response

Continued on next page.

```
{
   "templateLastUpdated" :
"2013-02-08T01:02:55+00:00",
   "uniqueID" : "uniqueID",
   "success" : true,
   "totalPasses" : 1,
```

PASS METHODS

Get Pass Details (by Pass Fields)

Continued from previous page.


```
   "templateName" : "Lesson package",
   "passRecords" : {
      "pass_1" : {
         "passMeta" : {
            "registrationIPCity" : "Central
District",
            "installIPCity" : "Central
District",
            "installIPCountry" : "HK",
            "passbookSerial" :
"passbookSerial",
            "registrationIP" :
"123.123.123.123",
            "lastDataChange" :
"2013-03-02T04:56:47+00:00",
            "passStatus" : "Active",
            "installIP" : "123.123.123.123",
            "deviceIsCurrent" : true,
            "recoveryURL" : "recoveryUrl",
            "registeredDate" :
"2013-03-02T04:56:44+00:00",
            "lastPushRefresh" :
"2013-02-10T14:45:08+00:00",
            "issueDate" :
"2013-02-10T08:19:30+00:00",
            "lastManualRefresh" :
"2013-03-02T04:57:09+00:00",
            "registrationIPCountry" : "HK"
         },
         "passData" : {
            "barcodeContent" :
"barcodeContent",
            "Expiry date" : "2014-02-10",
            "Issue date" : "2013-02-10",
            "Balance" : "8",
            "Student name" : "Test student"
         }
      }
   },
   "serialNumber" : "serialNumber"
}
```

PASS METHODS

Get Pass Details (by Pass Fields)

```
<PassKit_API_Response>
    <success>1</success>
    <serialNumber>serialNumber</serialNumber>
    <templateName>Lesson package</
templateName>
    <uniqueID>uniqueId</uniqueID>
    <templateLastUpdated>2013-02-08T01:02:55+
00:00</templateLastUpdated>
    <totalPasses>1</totalPasses>
    <passRecords>
        <pass_1>
            <passMeta>
                <passStatus>Active</
passStatus>
                <installIP>123.123.123.123</
installIP>
                <registrationIP>123.123.123.1
23</registrationIP>
                <installIPCountry>HK</
installIPCountry>
                <installIPCity>Central
District</installIPCity>
                <registrationIPCountry>HK</
registrationIPCountry>
                <registrationIPCity>Central
District</registrationIPCity>
                <recoveryURL>recoveryUrl</
recoveryURL>
                <issueDate>2013-02-10T08:19:3
0+00:00</issueDate>
                <registeredDate>2013-03-02T04
:56:44+00:00</registeredDate>
                <lastDataChange>2013-03-02T04
:56:47+00:00</lastDataChange>
                <lastPushRefresh>2013-02-10T1
4:45:08+00:00</lastPushRefresh>
                <lastManualRefresh>2013-03-02
T04:57:09+00:00</lastManualRefresh>
                <deviceIsCurrent>1</
deviceIsCurrent>
                <passbookSerial>passbookSeria
l</passbookSerial>
            </passMeta>
            <passData>
                <Student·name>Test student</
Student·name>
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

PASS METHODS

Get Pass Details (by Pass Fields)

```
                <Balance>8</Balance>
                <Issue·date>2013-02-10</
Issue·date>
                <Expiry·date>2014-02-10</
Expiry·date>
                <barcodeContent>barcodeConten
t</barcodeContent>
            </passData>
        </pass_1>
    </passRecords>
</PassKit_API_Response>
```

Continued from previous page.

GET PASS BY SHAREID

Description

URI (by ShareID)	https://api.passkit.com/v1/pass/shareid/{shareId} /full (optional) /?format=xml (optional)
Verb	GET
Auth	Yes

Performs a look-up based upon the share ID of the pass. The share ID is unique identifier for the pass used to determine parent/child relationships between passes.

Returns the unique pass ID for the pass.

URI PARAMETERS

shareID (string)
The share ID of the pass.

Request

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret}
"https://api.passkit.com/v1/pass/shareid/
{shareId}"
```

Response

RESPONSE PARAMETERS

success (string)
`true` on success (will not be present on error).

uniqueId (array)
The unique pass ID of the pass.

Get Pass by ShareID

EXAMPLES

A successful request should receive the following response:

```
{
   "uniqueID" : "uniqueId",
   "success" : true
}
```

JSON Response

```
<PassKit_API_Response>
    <success>1</success>
    <uniqueID>uniqueId</uniqueID>
</PassKit_API_Response>
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

Implementation Examples

```
// Include the wrapper class
    include_once("class-PassKit-v2.php");

    // Set required variables
    $apiKey = "My API key";
    $apiSecret = "My API secret";
    $shareId = "123412344534";

    // Create new PassKit object
    $pk = new PassKit($apiKey, $apiSecret);

    $result = $pk-
>getPassIdByShareId($passId);

    // Do something with result
    print_r($result);
```

PHP Implementation Example

Note: The PHP and C# examples make use of our SDK classes. The SDK classes have been written to make access to the PassKit API even easier. The SDK's can be downloaded at our Google Code Repository.

BATCH ISSUE PASS

Description

URI	https://api.passkit.com/v1/pass/issue/batch/template/{templateName} /?format=xml (optional)
Verb	POST
Auth	Yes

Issues batches of passes with parameters returned by Get Template Field Names, plus a relevant date, locations, and beacons. Returns the serial number and URL for each pass issued.

URI PARAMETERS

templateName (string)
The name of the template from which passes will be created.

Request

GENERIC PARAMETERS

{fieldName} (mixed) *optional*
The value for a dynamic field within the template. The submitted data type must match the data type of the field, as specified in the template. Date values must be in ISO 8601 format. If not provided, the pass will return the default value for the field or an empty string or zero value for number and currency fields if no default is present.

Continued on next page.

{fieldName}_changeMessage (string) *optional*
The text of the notification message that will be displayed when the field value changes. Include %@ in the string to include the new value in the notification.

{fieldName}_label (string) *optional*
The value of a dynamic label field. Label values are always treated as text.

serialNumber (string)
The serial number of the pass to be updated.

Batch Issue Pass

GENERIC PARAMETERS

Continued from previous page.

recoveryEmail (string) *optional*
The email address used to reinstall a pass. An invalid email address will return an error.

barcodeContent (string) *optional*
For templates that allow the barcode encoded content to be changed. Using %@ in the string will insert the pass serial number into the encoded content.

barcodeAltContent (string) *optional*
For templates that allow the barcode alternative content to be changed. Using %@ in the string will insert the pass serial number into the message.

relevantDate (string) *optional*
An ISO 8601 formatted date for pass types that support a Relevant Date.

COLOUR PARAMETERS

labelColor / labelColour (string) *optional*
The label color. Accepts a 6 digit HEX color code.

foregroundColor / foregroundColour (string) *optional*
The foreground color. Accepts a 6 digit HEX color code.

backgroundColor / backgroundColour (string) *optional*
The background color. Accepts a 6 digit HEX color code.

IMAGE PARAMETERS

The images on each pass are set by the image parameters. Each parameter accepts an imageID returned by the Upload Image method. ImageIDs can be checked with the Get Image Data method.

Continued on next page.

thumbnailImage (imageID) *optional*
For templates that accept a thumbnail image.

stripImage (imageID) *optional*
For templates that accept a strip image.

logoImage (imageID) *optional*
For templates that accept a logo image.

footerImage (imageID) *optional*
For templates that accept a footer image.

Batch Issue Pass

IMAGE PARAMETERS

Continued from previous page.

backgroundImage (imageID) *optional*
For templates that accept a background image.

iconImage (imageID) *optional*
For templates that accept an icon image.

INSTALL LOCATION PARAMETERS

The latitude and longitude of the device when the pass was installed. Defaults to the server making the API call.

installLatitude (float) *optional*
The latitude of the location.

installLongitude (float) *optional*
The longitude of the location.

locationDeclined (bool) *optional*
`true` if the user declined to give their location data when installing a pass.

installIP (string) *optional*
The IP address of the device.

LOCATION PARAMETERS

A Pass is allowed to store up to 10 locations. The location should be relevant to the Pass. With location(s) stored, as the device holding the pass nears this location(s) the relevant message will appear on the lock screen and allow for convenient access to the Pass. For each location, `address`, `latitude`,`longitude` and `locationAlert` are all required.

location{number}address (string) *required*
An address or other meaningful text to identify the location.

location{number}latitude (float) *required*
The latitude of the location.

location{number}longitude (float) *required*
The longitude of the location.

location{number}locationAlert (string) *required*
The message to be displayed when the device is close to the location.

Batch Issue Pass

BEACON PARAMETERS

A Pass is allowed to store up to 10 iBeacon identities. The Beacon should be relevant to the Pass. With iBeacons(s) stored, as the device holding the pass nears this iBeacon(s) the relevant message will appear on the lock screen and allow for convenient access to the Pass. For each beacon, `name` is required.

beacon{number} (bool) *required*
`true` for beacon in use.

beacon{number}name (string) *required*
A UUID or other meaningful text to identify the beacon.

beacon{number}major (16-bit unsigned integer) *optional*
The major identifier of a beacon.

beacon{number}minor (16-bit unsigned integer) *optional*
The minor identifier of a beacon.

beacon{number}beaconAlert (string) *required*
The message to be displayed when the device is close to the beacon.

EXAMPLES

Requests should be submitted as an array of passes, each with a unique key (e.g. pass_1, pass_2, etc.), with each key each containing an array of key/value pairs.

JSON Request

Continued on next page.

```
{
   "pass_1": {
      "Check In":
"2012-10-06T00:00:00+08:00",
      "Check Out":
"2012-10-08T00:00:00+08:00",
      "No. Rooms": "1",
      "No. Guests": "2",
      "Reservation No.": "8864337",
      "Guest Name": "Daniel Allen",
      "Hotel Name": "JW Marriott Hotel Hong
Kong",
      "Hotel Address": "Pacific Place, 88
Queensway, Hong Kong",
      "Hotel Telephone": "+852 2810 8366",
      "Nightly Rate": "HK$ 3,315",
      "relevantDate":
"2012-10-06T00:00:00+08:00",
      "location1address": "Pacific Place, 88
Queensway, Hong Kong",
```

Batch Issue Pass

Continued from previous page.

```
"location1latitude": "22.27772",
      "location1longitude": "114.16481",
      "location1locationAlert": "Enjoy your
stay",
      "recoveryEmail":
"daniel.allen@example.com"
   },
   "pass_2": {
      "Check In":
"2012-10-14T00:00:00+09:00",
      "Check Out":
"2012-10-15T00:00:00+09:00",
      "No. Rooms": "1",
      "No. Guests": "1",
      "Reservation No.": "9423554",
      "Guest Name": "Wendy Chan",
      "Hotel Name": "Grand Hyatt Seoul",
      "Hotel Address": "322 Sowol-ro,
Yongsan-gu, Seoul, South Korea 140-738",
      "Hotel Telephone": "+82 2 797 1234",
      "Nightly Rate": "KRW 305,000",
      "relevantDate":
"2012-10-14T00:00:00+09:00",
      "location1address": "322 Sowol-ro,
Yongsan-gu, Seoul, South Korea 140-738",
      "location1latitude": "37.539448",
      "location1longitude": "126.997193",
      "location1locationAlert": "Enjoy your
stay",
      "recoveryEmail":
"wendy.chan@example.com"
   }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<PassKit_API_Request>
  <pass_1>
    <Check·In>2012-10-06T00:00:00+08:00</
Check·In>
    <Check·Out>2012-10-08T00:00:00+08:00</
Check·Out>
    <No.·Rooms>1</No.·Rooms>
    <No.·Guests>2</No.·Guests>
    <Reservation·No.>8864337</
Reservation·No.>
    <Guest·Name>Daniel Allen</Guest·Name>
```

XML Request

Note that for XML requests, spaces in tag names must be replaced by a middle dot (·), and tags starting with numbers (and other invalid characters) are prefixed with two underscores (__).

Continued on next page.

PASS METHODS

Batch Issue Pass

Continued from previous page.

```
<Hotel·Name>JW Marriott Hotel Hong Kong</
Hotel·Name>
    <Hotel·Address>Pacific Place, 88
Queensway, Hong Kong</Hotel·Address>
    <Hotel·Telephone>+852 2810 8366</
Hotel·Telephone>
    <Nightly·Rate>HK$ 3,315</Nightly·Rate>
    <relevantDate>2012-10-06T00:00:00+08:00</
relevantDate>
    <location1address>Pacific Place, 88
Queensway, Hong Kong</location1address>
    <location1latitude>22.27772</
location1latitude>
    <location1longitude>114.16481</
location1longitude>
    <location1locationAlert>Enjoy your stay</
location1locationAlert>
    <recoveryEmail>daniel.allen@example.com</
recoveryEmail>
  </pass_1>
  <pass_2>
    <Check·In>2012-10-14T00:00:00+09:00</
Check·In>
    <Check·Out>2012-10-15T00:00:00+09:00</
Check·Out>
    <No.·Rooms>1</No.·Rooms>
    <No.·Guests>1</No.·Guests>
    <Reservation·No.>9423554</
Reservation·No.>
    <Guest·Name>Wendy Chan</Guest·Name>
    <Hotel·Name>Grand Hyatt Seoul</
Hotel·Name>
    <Hotel·Address>322 Sowol-ro, Yongsan-gu,
Seoul, South Korea 140-738</Hotel·Address>
    <Hotel·Telephone>+82 2 797 1234</
Hotel·Telephone>
    <Nightly·Rate>KRW 305,000</Nightly·Rate>
    <relevantDate>2012-10-14T00:00:00+09:00</
relevantDate>
    <location1address>322 Sowol-ro, Yongsan-
gu, Seoul, South Korea 140-738</
location1address>
    <location1latitude>37.539448</
location1latitude>
    <location1longitude>126.997193</
location1longitude>
    <location1locationAlert>Enjoy your stay</
location1locationAlert>
    <recoveryEmail>wendy.chan@example.com</
recoveryEmail>
  </pass_2>
</PassKit_API_Request>
```

Batch Issue Pass

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret} -F
@{filename} -X POST
"https://api.passkit.com/v1/pass/issue/batch/
template/{templateName}"
```

Response

RESPONSE PARAMETERS

success (string)
`true` on success (will not be present on error).

passes (array)
An array containing data for each pass issued.

serial (string)
The serial number of the issued pass.

url (string)
The URL for downloading the issued pass.

EXAMPLES

JSON Response

```
{
   "passes" : {
      "pass_1" : {
         "serial" : "0905188361315670",
         "url" : "https://r.pass.is/
DXerKmGmXyrx"
      },
      "pass_2" : {
         "serial" : "0236642194858366",
         "url" : "https://r.pass.is/
B84Un7iu8xvo"
      }
   },
   "success" : true
}
```

PASS METHODS

Batch Issue Pass

XML Response

```
<PassKit_API_Response>
  <success>1</success>
  <passes>
    <pass_1>
      <serial>0586151635614775</serial>
      <url>https://r.pass.is/JNkpa3eMLVsg</
url>
    </pass_1>
    <pass_2>
      <serial>0790081069148314</serial>
      <url>https://r.pass.is/HutloddK8KGt</
url>
    </pass_2>
  </passes>
</PassKit_API_Response>
```

IMAGE INFORMATION

Uploading Your Images

Use the below dimensions for any image files that you want on the passes. PassKit automatically scales these images to fit within each allotted space, preserving aspect ratio, but are cropped if the aspect ratio is different than their allotted space.

Please note that your images do not have to be exactly these dimensions, as PassKit will automatically crop, scale and optimise them. However, please check in the Pass Designer that the images appear as you want them to. We also recommend that once you have saved your pass template you test the pass design in both iOS6 device and iOS7 device.

IMPORTANT: If you upload images that are smaller in size than the dimensions above they will not appear sharp. We strongly recommend you do not use small images.

Image Type	Position on the Pass	Other Notes	iOS7 Minimum Size (pixels)	iOS6 Minimum Size (pixels)
Background Image	Behind the entire front of the pass	Available on the Event Pass Type. The image is cropped slightly on all sides and blurred	360 x 440	360 x 440
Footer Image	Above the barcode	Available on the Transport Pass Type only	572 x 30	572 x 30
Icon Image	On the lock screen and by apps like Mail when showing an attached Pass	A shine is automatically applied to the icon for you	120 x 120	116 x 116

IMAGE METHODS

Image Information

Image Type	Position on the Pass	Other Notes	iOS7 Minimum Size (pixels)	iOS6 Minimum Size (pixels)
Logo Image	Displayed in the top left corner of the Pass	In iOS6 the maximum width is 620 pixels and in iOS7 the maximum width is 320 pixels. In most cases it should be narrower. If you use the full 620 pixel width note you will not have room for any Logo Text, or Header fields.	350 x 100	620 x 100
Strip Image	Displayed behind the Primary Fields	In iOS6 a shine effect is applied by default but you can turn this off in the Pass Designer. In iOS7 the shine effect is no longer supported.	Event Tickets: 640 x 168 Square Barcode: 640 x 220 All other cases: 640 x 246	Event Tickets: 624 x 168 Square Barcode: 624 x 220 All other cases: 624 x 246

UPLOAD IMAGE

Description

URI	https://api.passkit.com/v1/image/add/{imageType} ?url={url} (optional) /?format=xml (optional)
Verb	POST
Auth	Yes

Uploads images for use in templates and passes. Each image is assigned a unique ID, and is processed according its imageType: `image/jpeg`, `image/png`, or `image/gif`. Returns an imageID.

URI PARAMETERS

imageType (string)
One of the following: background, footer, logo, icon, strip, thumbnail.

url (string) optional
The url of an online image.

Request Contents

The request type must be `Content-Type multipart/form-data`, and the request must contain a field name `image` and a file type of `image/jpeg`, `image/png` or `image/gif`.

cURL Syntax

```
curl --digest -u {APIKey}:{APISecret} -F
"image=@{filename};type={MIME type}" -X POST
\
"https://api.passkit.com/v1/image/add/
{imageType}"
```

IMAGE METHODS

Upload Image

Response

RESPONSE PARAMETERS

success (boolean)
`true` for a successful upload (will not be present on error).

imageID (string)
imageID for use in other methods.

usage (string)
The usage type that image has been processed for.

EXAMPLES

A successful upload should receive the following response:

JSON Response

```
{
   "usage" : "Thumbnail Image",
   "success" : true,
   "imageID" : "UPQCQqyzzLaJEvfq9X0pM"
}
```

XML Response

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

```
<PassKit_API_Response>
    <success>1</success>
    <imageID>UPQCQqyzzLaJEvfq9X0pM</imageID>
    <usage>UPQCQqyzzLaJEvfq9X0pM</usage>
</PassKit_API_Response>
```

IMAGE METHODS

Upload Image

Implementation Examples

```
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My template name";
$pathToLocalImage = "home/myimage.jpg";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

$result = $pk->addImageByLocalFile("strip",
$pathToLocalImage);

// Grab the imageId to use in the issue call
$data["stripImage"] = $result["imageID"];

// Issue new pass with the stripImage
$result = $pk->issuePass($templateName,
$data);

// Do something with result
print_r($result);
```

PHP Implementation Example

Upload a local image file

Note

The PHP and C# examples make use of our helper classes. The helper classes have been written to make access to the PassKit API even easier. The examples can be downloaded at our Google Code Repository.

```
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$templateName = "My template name";
$imageUrl = "http://www.passkit.com/
myimage.jpg";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

$result = $pk->addImageByURL("strip",
$imageUrl);

// Grab the imageId to use in the issue call
$data["stripImage"] = $result["imageID"];
```

PHP Implementation Example

Upload an image file from a URL

Continued on next page.

IMAGE METHODS

Upload Image

```
// Issue new pass with the stripImage
$result = $pk->issuePass($templateName,
$data);

// Do something with result
print_r($result);
```

Continued from previous page.

C# Implementation Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        private string apiAccount =
"apiAccount";
        private string apiSecret =
"apiSecret";

        protected void Page_Load(object
sender, EventArgs e)
        {
            string template_name = "My
template";
            string image_path = @"home/
my_image.jpg";

            // Initialize new instance of
PassKit API wrapper
            PassKit pk = new
PassKit(apiAccount, apiSecret);
```

Continued on next page.

IMAGE METHODS

Upload Image

```
            PassKitResponse uploadResult =
pk.UploadImage(image_path,
        PassKitImageType.strip);

            // Get the image ID from the
result
            if
(uploadResult.response["success"])
            {
                string image_id =
(string)uploadResult.response["imageID"];

                // Use the image id to issue
a pass
                Dictionary<string, string>
data = new Dictionary<string, string>();
                data["stripImage"] =
image_id;
                data["field1"] = "Field 1
data";
                data["field2"] = "Field 2
data";

                PassKitResponse issueResponse
= pk.IssuePass(template_name, data);

                // Do something with response
            }
        }
    }
}
```

Continued from previous page.

GET IMAGE DETAILS

Description

URI	https://api.passkit.com/v1/image/{imageID} /?format=xml (optional)
Verb	GET
Auth	No

Returns data about a particular imageID, specifically the imageTypes it has been processed for. Validates imageIDs for use as a particular imageType.

URI PARAMETERS

imageID (string)
The imageID returned after the image upload.

Request

cURL Syntax

```
curl "https://api.passkit.com/v1/image/{imageID}"
```

Response

RESPONSE PARAMETERS

Continued on next page.

imageID (string)
The image ID requested in the API call.

background (boolean)
`true` if image was processed as a background image, absent otherwise.

footer (boolean)
`true` if image was processed as a footer image, absent otherwise.

Get Image Details

RESPONSE PARAMETERS

Continued from previous page.

logo (boolean)
`true` if image was processed as a logo image, absent otherwise.

icon (boolean)
`true` if image was processed as an icon image, absent otherwise.

strip (boolean)
`true` if image was processed as a strip image, absent otherwise.

thumbnail (boolean)
`true` if image was processed as a thumbnail image, absent otherwise.

EXAMPLES

Returns the following upon successful upload:

JSON Response

```
{
   "imageID" : "3YrwmjaWGrESOM5rVjd6dK",
   "Icon" : "true"
}
```

XML Response

```
<PassKit_API_Response>
  <imageID>3YrwmjaWGrESOM5rVjd6dK</imageID>
  <Icon>true</Icon>
</PassKit_API_Response>
```

Note that for XML responses, spaces in tag names will be replaced by 'middle dot' (·). If a tag starts with an invalid character (e.g. a number), the tag will be prefixed with two underscores (__).

IMAGE METHODS

Get Image Details

Implementation Examples

PHP Implementation Example

```php
// Include the wrapper class
include_once("class-PassKit-v2.php");

// Set required variables
$apiKey = "My API key";
$apiSecret = "My API secret";
$imageId = "3YrwmjaWGrESOM5rVjd6dK";

// Create new PassKit object
$pk = new PassKit($apiKey, $apiSecret);

$result = $pk->getImageDetails($imageId);

// Do something with result
print_r($result);
```

Note

The PHP and C# examples make use of our helper classes. The helper classes have been written to make access to the PassKit API even easier. The examples can be downloaded at our Google Code Repository.

C# Implementation Example

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;
using System.Text;
using System.Reflection;
// Make sure to include PassKitAPIWrapper
class
// (takes care of all communication with the
API)
using PassKitAPIWrapper;

namespace PassKitWebDemo
{
    public partial class Default :
System.Web.UI.Page
    {
        private string apiAccount =
"apiAccount";
        private string apiSecret =
"apiSecret";
```

IMAGE METHODS

Get Image Details

```
        protected void Page_Load(object
sender, EventArgs e)
        {
            string image_id = "image ID";

            // Initialize new instance of
PassKit API wrapper
            PassKit pk = new
PassKit(apiAccount, apiSecret);

            PassKitResponse result =
pk.GetImageData(image_id);

            // Do something with result
        }
    }
}
```

Continued from previous page.