Card Notification Services
API Specification

Confidentiality

The contents of this document, as well as any appendices, supplements, or follow up communications, are confidential and proprietary. They should be shared only with disclosed individuals, and on a need to know basis within Apple, and with disclosed partners.

If you are not certain that you are disclosed, or that you should have access to this document, please stop reading now, and consult the appropriate legal resource within your company before proceeding.

Version Information

Date	Version	Change
4/21/2014	1.0.0	Refactored Transaction Notifications document into Card Notification Services document which now supports messages. Removed specification for FPAN based endpoints, per behavior change of transaction requests. Changed the response of a registration request to return a dictionary containing the authentication token. Additional information added on 3D Secure based transaction identifiers.
6/3/14	1.0.1	Corrected transactionIdentifier derivation. Clarified supported date formats. Improved signature verification documentation, sample payloads, sample code. Documented accepted response status codes. Added Message expiration date field. Documented request headers. Included suggested lengths for relevant data elements.
9/4/14	1.0.2	Added explicit definition for an FPAN based deep linking URL scheme. Clarification of date & time formats. Clarification on device identifier lifecycle, registration authentication tokens for 2nd DPANID registrations, and "best effort" design of de-registration requests. Added information on best-practice use of LastModifiedTag.
10/1/14	1.0.3	Added examples of different scenarios for transaction details responses. Clarified uniqueness of transaction detail responses based on identifier. Added additional field in transaction response for "raw" merchant name.
12/11/14	1.0.4	Terminology updates to transaction detail response examples: "R*efunded/Voided*" to "*Voided (Full Reversal)*"; "*Partial Refund / Reversals*" to "*Refund (Partial)*". Minor corrections in response payloads: "*Authorized*" to "*Approved*"; consistent timestamp examples, Additional Split Transaction examples.
4/27/15	1.1.0	Added '*dpanIdentifiers*' property to the multiple DPAN re-registration request. Added new transaction details response payload property: '*invalidated*' which indicates that the transaction is no longer valid and should be removed from the transaction history. Added additional transaction type of '*completion*' which indicates to the device that it should try to update an existing transaction, using "soft-matching" if needed. An additional example of a completion transaction has also been added. Introduced "*supportsSettlement*" registration response property to indicate if a transaction notification service can update pending transactions to a "settled" state, such as '*Approved*'.
5/27/15	1.1.1	The '*Completion*' transaction type has now been moved to its own property on the transaction detail record.

Date	Version	Change
4/1/16	1.1.2	Updated registration request samples

Terminology and Acronyms

Terms presented throughout this document are defined in the following table. Some acronyms are commonly used in publications and standards while others have been internally generated.

Term	Definition
APDU	Application Protocol Data Unit: An atomic message between two entities.
API	Application Programming Interface
Card Issuer	The entity that manages the lifecycle of a card, and its assignment to a user.
CASD	Controlling Authority Security Domain
DPAN	Device Primary Account Number: A payment element which serves as the logical link between a NFC-enabled device and a physical card (F-PAN).
FPAN	Funding Primary Account Number: The credit or debit card issued by the Issuing bank.
NFC	Near Field Communication: A proximity-based communication mechanism which enables contact-less transactions between mobile devices and payment terminals.
POS	Point of Sale: The NFC-enabled merchant payment device.
SE	Secure Element: The physical execution context on a device.
APNs	Apple Push Notification Service: Developer service for delivering notifications to a mobile device.

Overview

The card notification services provides a mechanism for a PNO or issuer to deliver information directly to a user's device about payment transactions that have occurred, along with messages for the user. This is accomplished via a generic set of RESTful web services which are leveraged to provide notifications in two main flows:

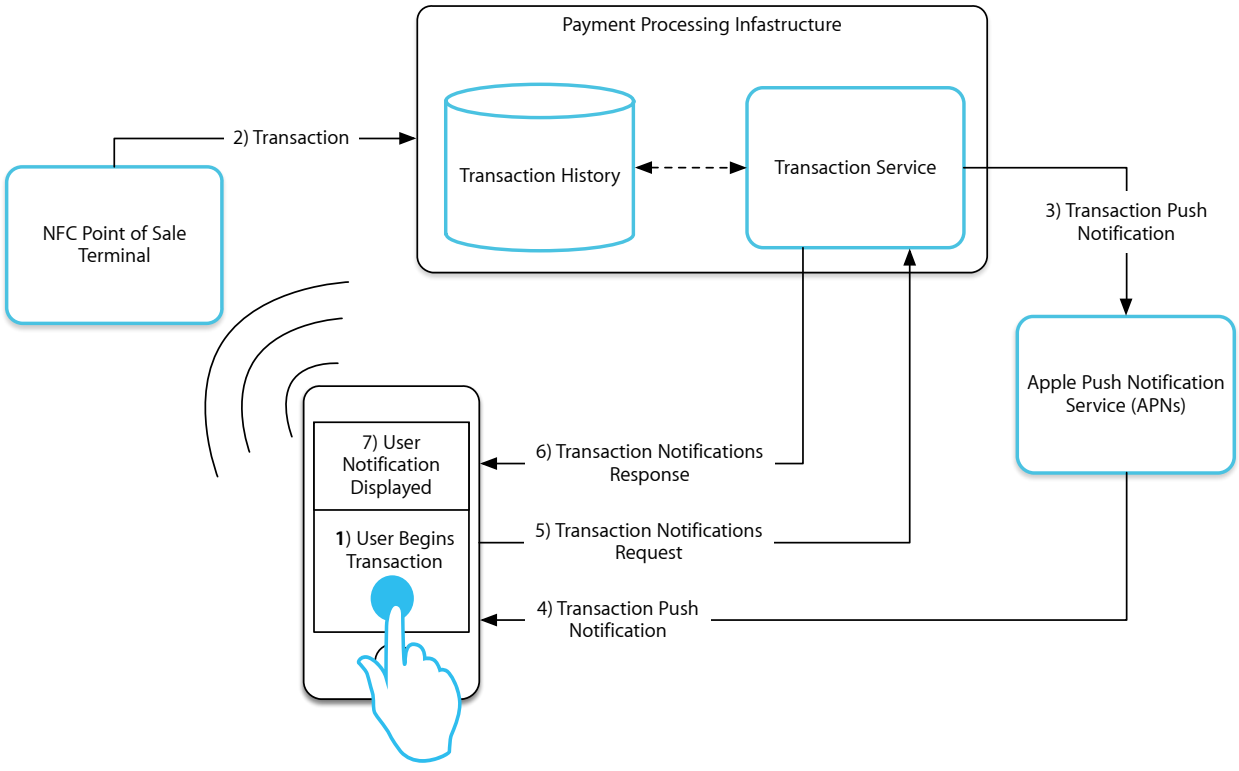
1) Alerting the device of a new notification, such as a transaction notification which occurred over the NFC interface. This stateless notification, sent over the Apple Push Notification Service, instructs the device to request the latest update(s) via the REST web service. This update will include the actual information for the notification.

2) Allowing a user to manually (triggered via a UI interaction) request the latest notifications from the provider that have occurred for a particular service.

To add some context to notification services, below are two examples of how these flows will be used to provide both alerts for transactions along with a list of previous transactions.

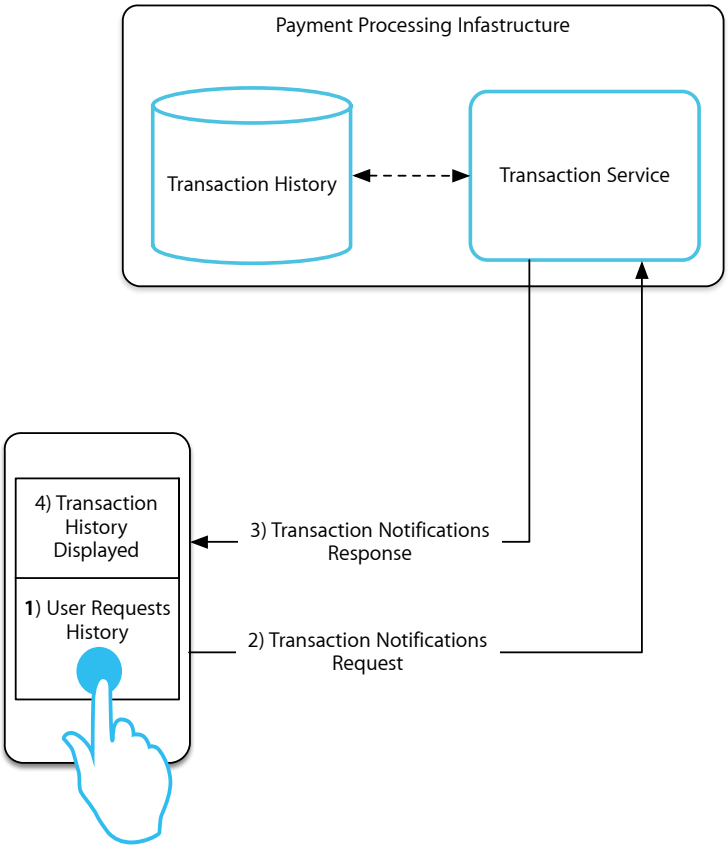
Example 1 - Transaction Notification Alert Flow

In this flow, a NFC transaction has occurred and has been routed to the party responsible for providing the transaction service. As soon as possible, a push notification should be sent to the user's device via the Apple Push Notification Service (APNs). This stateless notification acts as a prompt to the device to request any updates from the service. The service will then respond with transaction detail records that have occurred since the device last checked in. Using this approach, if a user is in an offline scenario (i.e a location internet connectivity is unavailable, such as a subway), and performs multiple purchases but "misses" notifications, it will receive all transaction updates as soon as it comes online and receives a single notification from APNs.



Example 2 - Transaction Notification History Flow

In this flow, a user is actively requesting transaction details from the service. These transactions may either be from the DPAN on the device's SE or, if supported, any transaction events for the FPAN. Because the user is actively requesting information, the device will reach out directly to the transaction service provider, which will in turn, reply with any updated transaction notification records since the last request.



Registration

Explanation

A single API is used to register, re-register, and de-register a device to receive card notifications for all service types. Issuing the POST command on the endpoint will either register or re-register a device. Issuing DELETE on the endpoint will de-register a device. For requests that contain registration data (register & re-register), the API uses the signed SEID to authenticate the requestor.

For reference, a device will perform the following steps to create the payload for these requests:

1. Create a JSON dictionary with the appropriate registration data.
2. Convert the JSON dictionary to a UTF-8 string.
3. Convert the UTF-8 string to an array of bytes.
4. Perform a SHA-256 hash over the bytes.
5. Use the INTERNAL AUTHENTICATE APDU to ask the Secure Element to sign the hash.
6. Build a wrapper JSON dictionary to transport the signature and the registration data object.

When the notification service provider receives this request, the following steps should be performed:

1. Retrieve the CASD Certificate from the JSON payload, hex-decode the contents, and parse the resulting bytes into TLV.[1]
2. Using the CA Identifier field of the CASD Certificate as a key, retrieve the appropriate trusted CASD CA Certificate from your key store.[2]
3. Concatenate all signed CASD Certificate field data[3], perform a SHA256 hash of the bytes, and verify the CASD Certificate's ECDSA Signature against the CASD CA Certificate from step 2.
4. Retrieve the signature parameter of the JSON payload and Base64-decode the contents.
5. Retrieve the registrationData parameter of the JSON payload and Base64-decode the contents, and perform a SHA256 hash of the resulting bytes.
6. Assemble the signed message by concatenating the following material:
    i. SHA256 hash from step 5.
    ii. Random Nonce from the signature that was decoded in step 4.
    iii. Tag 93 "Certificate Serial Number" value of the CASD Certificate.
    iv. Tag 53 "Discretionary Data" value of the CASD Certificate, if the tag is present.

1 The certificate is in GP-based format, and stored within the Secure Element's Controlling Authority Security Domain.

2 This is an X.509 certificate, distribution method TBD.

3 See Appendix A: CASD Certificate Required Data

7. Verify the signature from step 4 over the bytes from step 6 against the CASD Certificate from step 1.[4]

8. Validate that the DPAN Identifier is valid for the Secure Element identified as part of the signature.

Authentication Token

When a device registers with a notification service, it shall be provided with an authentication token. This token shall be valid for any notification requests for any DPANs registered to that service from the device (based on the supplied device identifier). That is to say, the the token is 'per device' AND 'per notification service'. Further, future registrations of new DPANs added to the device will include the shared authentication token from any currently registered DPANs for the same notification service. Any new authentication token supplied in a registration (or re-registration) will replace the shared authentication token stored token for that service, allowing for 'rolling' of the authentication token at registration time for all DPANs on the device.

Re-authenticating the Device

If a device issues a request to any notification endpoint (defined in subsequent sections of this document) with an expired or invalid token, the provider should reply with an HTTP status 401 (Unauthorized). Upon the receipt of a 401 status code, the device will perform a full re-register request.

If the originating request was for notifications for a specific DPAN (where the DPAN Identifier is supplied as part of the URL), the re-reregistration request will be sent to the URL for that specific DPAN Identifier: '*/devices/<deviceIdentifier>/registrations/dpan/<dpanIdentifier>*'. However, if the originating request was for notifications for all DPANs registered on the device (where no DPAN Identifier is supplied as part of the URL), the re-registration request will be sent to the following URL: *'</devices/<deviceIdentifier>/registrations/'*. This request should re-register all DPANs currently registered to the service.

With all re-registration requests, The old token will be supplied in the Authorization header, along with all standard registration data. After successful re-registration, the device will retry the original request with the new token. Authenticating the device is an expensive operation which interrupts the retrieval of transactions. It should be used only when necessary and not delay the user's experience of timely notifications. To that end, the provider should send a push to the device as soon as the token expires to initiate device re-authentication. Token expiration can also be timed such that re-authentication occurs at opportune times for the user and notification provider.

An optional 'inline' authentication token can also be supplied with all notification responses to limit the lifetime of a single token.

De-Registration

A de-registration request will be issued from a device when a user elects to stop receiving card notifications from a service. While the device will attempt a "best effort" to perform a de-register, there is no guarantee of this request. For this reason, services should not rely on this request. If a new register request (with potentially different device identifier) is received for the same DPAN Identifier, it should be honored.

4 See Appendix B: CASD Certificate Signatures

Supported Endpoints

Notification registration requests will use a single endpoint for registration, called once per service URL. The deviceIdentifier is a GUID provided by the device at registration time, created on demand at time of the first registration. Its sole purpose is device identification on a particular notification service and is not related or derived from the Secure Element Identifier. Future calls to request notifications will reference this device Identifier. For re-registration events, such as authentication token rolling a second endpoint, which does not include a specific DPAN Identifier, can be called, depending on the context of the re-registration event. For more details, see section entitled *Re-authenticating the Device*.

Single DPAN: <*serviceURL*>/devices/<*deviceIdentifier*>/registrations/dpan/<*dpanIdentifier*>
Registers, re-registers, or de-registers a specific device for a notification service for a specific DPAN.

Multiple DPANs*:* <*serviceURL*>/devices/<*deviceIdentifier*>/registrations/
Re-registers a specific device for a notification service for ALL previously registered DPANs.

Request Components

Registration Data

The registration request (POST) contains the following registration information.

Data Point	Explanation
pushToken	APNs push token, supplied by the iOS device.
accountHash	Used to identify a user by their Apple account. As this value is also provided during 'Link and Provision', is can be used during registration for validation purposes.
dpanIdentifiers	Multiple DPAN re-registration only. An array of all DPAN Identifiers associated with the request. If this value is not provided, it should be assumed that all previously registered DPAN Identifiers are being re-registered.

Input: Payload

The request payload will contain the encoded registration data, signature, and the CASD certificate. As de-register does not contain any registration data, the request will not have a payload.

Data Point	Explanation
registrationData	Base64 encoded UTF-8 bytes of registration data JSON dictionary.
signature	Base64 encoded bytes of INTERNAL AUTHENTICATE APDU response
casdCertificate	Hex encoded CASD certificate

Input: Headers

Re-registration, de-registration, and additional DPAN registration requests will include the authentication token for request authentication purposes.

Data Point	Explanation
Authorization	Supplies the authentication token from a previous registration request (if it exists). Header value should be in the form:"*AuthenticationToken <authenticationToken>*"

Output: Status Codes

The following HTTP status codes will be explicitly handled by the device. All other status codes will be treated as failures.

Status Code	Explanation
200 (OK)	The supplied registration signature was validated and registration, re-registration, or de-registration was successful.

Output: Payload

Data Point	Explanation
authenticationToken	Token used in authorization header to get transaction details for this device. Suggested length: 32-64 characters.
supportsSettlement	Transaction notification services only. The transaction service supports settlement of transactions from '*Pending*' to a settled state, such as '*Approved*'.

Sample Request & Response

Registration Data

```
{
  “pushToken”:
      ”ccedc5adde07d3e55b4a28e2298553c6348bbb7f60952dd9a6da97c35432894c“,
  “accountHash”:
      “e1948ba89f39ee884962d0b76a65b7cc494eab81a6a31d1eb68ba3340ec722d5=“,
  “dpanIdentifiers:
       [“DTN1238122132DSW32A3129012”, “DTN82245318913DSW3A4291228”]
}
```

Request Payload

```
{
  “signature”:
      "QM79\/NnW+SCJ5II3WC7RVV8oDllVlrxgowzmm\/RX5+Mea8J0YbGD9gNe4+wQ2+G
      +0E5B4XkGZSJSEOCILxF842kOiVuaJQrufcjNabZm6qQr",
  “registrationData”:
      "eyJwdXNoVG9rZW4iOiJjY2VkYzVhZGRlMDdkM2U1NWI0YTI4ZTIyOTg1NTNjNjM0OGJiYjdmNjA5
      NTJkZDlhNmRhOTdjMzU0MzI4OTRjIiwiYWNjb3VudEhhc2giOiJlMTk0OGJhODlmMzllZTg4NDk2M
      mQwYjc2YTY1YjdjYzQ5NGVhYjgxYTZhMzFkMWViNjhiYTMzNDBlYzcyMmQ1In0=",
  “casdCertificate”:
      "9310042E3C19DC2980014052004212916807420799990001900001 5F2001009501825F240421
      1401194501005308259 6AB19F1F2E8095F374030B4423E313FA0A712A4ACE0A5ED70750989B3E
      F6CA7131E701D638EA154C8EC0EDD12290DE916A724C3430CDDC83CD829C3C942A86E894270CC
      CBF62EDB96F47F49438641046B67707FAC95428FBDA05FA10FF1C190EE5FE01A172B3CC4124AF
      A3BC6A8C0544F77BB8DE913597DF2CDFC1DE70E4E9C8C86D938F101A332E4548E90B81B91A7"
}
```

Response Payload

```
{
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d56”,
  “supportsSettlement” : TRUE
}
```

Push Notifications

Explanation

The Apple Push Notification Service will be used to alert a user's device that new notifications are available and should be fetched. A specific 'push topic' will be used to alert the device that either card transactions or messages are available, and should be requested. More information on these push topics can be found in the Passbook Card Customization document.

This secure service relies on certificates and device 'push tokens' to ensure delivery of notifications to the intended device. As APNs has been available for some time as a tool for registered iOS Developers to deliver short notifications to their users' devices, comprehensive documentation can be found at the Apple Developer Portal. For a detailed overview of the service, please see the 'Apple Push Notification Section' of the Local and Push Notification Programming Guide.

Registration and Setup

To develop and deploy the provider side of the APNs, SSL certificates must be obtained from the iOS Developer Center for each service (transactions or messages) providing push notifications. Each certificate is identified by its bundle ID. Certificates are also limited to one of two development environments (Development and Production). For the purposes of Notifications, only the production APNs environment will be used.

It is required that every party responsible for delivering notifications registers as an iOS Developer to obtain their certificates. After an account is created, the following steps should be taken to generate the SSL certificate.

1. Click App IDs in the sidebar on the left side of the window. The next page displays your valid application IDs. An application ID consists of an application's bundle ID prefixed with a ten-character code generated by Apple. Create a new application with an appropriate bundle ID. For the purpose of notifications, this bundle ID as the 'push-topic' which iOS will be listening for.

2. Locate the application ID for the development SSL certificate (and that is associated with the Development provisioning profile) and click Configure. You must see "Available" under the Apple Push Notification Service column to configure a certificate for this application ID.

3. In the Configure App ID page, check the Enable Push Notification Services box and click the Configure button. Clicking this button launches an APNs Assistant, which guides you through the next series of steps.

When you finish the APNs Assistant, you are returned to the Configure App ID page of the iOS Dev Center portal. The certificate should be badged with a green circle and the label "Enabled". Further instructions on installation of the SSL certificate can be found in the Local and Push Notification Programming Guide.

The push topic from the Push Notification certificate will need to be transmitted to Apple. This will occur during the Link and Provision flow. See the API specification for details.

Push Notification Transmission

As a provider, you communicate to the intended device through the Apple Push Notification service over a binary interface using the Push Token (delivered during registration). This interface is a high-speed, high-capacity interface for providers; it uses a streaming TCP socket design in conjunction with binary content. The binary interface is asynchronous.

As a provider, you are responsible for the following aspects of push notifications:

- You must compose the notification payload (see 'The Notification Payload' in the Local and Push Notification Programming Guide). However, notifications will be used only to alert the device that updated transaction information is available, so the payload should be empty.

- Connect regularly with the feedback service and fetch the current list of those devices that have repeatedly reported failed-delivery attempts. Then stop sending notifications to the devices associated with those applications.

The binary interface of the production environment is available at "*gateway.apple.com*", port 2195.

Full documentation on the binary interface can be found in the 'Binary Interface and Notification Format' section of Local and Push Notification Programming Guide.

Transactions

Explanation

This API is used to get transaction details for one or more DPAN Identifiers. Transaction details provide the information about the latest state of transactions associated with an account, uniqued on the "identifier" field. Calls to this API either return transaction details for all DPAN Identifiers for which the device has called REGISTER or transaction details for a specific DPAN identifier, if provided in the URL query string. The API will be called directly from the iOS device. It is implemented using an HTTP GET and leverages a "tag" query parameter to determine if any updates to the transaction data exist since the previous request. Further, the Authorization header will be supplied containing the authenticationToken provided to the device during the Registration process.

GET TRANSACTIONS may be called:

A. As a result of the provider sending a Push Notification

B. Manually from the device when the UI needs to be refreshed

Before this call can be made, a REGISTER call must be made. The register API allows the provider to match the secure element, DPAN identifier and ultimately record the Push Notification token.

Transaction Identifier

The goal of the transaction identifier is to define an opaque and unique method of identifying a transaction which can be generated by parties such as the payment device, PNO, and issuer (if applicable) for uniquing and reference purposes. The transaction identifier is defined as follows:

MagStripe Transactions

```
16-LSB (SHA-256 (Track 1 Data)) | 16-LSB (SHA-256 (Track 2 Data))
```

- '|' denotes concatenation
- 16-LSB denotes 16 least significant bytes
- Track 1 Data should be encoded in ASCII without the start sentinel, end sentinel, and longitudinal redundancy check (LRC). All other special characters should be included.
- Track 2 Data should be encoded in 4-bit packed BCD (binary-coded decimal). Odd-length data is padded with a hex 'F' at the end.
- Should Track 1 or Track 2 Data be unavailable it should be replaced with zeros. To account for this, the device will partially match on the available half of the transaction ID within a time window.
- Both Track 1 and Track 2 Data should correspond to Track 1 and Track 2 data 'as retrieved' from the payment device (i.e, padding characters such as spaces should be retained).

Sample MagStripe Transaction ID

1. Get Track 1 Data (ASCII encoded):

```
%B1234987623458765^APPLESEED/JOHN  ^1509123000000000?
```

2. Remove sentinels and LRC, if required:

```
B1234987623458765^APPLESEED/JOHN  ^1509123000000000
```

3. SHA-256 hash the data:

```
1d21430fc1f67370aa0780139d554128847294657b04c9d1d101abbc78f2d7a9
```

4. Get Track 2 Data:

```
1234987623458765d15091230000000000000
```

5. Pad with an Hex 'F' if odd:

```
1234987623458765d150912300000000000f
```

6. BCD encode Track 2:

```
0001 0010 0011 0100 1001[…] 1111
1———2———3———4———9———[…]—F———
```

7. SHA-256 hash the BCD encoded data:

```
b75c0ff874d16e55727b0a498b3306cea334daeee4f7a991c5def7510ed4fb04
```

8. Take the 16 least significant bytes of step #3 and concatenate with the 16 least significant bytes of step #7.

```
1d21430fc1f67370aa0780139d554128847294657b04c9d1d101abbc78f2d7a9
b75c0ff874d16e55727b0a498b3306cea334daeee4f7a991c5def7510ed4fb04
```

9. This will result in the Transaction Identifier:

```
847294657b04c9d1d101abbc78f2d7a9a334daeee4f7a991c5def7510ed4fb04
```

EMV & 3D Secure Transactions

For EMV Transactions

```
SHA-256 (DPAN | ATC | Application Cryptogram)
```

For 3D Secure Transactions

```
SHA-256 (DPAN | Remote Payment Cryptogram)
```

- '|' denotes concatenation.
- DPAN used is as provisioned in tag '*5A*'.
- ATC is the counter value provided to the terminal in tag '*9F36*'.
- Application Cryptogram is value provided to the terminal in tag '*9F26*'.
- Remote Payment Cryptogram is the binary value.
- Note: The value will be all '*00*'s in the event the status word is different than '*9000*'.

A stated goal of this design is to avoid exposing the DPAN outside of the SE where possible.

Supported Endpoints

<*serviceURL*>/devices/<deviceIdentifier>/transactions
Returns transactions for all D/FPANs a device has registered for.

<*serviceURL*>/devices/<deviceIdentifier>/dpan/<*dpanIdentifier*>/transactions
Returns D/FPAN transactions for a particular DPAN identifier.

Request Components

Transaction Details

Key Name	Type	Required	Note	Sample
identifier	String	Yes	Unique identifier for a transaction from the provider. Suggested length: 32-64 characters (GUID).	3923434801332489 03239032
transactionIdentifer	String	Yes*	The transaction identifier for a transaction, generated using the algorithms supplied in this specification.	847294657b04c9d 1d101abbc78f2d7a 9a334daeee4f7a99 1c5def7510ed4fb04
transactionType	String	Yes	This defines the type of the transaction. Current values are '*Purchase*' and '*Refund*'.	Purchase
dpanIdentifier	String	No**	Device PAN Identifier	DA989809879866
fpanIdentifier	String	No**	Funding PAN Identifier	F12339348293823
transactionDate	String	Yes	The date of the transaction event. See Appendix C: Acceptable Date & Time Format.	2014-11-05T08:15:3 0-05:00

Key Name	Type	Required	Note	Sample
currencyCode	String	No	The currency of the transaction will be defined using ISO 4217 currency codes. If a code is not supplied, the device's current locale currency code will be used.	US
rawMerchantName	String	Yes	This field should contain the original L1 value of the merchant, as seen on a customer's bank statement.	PEETSCOFFEE4189
merchantName	String	Yes	Name of the merchant with any server-side cleanup or a doing-business-as name. If no processing is applied, this value should match the rawMerchantName.	Peet's Coffee
amount	Number	No	Transaction amount. Sign of transaction should match transaction type (i.e. + for purchase, - for refund).	10.00
transactionStatus	String	Yes	This will be used to track the authorization status of a transaction. Currently, the valid statuses are: '*Pending*', '*Approved*', '*Refunded*', and '*Declined*'. Note: A transaction status of '*Refunded*' should be used to indicate a voided transaction.	Pending
industryCategory	String	No	This will be used to describe what type of category the merchant falls into.	Grocery, Fuel, Dining
industryCode	Number	No	The industry category should be described using the ISO 18245 code, if possible.	11
invalidated	Boolean	No	If a transaction is no longer valid (i.e. it is canceled or revoked), supplying this property will remove it from the device's history.	TRUE
completion	Boolean	No	New transactions (with a different identifier) that should completely replace an existing *pending* transaction can supply this property to indicate the device should use "soft-matching" to perform a best-effort replacement.	TRUE

* Required for all DPAN transactions but not applicable and should not be returned for FPAN transactions.

** One and only one of either the dpanIdentifier or the fpanIdentifier must be returned for the transaction.

Input: Headers

Data Point	Explanation
Authorization	Supplies the authentication token from a previous registration request (if it exists). Header value should be in the form: "AuthenticationToken <authenticationToken>".

Input: Query Parameters

Data Point	Explanation
tag	The lastUpdatedTag supplied in the last transaction notifications request. This tag can be used for filtering of transactions to only the subset that has not been delivered to the device. The value of the lastUpdatedTag is at the providers description, but must be a valid as a component of a URL and not contain any special or reserved characters.

Output: Status Codes

The following HTTP status codes will be explicitly handled by the device. All other status codes will be treated as failures.

Status Code	Explanation
200 (OK)	The request was successful and new transaction details are available, given the supplied tag.
304 (Not Modified)	The request was successful but there are no new transaction details available.
401 (Unauthorized)	The authentication token is invalid and a re-registration must take place.

Output: Payload

Data Point	Explanation
lastUpdatedTag	A tag to be included in the next transaction notifications request. This tag can be used for filtering of transactions to only the subset that has not been delivered to the device. It is at the provider's discretion as to the value of the tag. If a date value is desired for a lastUpdatedTag, such as the date a record is inserted or updated in a database, it is recommended that the value should be formatted as a a string representation of UNIX time, or the number of sections since epoch, e.g. "1409097080". This will ensure correct formatting of the value when added as a component of a request URL.
authenticationToken	An optional inline 'roll' of the authentication token may be supplied, in lieu of a full re-registration.
appLaunchToken	An optional token that will be included in the "deep link" URL passed to the issuer application from iOS when a transaction item is selected.* Suggested length: 32-64 characters.
transactionDetails	Array of TransactionDetails for the DPAN Identifier the using a supplied lastUpdatedTag for filtering.

*Deep linking will only be enabled if the transaction URL scheme is included in the card customization.

Sample Request & Response

Below are transaction detail responses for basic pending and approved transactions. Additional examples are available in Appendix D: Supplementary Transaction Detail Examples.

Request Headers

```
Authorization: AuthenticationToken 3adadff4dc01e47b6b69462eb4219d56
```

Response Payload

```
{
  “transactionDetails”:[
    {
      “identifier”: “72e5640d5ad04f57b6240d0dba6a5dac“,
      “transactionIdentifier”:
          “847294657b04c9d1d101abbc78f2d7a9a334daeee4f7a991c5def7510ed4fb04“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-1-13T19:05:30-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Peet’s Coffee”,
      “rawMerchantName” : “PEETSCOFFEE 4532”
      “amount”: 10.55,
      “transactionStatus”: “Approved”,
      “industryCategory”:”Fast Food Restaurants”,
      “industryCode”:2
    },
    {
      “identifier”: “3a338a67c1994649a1c521a002cec08d“,
      “transactionType”: “Purchase”,
      “fpanIdentifier": ”nf3bcv434nf4t4une5ys3dkdfjq32yj56msbre2==“,
      “transactionDate”: “2014-1-13:T19:05-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Target”,
      “rawMerchantName” “TARGET”,
      “amount”: 69.34,
      “transactionStatus”: “Pending”,
      “industryCategory”: ”Grocery”,
      “industryCode”: 10
    }
  ],
  “lastUpdatedTag” : ”1411509640”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”,
  “appLaunchToken” : “2t3322ygws43b5m57wrwefwgvbew42143t3454534534ew”
}
```

Messages

Explanation

This API is used to get message notifications for one or more DPAN Identifiers. A message can include account information updates or offers for the user. This call either returns messages for all DPAN Identifiers for which the device has called REGISTER or messages for a specific DPAN Identifier. The API will be called directly from the iOS device. It is implemented using an HTTP GET and leverages a "tag" query parameter to determine if any updates to the message notification data exist since the previous request. Further, the Authorization header will be supplied containing the authenticationToken provided to the device during the Registration process.

GET MESSAGES may be called:

A. As a result of the provider sending a Push Notification

B. Manually from the device when the UI needs to be refreshed

Before this call can be made, a REGISTER call must be made. The register API allows the provider to match the secure element, DPAN Identifier and ultimately record the Push Notification token.

Supported Endpoints

<*serviceURL*>/devices/<deviceIdentifier>/messages
Returns messages for all DPANs a device has registered for.

<*serviceURL*>/devices/<deviceIdentifier>/dpan/<*dpanIdentifier*>/messages
Returns messages for a particular DPAN identifier.

Request Components

Message Details

Key Name	Type	Required	Note	Sample
identifier	String	Yes	Unique identifier for a message from the provider. Suggested length: 32-64 characters.	3923434801332489032390
dpanIdentifier	String	Yes	Device PAN Identifier	989809879866
messageDate	String	Yes	The date when the message was generated. See Appendix C: Acceptable Date & Time Format.	2014-11-05T08:15:30-05:00
expirationDate	String	Yes	The date when the message is no longer valid and should be removed. See Appendix C: Acceptable Date & Time Format.	2014-11-15T08:15:30-05:00
content	String	Yes	A plain-text message to be displayed to the user. Messages longer than 128 characters may be truncated.	Your credit limit has been updated to $30,000.00.
allowDeepLink	Boolean	Yes	A flag indicating if the message should "deep link" to the provider's/issuer's iOS App.	TRUE

Input: Headers

Data Point	Explanation
Authorization	Supplies the authentication token from a previous registration request (if it exists). Header value should be in the form: "AuthenticationToken <authenticationToken>".
Accept-Language	Supplies the language code the device is currently requests. The value for this language code is an IETF Language Tag, such as "en-us". All messages MUST be generated using this language code.

Input: Query Parameters

Data Point	Explanation
tag	The lastUpdatedTag supplied in the last message notifications request. This tag can be used for filtering of messages to only the subset that has not been delivered to the device. The value of the lastUpdatedTag is at the providers description, but must be a valid as a component of a URL and not contain any special or reserved characters.

Output: Status Codes

The following HTTP status codes will be explicitly handled by the device. All other status codes will be treated as failures.

Status Code	Explanation
200 (OK)	The request was successful and new message details are available, given the supplied tag.
304 (Not Modified)	The request was successful but there are no new message details available, given the supplied tag.
401 (Unauthorized)	The authentication token is invalid and a re-registration must take place.

Output: Payload

Data Point	Explanation
lastUpdatedTag	A tag to be included in the next message notifications request. This tag can be used for filtering of messages to only the subset that has not been delivered to the device. It is at the provider's discretion as to the value of the tag. If a date value is desired for a lastUpdatedTag, such as the date a record is inserted or updated in a database, it is recommended that the value should be formatted as a a string representation of UNIX time, or the number of sections since epoch, e.g. "1409097080". This will ensure correct formatting of the value when added as a component of a request URL.
authenticationToken	An optional inline 'roll' of the authentication token may be supplied, in lieu of a full re-registration.
appLaunchToken	An optional token that will be included in the "deep link" URL passed to the issuer application from iOS when a message item is selected.* Suggested length: 32-64 characters.
messageDetails	Array of MessageDetails for the supplied DPAN Identifier the using a supplied lastUpdatedTag for filtering.

*Deep linking will only be enabled if the message URL scheme is included in the card customization.

Sample Request & Response

Request Headers

```
Authorization: AuthenticationToken 3adadff4dc01e47b6b69462eb4219d56
Accept-Language: en-US
```

Response Payload

```
{
  “messageDetails”:[
    {
       “identifier”: “323932032093209“,
       “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
       “messageDate: “2014-1-13T19:05:30-05:00”,
       “expirationDate: “2014-1-23T19:05:30-05:00”,
       “content": ”Your credit limit has been increased to $30,000.00“,
       “allowDeepLink” : false
    }
  ],
  “lastUpdatedTag” : ”14115024562”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
  “appLaunchToken” : “2t3322ygws43b5m57wrwefwgvbew42143t3454534534ew”
}
```

Notification Deep Linking

Explanation

To allow a user to obtain additional details about a notification or a more complete look at all of their, notifications, such as their transaction history, iOS supports optional deep linking of a notification directly into an issuer's App. This is accomplished by iOS generating a URL for the notification from data in the response and then passing it to the registered App, if it exists on the device. The App is then launched with that URL, where the App is then responsible for taking the user to a relevant view of either the individual notification, if supplied, or all notifications of a specific service type.

URL Format

The responsible App, if opting into transaction deep linking, will need to support a URL scheme which can process URLs in the following formats, based on what type of transaction details are supported:

<*serviceScheme*>://dpan/<*dpanIdentifier*>/<serviceType>/<*identifier*>?token=<*appLaunchToken*>
Should route a user to information on a DPAN based notification record for the specified service type.

<*serviceScheme*>://fpan/<*fpanIdentifier*>/<serviceType>/<*identifier*>?token=<*appLaunchToken*>
Should route a user to information on a FPAN based notification record for the specified service type.

<*serviceScheme*>://dpan/<*dpanIdentifier*>/<serviceType>?token=<*appLaunchToken*>
Should route a user to a general summary of notification records for the specified service type.

The above URL formats include an optional '*token*' query parameter. The value for this parameter is populated from the latest '*appLaunchToken*' value returned in a particular notification service response. The intent of this parameter is to provide a mechanism for the provider's system to communicate to the iOS App for purposes of authentication or state. This value is opaque to iOS.

URL Parameters

Data Point	Explanation
serviceScheme	The URL scheme registered by the issuer's iOS application for the particular service type (transactions, messages). An example would be "my-bank-app"://. This same scheme can be used for multiple services, if desired.
dpanIdentifier	Device PAN Identifier from the associated notification.
fpanIdentifier	Funding PAN Identifier from the associated notification.
identifier	The '*identifier*' from the associated notification.
serviceType	The type of notification service. Possible values are '*transactions*' and '*messages*'.
appLaunchToken	The latest '*appLaunchToken*' returned in the specific notification device response.

Scheme Registration

In order to receive deep link URLs, the responsible App must register the URL scheme, which supports the above format, with the system. Further, the scheme name AND the App's bundle identifier must be included in the card's pass information for transaction deep linking to be enabled on a user's device. For more information, consult the Implementing Custom URL Schemes documentation and the Passbook Card Customization document.

Appendix A: CASD Certificate Required Data

Explanation

CASD Certificate ECDSA Signature is calculated off card over the following CASD Certificate fields, inclusive of all tag, length, and data bytes, in the order given here:

Tag	Length	Description	Presence
7F49	67	Public Key	Mandatory
86	65	ECC Public Key (Format: 04 | X | Y)	Mandatory
93	1-16	Certificate Serial Number	Mandatory
42	1-16	CA Identifier	Mandatory
5F20	1-16	Subject Identifier	Mandatory
5F25	4	Effective Date (YYYYMMDD, BCD Format)	Optional
5F24	4	Expiration Date (YYYYMMDD, BCD Format)	Mandatory
95	1	Key Usage '82'	Mandatory
45	1-16	CA Security Domain Image Number	Mandatory
53	8	Discretionary Data	Optional

Ref: Mobile Payment Secure Element Specification

Appendix B: CASD Certificate Signatures

Explanation

The Secure Element outputs ECDSA signatures as a Length-Value data structure (except the nonce which has only value and no length tag) with the following definition:

Length	Data Element	Presence
1	Length of the ECDSA Signature	Mandatory
64	Signature Data	Mandatory
16	Random Nonce	Mandatory

Appendix C: Acceptable Date & Time Format

Notification services should provide date and time values using the W3C's definition of the ISO 8601 date and time format. Specifically, the value must be a complete date with hours and minutes, and may optionally include seconds. Examples of acceptable date and time values follow:

Format String	Example
YYYY-MM-DDThh:mmTZD	2014-08-16T19:20+01:00
YYYY-MM-DDThh:mm:ssTZD	2014-08-16T19:20:30+01:00

Appendix D: Supplementary Transaction Detail Responses

Example 1 - Voided (Full Reversal) Transactions

For a voided transaction, the original transaction record, indicated by the '*identifier*' field, should be amended with a transaction status of '*Refunded*'. The amount value should remain as the amount from the original transaction.

Orignal Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “9e540e54bb924352834ee2842a00f181“,
      “transactionIdentifier”: “63234cfb96f94528…919b7e19f8338fc9“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-23T19:05:30-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Target”,
      “amount”: 75.00,
      “transactionStatus”: “Pending”,
      “industryCategory”: ”Grocery”,
      “industryCode”: 5411
    }
  ],
  “lastUpdatedTag” : ”1411509640”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Update to Voided (Reversed) State

```
{
  “transactionDetails”:[
    {
      “identifier”: “9e540e54bb924352834ee2842a00f181“,
      “transactionIdentifier”: “63234cfb96f94528…919b7e19f8338fc9“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-23T19:05:30-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Target”,
      “amount”: 75.00,
      “transactionStatus”: “Refunded”,
      “industryCategory”: ”Grocery”,
      “industryCode”: 5411
    }
  ],
  “lastUpdatedTag” : ”1411596321”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Example 2 - Refund (Partial) Transactions

For refunds that do not modify an existing transaction, a new record should be returned with a new identifier. This new transaction should have a transaction type of '*Refund*' . If there is no associated transaction identifier (presumably the refund transaction did not happen over contactless), the value may be omitted. Note the transactionDate of the refund transaction is the date of the refund event, not the original transaction.

Orignal Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “9e540e54bb924352834ee2842a00f181“,
      “transactionIdentifier”: “5e2bf7e66c1643c…ba792bd05be330722“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-23T19:05:30-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Banana Republic”,
      “amount”: 130.32,
      “transactionStatus”: “Pending”,
      “industryCategory”:”Clothing/Apparel”,
      “industryCode”: 5691
    }
  ],
  “lastUpdatedTag” : ”1411502345”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Partial Refund Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “f0b4c8d7a8614e829bcb7655dda486d9“,
      “transactionType” : “Refund”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-24T19:08:30-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Banana Republic”,
      “amount”: -50.25,
      “transactionStatus”: “Approved”,
      “industryCategory”: ”Clothing/Apparel”,
      “industryCode”: 5691
    }
  ],
  “lastUpdatedTag” : ”141151164”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Example 3 - Split Transactions

Split Transactions should be delivered as unique transaction detail records with unique identifiers, even if they share the same transaction identifier. The original, pending transaction record should be updated with the first split's details after it is authorized, reusing the original record's identifier.

Original Pending/Authorization Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “9e540e54bb924352834ee2842a00f181“,
      “transactionIdentifier”: “79e2cf5aaf494d18…a7fcba3e9e83d07f“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-23T19:05:30-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Apple Store”,
      “amount”: 100.55,
      “transactionStatus”: “Pending”,
      “industryCategory”: ”Electronics”,
      “industryCode”: 5732
    }
  ],
  “lastUpdatedTag” : ”1411509642”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

First Split Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “9e540e54bb924352834ee2842a00f181“,
      “transactionIdentifier”: “79e2cf5aaf494d18…a7fcba3e9e83d07f“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-24T13:03:00-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Apple Store”,
      “amount”: 22.20,
      “transactionStatus”: “Approved”,
      “industryCategory”: ”Electronics”,
      “industryCode”: 5732
    }
  ],
  “lastUpdatedTag” : ”1411510360”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”,
}
```

Second Split Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “3e136d888fc94063b9621d668667de50“,
      “transactionIdentifier”: “79e2cf5aaf494d18…a7fcba3e9e83d07f“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-25T07:21:40-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Apple Store”,
      “amount”: 50.00,
      “transactionStatus”: “Approved”,
      “industryCategory”: ”Electronics”,
      “industryCode”: 5732
    }
  ],
  “lastUpdatedTag” : ”1411511521”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Third Split Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “c1cca59abfb44db980fad8ffc75768fe“,
      “transactionIdentifier”: “79e2cf5aaf494d18…a7fcba3e9e83d07f“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-26T07:22:10-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Apple Store”,
      “amount”: 28.35,
      “transactionStatus”: “Approved”,
      “industryCategory”: ”Electronics”,
      “industryCode”: 5732
    }
  ],
  “lastUpdatedTag” : ”1411516242”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Example 4 - Authorization Hold

Fuel or other transactions that have a authorization amount with a predefined value (e.g. $120.00, $150.00, etc.) should not include this amount in transaction details. Only when a final value is reported (or the transaction clears) should the record be updated with the amount.

Original Pending/Authorization Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “d903bffd4bb64c889a3819cf43be1c16“,
      “transactionIdentifier”: “af519455b4304785b…f34c163c92bc96c“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-23T19:05:30-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Esso Fuel”,
      “transactionStatus”: “Pending”,
      “industryCategory”: ”Fuel”,
      “industryCode”: 5542
    }
  ],
  “lastUpdatedTag” : ”1411501234”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Update to Pending with Amount

```
{
  “transactionDetails”:[
    {
      “identifier”: “d903bffd4bb64c889a3819cf43be1c16“,
      “transactionIdentifier”: “af519455b4304785b…f34c163c92bc96c“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-23T20:03:01-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Esso Fuel”,
      “amount”: 76.30,
      “transactionStatus”: “Pending”,
      “industryCategory”: ”Fuel”,
      “industryCode”: 5542
    }
  ],
  “lastUpdatedTag” : ”1411502345”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Update to Approved State

```
{
  “transactionDetails”:[
    {
      “identifier”: “d903bffd4bb64c889a3819cf43be1c16“,
      “transactionIdentifier”: “af519455b4304785b…f34c163c92bc96c“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-24T01:06:02-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Esso Fuel”,
      “amount”: 76.30,
      “transactionStatus”: “Approved”,
      “industryCategory”: ”Fuel”,
      “industryCode”: 5542
    }
  ],
  “lastUpdatedTag” : ”1411504456”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Example 5 - Authorization Hold with Completion

Fuel or other transactions that have a completion transaction that may not share the same identifier value as the original transaction record should use the completion transaction type to indicate to the device that it should "soft-match" to an existing record, if possible.

Original Pending/Authorization Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “d903bffd4bb64c889a3819cf43be1c16“,
      “transactionIdentifier”: “af519455b4304785b…f34c163c92bc96c“,
      “transactionType” : “Purchase”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-23T19:05:30-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Esso Fuel”,
      “transactionStatus”: “Pending”,
      “industryCategory”: ”Fuel”,
      “industryCode”: 5542
    }
  ],
  “lastUpdatedTag” : ”1411501234”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```

Completion Transaction

```
{
  “transactionDetails”:[
    {
      “identifier”: “a8fd1fbb819c3313181923bf19018“,
      “transactionType” : “Completion”,
      “dpanIdentifier": ”se23fcd342332fdgawer545Q4tk23dvc42d3fj924f=“,
      “transactionDate”: “2014-9-24T01:06:02-05:00”,
      “currencyCode”: “USD”,
      “merchantName”: “Esso Fuel”,
      “amount”: 76.30,
      “transactionStatus”: “Approved”,
      “industryCategory”: ”Fuel”,
      “industryCode”: 5542
    }
  ],
  “lastUpdatedTag” : ”1411509990”,
  “authenticationToken” : “3adadff4dc01e47b6b69462eb4219d57”
}
```