HTNG
PAYMENT
TOKENIZATION
SPECIFICATION

21 February 2018

About HTNG

Hospitality Technology Next Generation (HTNG) is a non-profit association with a mission to foster, through collaboration and partnership, the development of next-generation systems and solutions that will enable hospitality professionals and their technology vendors to do business globally in the 21st century. HTNG is recognized as the leading voice of the global hospitality community, articulating the technology requirements of hospitality companies of all sizes to the vendor community. HTNG facilitates the development of technology models for hospitality that will foster innovation, improve the guest experience, increase the effectiveness and efficiency of hospitality venues, and create a healthy ecosystem of technology suppliers.

Manufacturers and software providers shall not claim compliance with portions of the requirements of any HTNG specification or standard, and shall not use the HTNG name or the name of the specification or standard in any statements about their respective product(s) unless the product(s) is (are) certified as compliant to the specification or standard.

Permission is granted for implementers to use the names, labels, etc. contained within the specification. The intent of publication of the specification is to encourage implementations of the specification.

This specification has not been verified for avoidance of possible third-party proprietary rights. In implementing this specification, usual procedures to ensure the respect of possible third-party intellectual property rights should be followed. Visit http://htng.org/ip-claims to view third-party claims that have been disclosed to HTNG. HTNG offers no opinion as to whether claims listed on this site may apply to portions of this specification.

The names Hospitality Technology Next Generation and HTNG, and logos depicting these names, are trademarks of Hospitality Technology Next Generation. Permission is granted for implementers to use the aforementioned names in technical documentation for the purpose of acknowledging the copyright and including the notice required above. All other use of the aforementioned names and logos requires the permission of Hospitality Technology Next Generation, either in written form or as explicitly permitted for the organization's members through the current terms and conditions of membership.

Table of Contents

1 This Specification at a Glance

This specification defines the messages used to convert sensitive cardholder information into tokens, which can be safely stored in systems that make up a hotelier’s environment. This process of converting payment information into a token is called "tokenization."

The purpose of tokenization is to remove valuable payment card information from the business systems of the hotel. Tokenization replaces the Personal Account Number (PAN) and other possible card data with tokens. Tokens can be safely transmitted in messages and stored in databases. The actual PAN data is stored separately in a secure data vault. This vault is a separate, isolated system possibly hosted by a third party or in the cloud.

Tokenization reduces the number of hotel systems handling sensitive payment card information. The expected result is a reduction of the costs maintaining compliance with Payment Card Industry (PCI) rules. This cost reduction will occur by limiting the number of systems vulnerable to potential loss of payment information and reducing the systems in scope for PCI auditing.

Although tokens can represent a dynamic number of sensitive data elements, for the purposes of this specification, the token replaces the cardholder’s primary account number or PAN.

This specification provides guidance, but does not mandate how tokens should be created or formatted. This specification does not require any particular characteristics for a token, except that tokens should have no mathematical relation to the original PAN. This prevents reverse engineering using brute force techniques.

Tokens often are used in payment transactions, but only in the context of the closed-loop environment of a hoteliers’ payment infrastructure. Tokens should never have any value outside of a hoteliers’ secure system. For example, a token provided during reservation should be able to be used for a "no show" authorization and payment. However, that same token cannot ever be valid in any other retailer's environment.

Although a token by itself cannot be used to process a payment and does not need any special protection when stored or transmitted, it is still a best practice to secure a token as you would secure a customer’s other private information.

When the original payment information needs to be recovered, the token and additional authentication and authorization credentials are presented and must be accepted before the actual payment card information may be accessed. Any system with access to the original PAN data will be in PCI scope.

This specification is one document in the collection of documents that defines the HTNG Secure Payments Specifications. This specification replaces the previously issued "Data Proxy" Specification. This document is one specification in the collection of HTNG Open Specifications supporting a Secure Payments Framework within the hotel industry. This specification was developed by the Payments Workgroup to define the messages and interfaces to support tokenization.

Tokens replace payment information, such as the Personal Account Number (PAN). Unlike the PAN, the token has no value if the data is compromised or if the token is captured during transmission.

Tokens can be safely stored in these hotel systems while the actual payment information is kept separately in a secure data vault. Tokens can be safely stored and transferred between various systems used within the Industry. The services defined in this specification are designed to work in concert with services defined in the other Payment Specifications as a part of a Secure Payments Framework.

2 Document Information

2.1 Scope

This document describes the messages and interfaces required to obtain and manage payment tokens. This specification also addresses the ability to recover PAN and other payment information in a safe and secure manner consistent with the HTNG Payment Framework.

This specification was designed to meet the needs of the hotel industry. While it may be found useful in other environments, it is not intended to be compatible with other payment token specifications, such as EMVCo.

This document includes, directly or by reference, all information required to implement the interfaces described within. The document does not include information needed to implement specifications developed by other parties.

2.2 Relationship to Other Standards

This specification supersedes the HTNG Data Proxy Specification 2010A version 1.1.

2.3 Audience

The primary intended audience of this document is a developer or system designer seeking to implement the interface specifications within their products. As this document also provides business use cases, the secondary audience is generally business readers wishing to familiarize themselves with the interactions between Point of Sale (POS) and Gateways, especially to understand data security concerns.

3 Business Overview

This specification is one document in the collection of documents that defines the HTNG Secure Payments Specifications.

Securing cardholder payment data with tokenization can reduce the breach recovery costs, protect brand reputation and reduce customer frustration. Tokenization is recognized by the PCI Council as a means to help reduce scope and increase overall payment security.

Tokenization in the hotel environment can be extremely challenging due to the many systems that make up a hotel's ecosystem. PAN data can be introduced into the hotel environment from various sources including:

- Reservations
- Hotel front desk processes
- Guest profile and loyalty systems
- Accounting and reconciliation activities
- On-premise retail and dining outlets
- Other outlets within a hotel (golf, spa, catering, etc.)

This Tokenization Specification defines the messages used to convert sensitive cardholder information into a token that can be safely stored in hotel systems. The actual payment information is placed into a secure data vault. Instead of sensitive payment information, the safe token is what will be stored in hotel systems, such as a Property Management System (PMS) or a Point of Sale (POS) System.

Implementers should also consider supporting point-to-point encryption (P2PE) as a complimentary technology to tokenization. P2PE prevents the theft of all card-present PANs (whether from an inserted chip, a swiped card or a manually entered PAN) that are transmitted over networks on their way to be tokenized. The combination of tokenization and point-to-point encryption is seen as the most beneficial in reducing the greatest amount of PCI scope.

3.1 Roles

In HTNG's web services implementation, the different "end points" of each interface are defined by roles. Each product implements at least one role, and in a complex specification that defines many roles, a single system may implement multiple roles.

The use of roles allows any existing system, such as a PMS, CRS or Profile Management System, to implement any functionality deemed appropriate, while still conforming to this specification. Each system only needs to evaluate which roles its functionality plays in relevant transactions, and only needs to implement the required interface transactions defined for those roles. For example, if the specification does not set forth specific requirements for a PMS, but rather for the payment data roles performed by that system. A PMS that performs more business functions than another may, as a consequence, need to handle more payment data roles.

Any system may play multiple roles, and if it does, it must implement the required functionality of each role with respect to external messaging interfaces. The system then may handle internal transactions among the roles in any manner of its choosing.

It is important to note that in this context, certain roles will be within PCI scope, but the goal of this specification is to reduce the number of systems in roles that are in PCI scope.

This specification defines the following roles:

- Business Logic System (BLS)
- Payment Data Collection System (PDCS)
- Payment Card Information Source (PCIS)

- Payment Data Provider (PDP)
- Tokenization Service
- Token Vault
- Key Service
- Payment Information Proxy (PIP)

3.2 Business Logic System (BLS)

A Business Logic System is used by the merchant to manage key elements of its business, such as reservations, check-in, profile maintenance, etc. The system requires the handling and/or transmission of sensitive payment data. If the storage of sensitive payment data is required, it should use a token as a replacement for the sensitive data.

There are examples of BLS included in the list below. Note that these examples represent common implementations commonly found in the hospitality industry and are provided for ease of understanding, but by no means are meant to be all-inclusive.

- Property Management Systems
- Central Reservation Systems
- Customer Relationship Management Systems
- Event, Sales and Catering Systems
- Profile Management Systems
- Global Distribution Systems
- Distribution Gateways and Switches
- Kiosks (depending on design)
- Back Office AR and Reconciliation Systems
- Remote Reservation Call Center Systems

In general, it is desirable to prevent these systems from being included in PCI Scope.

3.2.1 Payment Data Collection System (PDCS)

A Payment Data Collection System is responsible for the input of payment information. Examples include swipe and dip terminals or hosted payment capture systems.

3.2.2 Payment Card Information Source (PCIS)

A Payment Card Information Source is a source of payment information. In older systems, this may be a swipe device or the message returned from a financial institution, a web page or a partner such as a Central Reservations System, which collects the information and provides it to the merchant's systems.

3.2.3 Payment Data Provider (PDP)

The Payment Data Provider is the person or organization providing or entering the payment card information. This may be a guest, an associate acting for the guest or a system that has actual credit card information.

3.2.4 Tokenization Service

The Tokenization Service accepts Payment Card Information (PCInfo) and exchanges the PCInfo for a token. The Tokenization Service can also provide a means to exchange a token for the original card information and exchange keys that can be used to encrypt and secure the PCInfo as it is being transported between systems. The PCInfo is securely stored in a "Token Vault." The Tokenization Service role falls within the Payment Card Industry (PCI) scope since it handles plain-text card information.

3.2.5 Token Vault

A Token Vault stores sensitive data in a secure manner, often in an encrypted form. It manages the relationship between sensitive data and one or more tokens representing the secured data. It is assumed that in general, a token vault will be in PCI scope, but this is design dependent.

3.2.6 Key Service

A Key Service provides an encryption key that can be used to encrypt the payload of a message containing payment information.

3.2.7 Payment Information Proxy (PIP)

The Payment Information Proxy is an optional system that can sit between a PCIS and a BLS. The PIP system can accept a message containing PCI, extract the actual payment information and replace it with a token returned by the Tokenization Service. It can also reverse this process when needed.

3.3 Use Cases

These business use cases describe common uses for the Tokenization Service.

3.3.1 Exchange Payment Information for a Token

For any interaction between a system that needs to exchange payment information for a token (e.g., guest enters/updates cards stored with their profile and supplies a new credit card, guest makes a reservation with a credit card, etc.), the following will occur:

- The card information is provided to the PCIS and sent to the Tokenization Service to request a token. This may occur directly or when a message with payment information is sent to a PIP.
- The Token Service stores the payment information into the vault and returns a token.
- The token is then sent in place of the credit card and may be stored in the BLS.

3.3.2 Use a Token to Retrieve Payment Information

This use case may also be called "PAN recovery." PAN recovery may not necessarily be exposed in situations where the token service provider also processes the payment card financial transactions.

The HTNG Secure Payment Framework supports the concept of a PIP, as a service that sits between the systems that may normally receive messages containing payment information and the systems that send the payment information. The role of the PIP is to remove the payment information from a message and replace it with a token retrieved from the tokenization service. This may also work in the other direction, to remove the token information from a message and replace it with the appropriate payment information. This second instance requires the ability to retrieve payment information using the token. If the information is retrieved, the system receiving the payment information will be in full PCI-scope. Therefore, it is recommended that this should not be a business system such as a PMS, which should be kept out of scope.

This use case covers any interaction between a system that has a token and needs to exchange it for a payment information (e.g. a back-office employee needs to access the actual PAN to resolve a billing issue) through the following steps:

- The originating system sends the request to the Tokenization Service to request credit card information.
- The Token Vault replies with payment information, the date entered into the vault and the date last accessed.

Notes:

- The initial request may also provide a key, a key identifier, a key lifespan and an encryption algorithm that will be used to encrypt the contents of the message.
- The initial request may be sent through a PIP and the resulting message forwarded to a third in-scope system.

3.3.3 Request an Encryption Key to Encrypt Message Payload

When point-to-point security is required to send payment information to the vault, the system provides the ability to use asymmetric encryption using public and private keys to securely encrypt plain-text payment data before it is sent to the token service:

- The originating system requests a key to use to encrypt the information.
- The key services reply with a public key, a key identifier, a key lifespan and an encryption algorithm.
- The originating system can then use the key to encrypt the contents of the message described in Section 3.3.1.

3.3.4 Token Status Check

The Token Status Check is used to check if a token exists in a vault without recovering or restoring the information backing the token.

- The requesting system sends the message with the token to be checked.
- The service returns the date the token was created, the date the payment information was last accessed and the last four digits of the PAN.

4 Composite Scenarios

4.1 Adding Payment Information to a Payment System

4.1.1 Brief Description

When an actor in a Payment Data Provider role provides payment card information that needs to be stored in a BLS, the provided card information needs to be exchanged for a token, which is sent to the BLS.

4.1.2 Actors & Roles

Role Name	Definition
Business Logic System (BLS)	This is a hotel system that needs to store payment information, but also needs to be out of PCI scope. For example, this includes POS Systems, CRS, PMS and Reservation Systems.
Payment Data Provider (PDP)	This may be a guest, an associate acting for the guest or a system that has actual credit card information.
Payment Data Collection System (PDCS)	This is the system that collects the payment information prior to sending the information to the BLS. For example, this could be a web user interface.

4.1.3 Assumptions

N/A

4.1.4 Preconditions

N/A

4.1.5 Trigger

N/A

4.1.6 Basic Flow

1. The use case starts when the CDP provides information about a payment card to the DCS.
2. The DCS issues an HTNG_TokenizeRQ to the Tokenization Service and receives an HTNG_TokenizeRS in return.
3. The token is sent to the BLS.

4.1.7 Post Conditions

The token is stored in the BLS as a replacement for the payment card number.

4.1.8 Exceptions

N/A

4.1.9 Alternate Paths

N/A

4.2 Retrieving Full Payment Information using a Token

4.2.1 Brief Description

In this business scenario, an actor needs to access the actual PAN of a credit card, but only has access to the token. For example, a merchant associate may need to retrieve the full card number in order to respond to a chargeback or inquiry. Any system that can access actual payment information falls within PCI scope; therefore, it is recommended that the system used to retrieve the PAN is isolated from the BLS.

4.2.2 Actors & Roles

Role Name	Definition
Card Retrieval System	The secure system used to retrieve card information; for example, this could be a PIP or a secure retrieval terminal.
Tokenization Service	The Tokenization Service
Merchant Associate	The merchant or hotel associate that needs to access the full payment card information. This may be a person in a back-office or in an accounting role.

4.2.3 Preconditions

1. The merchant associate must be authenticated and authorized to view PCInfo.
2. The Card Retrieval System must be able to access the Tokenization Service.

4.2.4 Basic Flow

1. The use case starts when the merchant associate needs to access actual PCInfo.
2. The associate uses the Card Retrieval System to issue an **HTNG_DetokenizeRQ** to the Tokenization Service. The card retrieval receives an **HTNG_DetokenizeRS** in return.
3. The associate views the payment information on the Card Retrieval System.

4.2.5 Post-condition

1. The associate can view the PCInfo.

4.3 Checking Token Status

4.3.1 Brief Overview

In this scenario, a token has been stored in a Central Reservation System, but a new reservation is being made that requires a credit card guarantee. The reservation application needs to validate the token to determine if it is necessary to prompt for updated payment information.

4.3.2 Actors & Roles

Role Name	Definition

CRS	The Central Reservation System is an example of a BLS
Tokenization Service	A party that can accept and securely store Payment Card Data and return a tokenized representation of that data, and return the Payment Card Data when presented with the Token

4.3.3 Preconditions

The CRS has retrieved the token and expiration date for the payment card from the guest's profile information.

The CRS has credentials to authenticate itself to the Tokenization Service and is authorized to access the status operation.

4.3.4 Basic Flow

The use case starts when the CRS requires a guarantee for a new reservation.

1. The Reservation System sends the token to the de-tokenize operation of the Tokenization Service.
2. The Tokenization Service retrieves the status information and returns it to the requestor.

4.3.5 Post-condition

The CRS knows the token is valid.

4.3.6 Exception

“Token not found” will occur if the token value does not match a token in the vault.

4.4 Request an Encryption Key

4.4.1 Brief Overview

In this scenario, a group booking agent wants to send credit card information for a group reservation. The payment information is in plaintext, but the agency needs encryption information including a key and an algorithm to encrypt the information prior to sending it. The key will be retrieved from the token service and can be used, for example, to encrypt one credit card per row of a spreadsheet.

4.4.2 Actors & Roles

Role Name	Definition
Payment Data Provider (PDP)	In this scenario, the PDP is the booking agent for the group who needs to be able to encrypt credit card data prior to sending it to the Token Vault.
Payment Card Information Source (PCIS)	The PCIS is the system used by the booking agent to send the information.
Token Service	The getKey operation of the token service will provide the key.

4.4.3 Assumptions

The PDP has an application that can be used to retrieve the key and encrypt the information when provided an encryption key. For example, this may be a macro for a spreadsheet. Once the payment information is encrypted, the information does not need to go directly to the token service, but may be pre-processed by a BLS to send the information to the service one row at a time.

4.4.4 Preconditions

The PCIS is authenticated and authorized to use the getKey operation of the Tokenization Service.

4.4.5 Trigger

N/A

4.4.6 Basic Flow

1. The use case starts when the PCIS requests a key from the Token Service using the getKey operation.
2. The getKey operation returns a key.

4.4.7 Post-condition

The PCIS has a key and the information required to encrypt the payment information. This information can only be decrypted by the token service.

4.4.8 Exceptions

Key generation failure, which should result in an error message and an optional key generation retry.

5 Messages

5.1 Tokenize

These messages are used to request a token for the provided payment information. A user should indicate that paymentcardtype and trackdata are encrypted data types.

5.1.1 Data Element Table – Request

Element | @Attribute	Num	Data Type	Description/Contents
HTNG_TokenizeRQ	1		The root element of the message.
./Metadata	0..1	xs:anyType	This is used to hold an optional Metadata object, which can contain any elements needed for the Metadata. A couple of examples are a correlation ID and timestamps. However, in SOAP messages, this is typically carried in the SOAP header.
./EncryptedData	0..1	EncryptedDataType	If the PaymentCard element is encrypted, the EncryptedData element appears in place of the PaymentCard element. The EncryptedData element is defined by the W3C XML Encrypted standard.
./PaymentCard	0..1	PaymentCardType	This is the element to hold the information for a Payment Card. The PaymentCard element holds the Card Number, Expiration Data and Card Code elements.
./PaymentCard/CardCode	0..1	String	This optional field provides the Card Brand Code.
./PaymentCard/CardNumber	1	String	This is the Primary Account Number to be secured by the Tokenization Service.
./PaymentCard/ExpireDate	1	MMYYDate	This is the expiration date for the Primary Account Number.
xsd:any	0..N	any	This schema contains xsd:any elements. In an XML schema, Xsd:any allows the implementer to include any additional elements they choose. These elements can come from a different namespace and can be validated against a separate schema that defines them. This specification does not guarantee that these elements will be stored, or if stored, that they will be secured or returned upon detokenization request. These behaviors are implementation-specific.

Role Name	Num	Data Type	Description/Contents
./TrackData	0..1	TrackDataType	This is the element to hold card track data elements. This element may contain the track data for a single track or data for both track 1 and track 2. The Tokenization Service will only store the PAN and the Expiration Date.
./TrackData/Track	0..3	TrackType	This is the element that holds track data for a single track.
./TrackData/Track/@trackNo	1	String	This attribute identifies the track number.
./TrackData/Track/@encoding	0..1	Enumeration	This attribute determines if the track data is encoded as hex, ASCII, or base64 and the default is ASCII. If the attribute is not provided, the value is ASCII.
./Signature	0..1	String	This is an optional XML signature for the message.

5.1.2 Data Element Table – Response

Element | @Attribute	Num	Data Type	Description/Contents
HTNG_TokenizeRS	1		This is the root element of the message.
./Metadata	0..1	xs:anyType	This is used to hold an optional Metadata object which can contain any elements needed for the Metadata. A couple of examples are a correlation ID and timestamps. However, in SOAP messages, this is typically carried in the SOAP header.
./PaymentToken	1	PaymentTokenType	This is the element that holds the returned token.
./PaymentToken/Token	1	Restricted String	This is the token.
./PaymentToken/ExpireDate	1	MMYYDate	This is the card expiration date.
./OriginTime	1	dateTime	This is the ISO 8601 representation of the date and time when the card was first tokenized and stored in the vault.
./LastAccessed	1	dateTime	This is the ISO 8601 representation of the date and time when the card was last accessed from the vault.

5.1.3 Example Tokenize Request Messages

In each of the following examples, a token is needed for a new card. The token will then be stored into the PMS system.

This is the simplest request message that can be sent. If this is the body of a SOAP message, then Metadata, encryption and signatures will be handled by the SOAP environment if needed.

```
<HTNG_TokenizeRQ>
  <PaymentCard>
      <CardNumber>401288XXXXXX1881</CardNumber>
      <ExpireDate>0521</ExpireDate>
  </PaymentCard>
</HTNG_TokenizeRQ>
```

The same message in JSON notation:

```
{
  "HTNG_TokenizeRQ": {
    "PaymentCard": {
      "CardNumber": "401288XXXXXX1881",
      "ExpireDate": "0521"
    }
  }
}
```

The same XML message with Metadata:

```
<HTNG_TokenizeRQ>
  <Metadata>
      <TimeStamp>2016-12-28T22:15:35.059Z</TimeStamp>
      <CorrelationID>350590598765</CorrelationID>
  </Metadata>
  <PaymentCard>
      <CardNumber>401288XXXXXX1881</CardNumber>
      <ExpireDate>0521</ExpireDate>
  </PaymentCard>
</HTNG_TokenizeRQ>
```

The same XML message with the content encrypted:

```
<HTNG_TokenizeRQ>
  <Metadata>
      <TimeStamp>2016-12-28T22:15:35.059Z</TimeStamp>
      <CorrelationID>350590598765</CorrelationID>
  </Metadata>
  <EncryptedData    Type="http://www.w3.org/2001/04/xmlenc#Element"
                  xmlns="http://www.w3.org/2001/04/xmlenc#">
    <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <KeyName>Tue99</Keyname>
    </KeyInfo>
    <CipherData>
      <CipherValue>0123456789ABCDEF1A2B3C4D</CipherValue>
    </CipherData>
  </EncryptedData>
</HTNG_TokenizeRQ>
```

5.1.4 Example Tokenize Response Message

The Tokenization Service will answer the request by providing the following response:

```
<HTNG_TokenizeRS
  xsi:schemaLocation="http://htng.org/2016B HTNG_TokenizeRS.xsd"
  xmlns="http://htng.org/2016B"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Metadata>
      <TimeStamp>2016-12-28T22:15:35.059-05</TimeStamp>
      <CorrelationID>350590598765</CorrelationID>
  </Metadata>
  <PaymentToken>
    <Token>9617653287451118</Token>
    <ExpireDate>0121</ExpireDate>
    <OriginTime>2016-12-28T22:15:35.059-05</OriginTime>
    <LastAccessed>2016-12-28T22:15:35.059-05</ LastAccessed>
  </PaymentToken>
</HTNG_TokenizeRS>
```

5.2 Detokenize Messages

5.2.1 Detokenize Request

To obtain the payment card information for a previously obtained token, the HTNG_DetokenizeRQ message is used.

Element | @Attribute	Num	Data Type	Description/Contents
HTNG_DetokenizeRQ	1		This is the root element of the message.
./Metadata	0..1	xs:anyType	This is used to hold an optional Metadata object which can contain any elements needed for the Metadata. A couple of examples are a correlation ID and timestamps. However, in SOAP messages, this is typically carried in the SOAP header.
./PaymentToken	1	PaymentTokenType	This is the element that holds the returned token.
./PaymentToken/Token	1	Restricted String	This is the token.
./PaymentToken/ExpireDate	1	MMYYDate	This is the card expiration date.
./EncryptedKey	0..1	EncryptedKeyType	This information is needed to encrypt the contents of the expected response. In general, this will be an identifier used to access a previously shared encryption key, a public key or certificate used to encrypt the response. This element is based on the W3C XML encryption standard.

5.2.2 Detokenize Response

The HTNG_DetokenizeRQ is answered using the HTNG_DetokenizeRS message.

Element | @Attribute	Num	Data Type	Description/Contents
HTNG_DetokenizeRS	1		This is the root element of the message.
./Metatdata	0..1	xs:anyType	This is used to hold an optional Metadata object which can contain any elements needed for the Metadata. A couple of examples are a correlation ID and timestamps. However, in SOAP messages, this is typically carried in the SOAP header.
./EncryptedData	0..1	EncryptedDataType	If the PaymentCard element is encrypted, the EncryptedData element will appear in place of the PaymentCard element. The EncryptedData element is defined by the W3C XML Encrypted standard.
./PaymentCard	0..1	PaymentCardType	This is the element to hold the information for a Payment Card. The PaymentCard element holds the Card Number, Expiration Date and Card Code elements.
./PaymentCard/CardCode	0..1	String	This optional field provides the Card Brand Code.
./PaymentCard/CardNumber	1	String	The Primary Account Number is to be secured by the Tokenization Service.
./PaymentCard/ExpireDate	1	MMYYDate	This is the expiration date for the Primary Account Number.
./PaymentCard/xsd:any	0..n	any	This schema contains xsd:any elements. In an XML schema, Xsd:any allows the implementer to include any additional elements they choose. These elements can come from a different namespace and can be validated against a separate schema that defines them. This specification does not guarantee that these elements will be stored, or if stored, that they will be secured or returned upon detokenization request. These behaviors are implementation-specific.
./Signature	0..1	SignatureType	The optional XML signature is defined in the w3c Signature Syntax and the processing standard.

5.2.3 Global Sample Message – Request

The Card Viewing Terminal obtains the card information for a card previously stored by the Tokenization Provider by transmitting the following message:

```
<HTNG_DetokenizeRS>
  <PaymentToken>
      <Token>401288XXXXXX1881</Token>
      <ExpireDate>0521</ExpireDate>
  </PaymentToken>
</HTNG_DetokenizeRS>
```

5.2.4 Global Sample Message – Response

The Tokenization service will respond to the request by providing the following response:

```
<HTNG_DetokenizeRQ>
  <PaymentCard>
      <CardNumber>401288XXXXXX1881</CardNumber>
      <ExpireDate>0521</ExpireDate>
  </PaymentCard>
</HTNG_DetokenizeRQ>
```

5.3 Token Status Messages

This operation allows a token to be validated against the data in the vault without recovering the card information.

5.3.1 Data Element Table – Request

The following message is used to request the status of a token.

Element | @Attribute	Num	Data Type	Description/Contents
HTNG_TokenStatusRQ	1		This is the root element of the message.
./Metatdata	0..1	xs:anyType	This is used to hold an optional Metadata object which can contain any elements needed for the Metadata. A couple of examples are a correlation ID and timestamps. However, in SOAP messages, this is typically carried in the SOAP header.
./PaymentToken	1		This is the element that holds the returned token.
./PaymentToken/Token	1	Restricted String	This is the token.
./PaymentToken/ExpireDate	1	MMYYDate	This is the card expiration date.

5.3.2 Data Element Table – Response

The following is the response to a token status request.

Element | @Attribute	Num	Data Type	Description/Contents
HTNG_TokenStatusRS	1		This is the root element of the message.
./Metadata	0..1	xs:anyType	This is used to hold an optional Metadata object which can contain any elements needed for the Metadata. A couple of examples are a correlation ID and timestamps. However, in SOAP messages, this is typically carried in the SOAP header.
./Status	1	statusType	This is the element which contains status information.
./Status/OriginTime	1	dateTime	This is the ISO 8601 representation of the date and time when the card was first tokenized and stored in the vault.
./Status/LastAccessed	1	dateTime	This is the ISO 8601 representation of the date and time when the card was last accessed from the vault.
./Status/LastFour	0..1	integer	These are the last four digits of the original PAN.
./Status/AccessCount	1	Integer	This shows the number of times the PAN has been recovered.
xs:any	0..N	xs:any	This schema contains xsd:any elements. In an XML schema, Xsd:any allows the implementer to include any additional elements they choose. These elements can come from a different namespace and can be validated against a separate schema that defines them. This specification does not guarantee that these elements will be stored, or if stored, that they will be secured or returned upon detokenization request. These behaviors are implementation-specific.

5.3.3 Global Sample Message – Request

In this request, a token is being provided in order to retrieve the status of the token:

```
<HTNG_TokenStatusRQ>
  <PaymentToken>
      <Token>401288XXXXXX1881</Token>
      <ExpireDate>0521</ExpireDate>
  </PaymentToken>
</HTNG_TokenStatusRQ>
```

5.3.4 Global Sample Message – Response

The status of the token provided in the request is shown in this sample response message:

```
<HTNG_TokenStatusRS
  xsi:schemaLocation="http://htng.org/2016B HTNG_TokenStatusRS.xsd"
  xmlns="http://htng.org/2017A"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <TokenStatus>
    <OriginTime>2016-12-28T22:15:35.059-05</OriginTime>
    <LastAccessed>2016-12-28T22:15:35.059-05</LastAccessed>
    <AccessCount>3</AccessCount>
    <LastFour>1881</LastFour>
  </TokenStatus>
</HTNG_TokenStatusRS>
```

5.4 Key Exchange Messages

5.4.1 Data Element Table – Request

This message is used to retrieve a key from the server that can be used to establish a session or to encrypt information that needs to be sent to the Tokenization Service.

Element | @Attribute	Num	Data Type	Description/Contents
HTNG_GetKeyRQ	1		This is the root element of the message.
./Metatdata	0..1	xs:anyType	This is used to hold an optional Metadata object which can contain any elements needed for the Metadata. A couple of examples are a correlation ID and timestamps. However, in SOAP messages, this is typically carried in the SOAP header.
./KeyType	0..1	string	The desired format for the key is usually X.509 or PEM.

5.4.2 Data Element Table – Response

This message is the response to the HTNG GetKeyRQ message.

Element | @Attribute	Num	Data Type	Description/Contents	
HTNG_GetKeyRS	1		This is the root element of the message.	
./Metatdata	0..1	xs:anyType	This is used to hold an optional Metadata object which can contain any elements needed for the Metadata. A couple of examples are a correlation ID and timestamps. However, in SOAP messages, this is typically carried in the SOAP header.	
./EncryptedData	1	EncryptedDataType	If the KeyInfo element is encrypted, the EncryptedData element will appear in its place. This element is defined by the W3C XML Encrypted standard.	There is only one instance of these three types: EncryptedDataType, KeyInfoType and EncryptedKeyType which may be chosen to occur in the message. Encrypted data can contain either the KeyInfo or the EncryptedKey element. KeyInfo may in turn, also contain an EncryptedKey Element or the EncryptedKey element may be used by itself.
./KeyInfo	1	KeyInfoType	This is the KeyInfoType from W3C XML-Signature.	
./EncryptedKey	1	EncryptedKeyType	This is the EncryptedKeyType from the W3C XML Encryption.	
./KeyExpiration	0..1	dateTime	This is the ISO 8601 Date/Time when the key must be renewed. This is optional for those types of keys or certificates that do not self-contain expiration information.	
./Signature	0..1	SignatureType	The optional XML signature is defined in the w3c Signature Syntax and the processing standard.	

5.4.3 Global Sample Message - Request

In this example, a public key is requested in the X509Data format:

```
<HTNG_GetKey>
   <KeyType>http://www.w3.org/2000/09/xmldsig#X509Data</KeyType>
</HTNG_GetKey>
```

5.4.4 Global Sample Message - Response

This is an example of a simple response to the HTNG_GetKey request:

```
<HTNG_GetKeyRS>
  <KeyInfo>
    <X509Data>
      <X509Certificate>MIICSTCCA...</X509Certificate>
    </X509Data>
  </KeyInfo>
</HTNG_GetKeyRS>
```

6 Appendices

6.1 Implementation Notes

6.1.1 Token Format

The HTNG Payment Workgroup understands that merchants and service providers may have differing requirements for the format of the token. This specification does not define the token format for any specific implementation, leaving that choice to the implementer. On the other hand, many merchants and hoteliers will desire tokens that are compatible with their legacy systems. These "legacy tokens" should have the following attributes:

- The legacy token must be numeric
- The token will pass a LUHN check
- Optionally, the last four digits of the legacy token match the last four digits of the card's PAN

When possible, it is recommended to use the longest character length available (19) for the PAN to have the largest diversity of tokens that support these legacy restrictions.

It is important to note that in all cases there should be no mathematical relationship between a token and the PAN.

In the previous version of this specification (The "Data Proxy" spec), HTNG established parameters that could be defined to govern the creation of the token. It was suggested that the service provider would associate the parameters with each merchant to allow customization of the token based on the merchant’s needs. The parameters are provided here for continued compatibility with existing systems:

- Token length
- Numeric or alphanumeric
- Specific bytes can represent the original card type (for example, if the second byte is a 4, then the original card was a Visa card)
- Passes a MOD-10 check

The following items will be fixed for all implementations of the token:

- It will retain the last 4 digits of the original card number
- It must be mathematically unrelated to a credit card number

6.1.2 Tokenizing from Track Data

The Tokenization Service supports the creation of a token from card track data. It is assumed that the PAN and expiration date will be extracted from the track data and the rest of the data will be immediately discarded in compliance with PCI rules. There is an allowance within PCI rules to hold track data (and possibly the Card Verification Value) until the card has been authorized. It is recommended that this information is not held in persistent storage. If it is required to be held in persistent storage for a short period of time, the information must be fully and securely encrypted until processed. Once the authorization is processed, data other than the PAN and Expiration Date should be securely deleted from both volatile and persistent storage.

6.1.3 Adding Elements to the Tokenize Message

The schemas for the messages are designed to be extensible. Extension schemas should define their own namespaces to prevent conflicts with the elements defined in the HTNG and OpenTravel schemas. We expect that a common set of extensions will add information required to perform a zero authorization on the card prior to tokenization. This information would normally include one or more of the following elements:

- Customer name as it appears on the card
- Customer card billing address
- Customer card zip or postal code
- CVV or other card security code

If this additional information is sent as a part of the tokenization message, it is recommended that the information is not included in the vault and that it is securely cleared from both volatile and non-volatile or persistent memory once the authorization has been completed.

A primary advantage to this approach is keeping the PAN and other information separate, making compromise more difficult.

6.1.4 DUKPT and the W3C KeyInfo Element

The financial industry commonly uses a Derived Unique Key Per Transaction (DUKPT) to manage keys used to securely encrypt payment information sent from devices like Automated Teller Machines and card swipe terminals to the financial company's payment infrastructure. The XML Encryption and XML Signature specifications created by the World Wide Web Consortium (W3C) provides a KeyInfo element for the purpose of providing material for understanding how to decrypt a message, but provision was not explicitly made to support the needs of DUKPT. DUKPT requires a Key Sequence Number (KSN) to be sent with each cipher text (encrypted message content) so the receiving system can derive the key needed to decrypt the message. The KSN is defined by a current X9.24 standard as a binary value 80 bits long. In order to ensure consistent implementations, HTNG provides a schema extension to be used within the KeyValue element which is within the KeyInfo element defined in the XML Encryption Specification.

Element | @Attribute	Num	Description/Contents
KeyInfo/KeyValue/KSN	1	The KSN needed to determine the key used to decrypt the information. This is provided using hex-ascii encoding (ws:hexBinary) of the 80 bit KSN. Note the length of KSN change when the new AES based DUKPT has been standardized.

A sample of the resulting HTNG_TokenizeRQ may look like the following message:

```
<HTNG_TokenizeRQ>
  <Metadata>
      <TimeStamp>2016-12-28T22:15:35.059-05</TimeStamp>
      <CorrelationID>350590598765</CorrelationID>
  </Metadata>
  <EncryptedData    Type="http://www.w3.org/2001/04/xmlenc#Element"
                    xmlns="http://www.w3.org/2001/04/xmlenc#">
    <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <KeyValue xmlns:htng="http://ns.htng.org/2017A/pmts">
            <htng:KSN>FFFF9876543210E00008</htng:KSN>
        </KeyValue>
    </KeyInfo>
    <CipherData>
      <CipherValue>0123456789ABCDEF1A2B3C4D</CipherValue>
    </CipherData>
  </EncryptedData>
</HTNG_TokenizeRQ>
```

6.1.5 Bulk File Processing

The Payments Workgroup understands there may be use cases that require the processing of a large volume or bulk sets of credit card numbers. While the methods defined in this document allow the processing of these numbers one at a time, they may prove to be inefficient in large volumes.

Some examples include:

- A merchant who has only tokens needs to transmit folio information to a third-party provider who requires actual payment card data.
- A merchant needs to retrieve all payment card numbers it has on file with the tokenization service provider.
- A merchant requiring the transfer of payment card numbers from a tokenized system to a system that requires actual payment information.

This need may be addressed in the next release of this specification.

6.2 Communications and Security

Integrators should use standard PCI-approved methods for communicating over private and public networks. In addition, because the Token Vault is storing a large volume of sensitive data, we recommend that integrators use extra layer(s) of security for authenticating clients who are performing transactions that retrieve sensitive data from the Token Vault. These additional methodologies may include the following:

- Mutual authentication of TLS connections and digital certificates
- Multifactor authentication

6.3 Faults

Subcode	Methods	Description
KeyGenerationFailed	GetKey	This subcode will be displayed when it has failed to create an encryption key.
UnsupportedKeyType	GetKey	This subcode will be displayed when the requested keytype is not supported by the implementation.
Invalid Key Material	GetKey	This subcode will be displayed when the provided key material is not valid for the requested key.
TokenNotFound	TokenStatus Detokenize	This subcode will be displayed when the token requested was not found in the vault.

6.4 Links

The following table lists useful links for understanding and implementing this specification:

Document Title	Location / URL
PCI-DSS documentation	https://www.pcisecuritystandards.org/
XML Encryption	https://www.w3.org/TR/2013/REC-xmlenc-core1-20130411/
XML Signature	https://www.w3.org/TR/xmldsig-core2/

6.5 Referenced Documents

The HTNG Data Proxy and Secure Payments Framework Documentation can be found at:
http://www.htng.org/page/SpecsbyProductType