Token and Pre-auth Services

Introduction

In Sales Connector we have one Token Service and one Pre-auth Service. The Token Service generates a reference to a given PAN while the Pre-auth Service generates a reference to a given authorisation transaction (including card data). The PAN-reference enables matching of the customer and is a unique token for the card within its expiry and scope. Hence, the Token Service is central to the VAS (Value Added Service) solution we have in Sales Connector. The pre-auth reference is a pre-auth ID, it enables central storage of PCI sensitive information and enables finalisation of a transaction without storing the card data locally in our terminals.

In this document we will describe both the Token Service and the Pre-auth Service by outlining their design, functionality, rules, interfaces and use-cases.

Details of all the request and response messages are documented in TCS PRP Payload Format.

Token Service

The Token Service is a card-token solution. The token is a unique PAN-reference within a company scope or within a service scope and enables recognition of a customer based on his or hers card.

Functionality and command details

The token itself is a randomly generated and unique long value of 17-19 digits.

We have the following token functions/commands available:

- Get token. Will return one token or list of tokens that matches the parameters.
    - Get existing active token. Parameter: PAN and scope
    - Get token forced, generate token if not exist. Parameters: PAN, expiry and scope.
- Update token. Update an existing token. Parameters: PAN or token ID, expiry and scope
- Delete token. Delete an existing token. Parameters: Token ID. The token will get status ‘deleted’, and will be removed later, based on the clean-up task.

Planned improvements and extensions:

- Search token. Will be a special variant of get token that get all tokens, also the ones expired or status deleted. Can i.e. be used to update an expired token when
- Get token forced, generate token if not exist. In situations where one datacenter is down, we cannot be 100% sure that all tokens are syncronised. In the get token forced scenario, we must avoid to generate a new token if it already exist in another datacenter. Therefore we would need a flag in tcs-terminal telling that we are syncronised and if not, a new responsecode telling that tcs-terminal cannot generate a new token in this datacenter at the moment. Problem is that this migth cause "System unavailable" situations, and the requester migth need to switch datacenter to retrieve the token.

Token scope and expiry

All tokens have a scope and an expiry. The scope will limit who can access the token and the expiry defines for how long it is valid. We have two types of scopes; Company scope or Service scope. Common for both is that the terminal or client that is requesting a token must be within the token scope to access it. The expiry defines at what time the token expires and an expired token is not available anymore.

Company related scope

For single merchant specific tokens and for tokens to be used within one business organisation, the Company scope level must be defined.

The company related token scopes are:

- Terminal group
- Company

To use a token you need to be within the scope that is defined for that token. So the scope defines the token limitation. I.e. to access a token with Terminal Group scope, the requesting terminal must be within the same Terminal Group that is defined for that token.

So there may be several tokens for one PAN, one for each scope level. You can get one token for the Terminal Group scope and one for the Company scope, if you have access to both.

It is suggested to expand the scope limitations to include all these levels:

- Terminal
- Terminal group (implemented)
- Merchant / Bank connection
- Company (implemented)
- Company Group (share between several companies)

Service related scope and VAS-token

For tokens to be used across merchants or business organisations but within one service only, the Service scope must be defined. A service is typically a loyalty program or some other VAS program.

To use a token you need to be within the same service that is defined as the scope for that token. I.e. to access a token with scope service code VAS_ID1, requesting terminal must belong to a Terminal Group of Company that is associated with that service code VAS_ID1.

Main features

- Cross merchant token within one service. The VAS Service Code defines the token scope.
- It is only one active token per PAN within a VAS Service Code to enable one unique match
- VAS Service Code association for a Company or Terminal Groups is administered in TCS Management and partly in TCS Client.

VAS Provider Integrations

The VAS Router in FTRS is documented in VAS Routing Design.

Current 3rd Party VAS Provider Integrations are documented in VAS Provider Integrations.

Future features

- Split token into internal and external token, one visible to merchants and one visible to VAS Provider. May be required due to VISA Best Practices for Tokenisation. A service that include "all" merchants will not be compliant.
- Rotate or update the token value i.e. once a year to avoid misuse of old tokens

Token expiry

The token expiry defines for how long the token is available for usage. The expiry is selected by the requester, and should be set to cover normal usage of the token.

General rules for expiry:

- Can be specified in number of days or timestamp. Shortest one is selected if both are given.
- Must be between now and 10 years. Expiry back in time is not allowed.

General expiry suggestions:

- Expiry can be set to expiry date of the card. On service-token we recommend to add some period after card expiry as well to be able to match the customer for some extended time, but no longer than 12 months.
- If the token is to be used only for a short period and there are no need to keep track of the same customer later on, the expiry should be set for that period only.

Expired tokens can be extended by the update token command.

Use cases

<This sections will be updated..>

Movie theater / Pay online -pick up at store

Currently the "movie theater" use case is supported by Sales Connector. A quick description of the use case:

- Customer orders a ticket on the Web. Point Gateway creates a token for the card number.
- The token is stored in the movie theaters ERP system.
- Customers inserts his/her card in a payment terminal at the movie theater.
- A request with the card number is sent to Point Gateway.
- The TCS finds the correct token and returns it.
- The terminal returns the token to the ERP system.
- The ERP system can print out the correct ticket.

Parking

<This sections will be updated..>

Loyalty customer

<This sections will be updated..>

Pre-auth Service

The Pre-auth Service is a transaction-token solution. The transaction-token or pre-auth ID is a reference to one authorisation transaction and its card data. So after an authorisation transaction is approved, the pre-auth can be stored in Sales Connector. Later when exact amount is known, the card-data can be retrieved by requesting the pre-auth ID and the finalisation or capture transaction can be performed without need for new card insert or any local storage of any PCI sensitive card data.

The pre-auth has scope, expiration and status. We also store optional merchant reference to the pr-auth for easy retrieval.

Functionality and command details

The pre-auth ID (token) itself is a randomly generated and unique long value of 17-19 digits.

We have the following pre-auth functions/commands available:

- Save pre-auth info. Save an authorisation transaction and get the pre-uth ID (token) in return. Parameters: Transaction packet, expiry, scope, reference
- Get pre-auth info. Get a stored pre-auth or a list of active pre-auth ID’s. Parameters: Pre-auth ID, PAN, scope, original terminal serial, receipt number, transaction time, reference. Search by pre-auth ID will give complete pre-auth in return, using the other values will give list of ID’s in return if mpre that one match (to avoid too large response message)
- Update pre-auth info. Updates an existing pre-auth, usually with new status or expiry or transaction data. Parameters: Pre-auth ID, status, expiry, scope, transaction packet.
- Delete pre-auth. Deletes a given pre-auth. Will set status ‘deleted’. The pre-auth will then be removed by the clean-up task later on. Parameter: Pre-auth ID.

Marked specific usage

The Pre-auth Service is currently being used in Finnish, Norwegian and Australia/New Zealand.

Finland

Norway

Autsralia / New Zealand

Use cases

Hotel solution

The hotel solution use case is supported by the Point gateway. A quick description of the use case:

- Customer orders a hotel room on the Web. Point Gateway creates a pre-auth-ID for the authorisazation /incl card number.
- The ID is stored in the hotel ERP system.
- At show up, the customers inserts his/her card in a payment terminal at the hotel desk.
- A request with the card number is sent to Point Gateway.
- The TCS finds the correct pre-auth and returns the ID. It could be more than one.
- The terminal returns the ID to the ERP system.
- The ERP system can match the order.
- At check out, the ERP system can do a capture with ID and correct amont and send this to Point Gateway.
    - Capture in terminal: ERP system send pre-auth-ID and total amount to terminal. Terminal request the pre-auth from TCS. Then use received data to complete the transaction and send ti to TCS for clearing. Optional is to request a pre-auth delete to TCS.
    - To be done: Implement additional authoriszation / update of an existing pre-auth.
    - To be done: Define what happens if teminal is offline at check out. It is possible to store the capture with only pre-auth-ID and total amount. Then when received in TCS, the card-data may be requested there. Or requested in terminal when terminal is online and ready to send its stored transactions.
    - To be done: Expiry of pre-auth. Treatment of expiry og an authoriszation is not completed.

Petrol self service

<This sections will be updated..>

Token and Pre-auth Implementation and Interfaces

The Token and Pre-auth Services are located in TCS TerminalServer, and the tokens and pre-auts are stored in TCS RealTime (RT) database:

- rt_token
- rt_pre_auth

Main internal logic is implemented in the tcs-terminal HibernateTcsRtDbRepository class while the requests are handled by the PrpConnection class in tcs-terminal.

The Token and and Pre-auth Services has the following interfaces:

- Directly from POS terminal to TCS Terminal as TCS PRP operation codes
- As part of online auth processing in FTRS
- Ecom payment interface
- Ecom server interface

To ensure active-active synchronisation between datacenters we are both distributing new tokens and pre-auths, and we regularly syncronise the tables.

To ensure that outdated and deleted tokens and pre-auths are not filling up the databases we have implemented clean-up scripts.

Administration of tokens and pre-auths are also added to TCS Management and TCS Client.

TCS Terminal PRP Interface

- A token can be requested directly from the TCS terminal server. Relevant data such as card number, token expiration, token scope etcetera must be set in the PRP message to TCS terminal server.
    - Operation codes: Get token, delete token
- A pre-auth-ID can be requested directly from the TCS terminal server. Relevant data such as authorization transaction, card number, pre-auth expiration, must be set in the PRP message to TCS terminal server.
    - Operation codes: Send pre auth info, get pre auth info, delete pre auth info

The PRP format is described in TCS PRP Payload Format.

From FTRS during online authorization

NetsNO Sink node

To be able to request a token or a send pre-auth request from an authorization request from the terminal, a TCS handler in the FTRS has been implemented. By implementing AsyncOpRequester and AsyncOpResponseHandler in the relevant sink node requests and responses to and from the TCS can be handled asynchronously. The SinkResponse has been expanded to support a List of tokens or pre-auth ID.

Payment Interface

In the ecom payment interface PPG - eCommerce Interface Reference - Hosted Pages we have these fields available for the Token Service:

- i-t-1-1_register-token
- s-t-1-36_service-code

We have not added interface for the Pre-auth Service yet.

Server Interface

In the ecom server interface PPG - eCommerce Interface Reference - Server to Server. we have these functions available for the Token Service:

- Save Token (save-token)
- Get Token (get-token)
- Delete Token (delete token)

We have not added interface for the Pre-auth Service yet.

Synchronisation and active-active

Since the Token and Pre-auth Services are located in TCS Terminal server and since we have an infrastructure with several separate datacenters, we need active-active implementations to ensure that data are synchronised.

For all new tokens and pre-auths and any changes to these, we are distributing these changes to the other database instances. This is done in the tcs-terminal RtBackgroundTaskHandler class and the RtBackgroundTaskWorker class there.

The RtBackgroundTaskWorker is dependend of configuration in TCS Terminal, tcs-terminal-properties:

- tcs_db_conf_file=/opt/point/tcs-db.yml

and not using old config

- tcs_rt_hibernate_properties_files=/opt/point/tcs_rt.properties

Token synchronisation

In addition to RtBackgroundTaskHandler in TCS-Terminal, TCS-BankServer performs periodical token synchronization between RT instances. TCS-BankServer uses tokens "modified" property to get most recent token from all RT instances and synchronize it to others.

TCS-Bank token synchronization is dependent on configuration in tcs-bank.properties

- tcs_rt_hibernate_properties_files

Pre-auth synchronisation

In addition to RtBackgroundTaskHandler in TCS-Terminal, TCS-BankServer performs periodical pre authorization synchronization between RT instances. TCS-BankServer uses pre authorization "modified" property to get most recent pre authorization from all RT instances and synchronize it to others.

TCS-Bank pre authorization synchronization is dependent on configuration in tcs-bank.properties

- tcs_rt_hibernate_properties_files

and can be enabled or disabled by property in tcs-bank.properies filed

Generation of the unique token and pre-auth ID

The implementation for generating the token or ID is done by function allocateRandomId in the HibernateTcsRtDbRepository class. It generates a long vaule based on a algorithm for random number genereration. To avoid duplicates we are checking the local database-instance before it returns the value.

Planned improvements:

- To avoid the possibility for duplicates that may occur if generation and check is done before syncronisation is performed, we are planning to let the database instance-id be part of the value:
    - ID=<random value>+<DB ID> or ID=<DB ID> + <random value>, totally within range of the long datatype.
    - Since the database instance-ID always will differ, we are 100% sure that the value will be unique.

Clean-up

Clean-up of tokens and pre-auths are handled by TCS BankServer.

In ConfigurationSynchronizer the rules for delete are added to CLEANUP_STATEMENTS from branch 1.36.

Delete/remove tokens on these criterias:

- Status "Open" or not set and expired 12 months ago
- Status "Closed" and expired 3 months ago
- Status "Deleted" and modified 3 months ago

Delete/remove pre-auths on these criterias:

- Status "Open" or not set and expired 12 months ago
- Status "Closed" and expired 3 months ago
- Status "Deleted" and modified 3 months ago

TCS Management

In TCS Management 2.0 we have separate tabs for viewing tokens and pre-auths. Granted users can also edit the token and pre-auth details.

TCS Client

We have currently not added any functionality for token or pre-auths to TCS Client.

Planned development:

- Search option for token and pre-auths for the Company users
- Edit option for token and pre-auths for the Company users