TIM API

-Integration information and validation procedure

Market Austria

Version 2022.11

Änderungshistorie

Version	Datum	Autor	Änderungen
2022.05	02.05.2022	A741128	New Design, ammendments
2022.10	09.11.2022	W129032	New Content, ammendments

INHALTSVERZEICHNIS

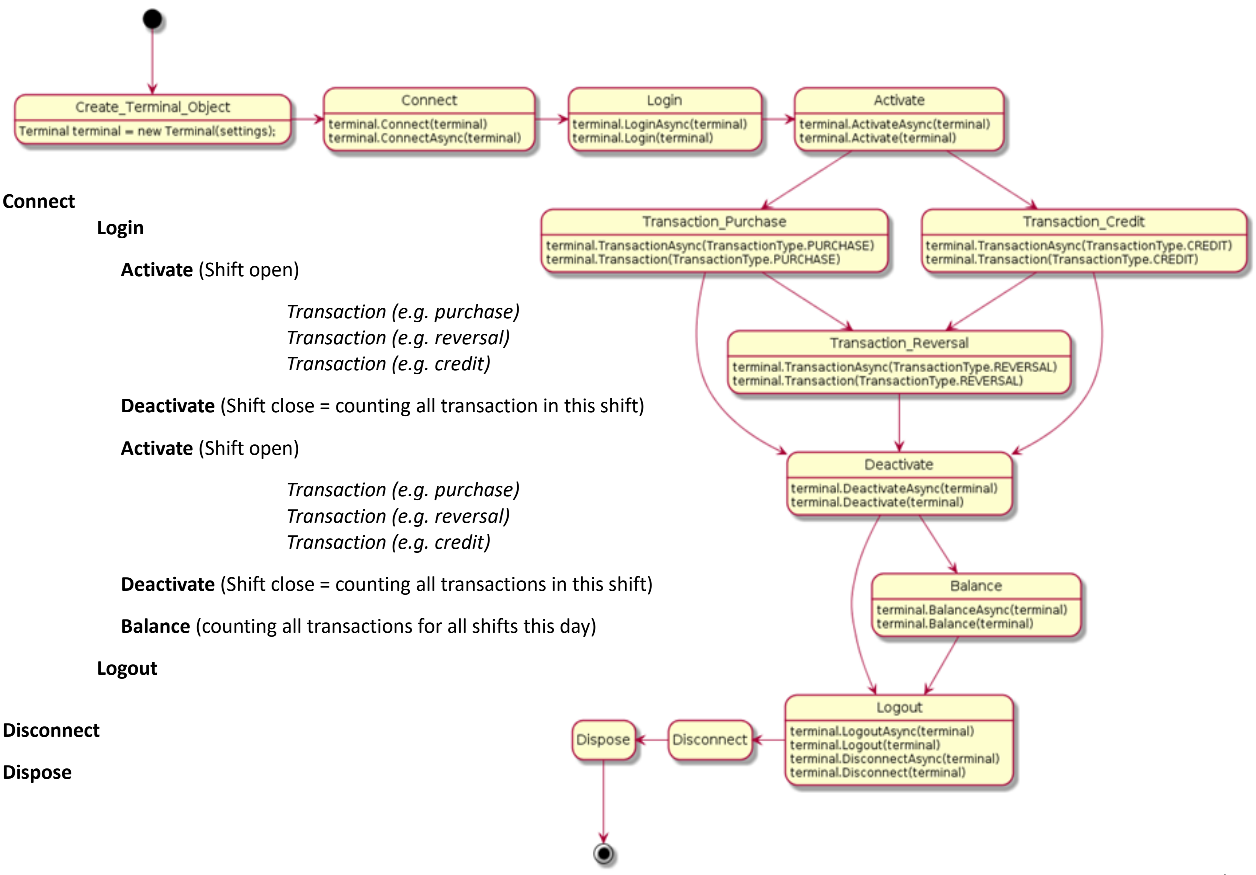
1 GETTING STARTED

Before you start an integration with our TIM API, make sure you have designed your integration. This in-cludes:

1. Take care about mandatory use cases.
2. Determining receipt data handling and log file storage principles.
3. A good understanding of different states of terminal link and shift mode.
4. Think about optional use of available information during transaction (eg. Transaction status and display information of terminal) for info-display of operator.
5. Provided test kit operates like a live terminal, please use low transaction values and correct PINs. Card limits and wrong PIN counters are active.
6. Prepare your network, if firewalls are enabled.

Connection and setup for TIM API

- **Terminal object settings**: the terminal settings have to be set before you generate the terminal object! In these settings, you have to set the Integrator ID (UIID) which identifies the integrator of the used application and is provided by Worldline. The logging destination and level is part of the setup as well. Decide, either you set your connection to broadcast in the network with a specific Terminal ID (TID) or connect it via a fix IP number. If you use JavaScript, the connection will be done by WebSocket with a fixed IP number and port 8080 (broadcast is not available). Basic Socket connections run on Port 7784. Be aware that we do not support secure WebSocket at this time (HTTPS).
- **Example of a terminal object creation**:


Create_Terminal_Object
Terminal terminal = new Terminal(settings);
Connect
terminal.Connect(terminal)
terminal.ConnectAsync(terminal)
Login
terminal.LoginAsync(terminal)
terminal.Login(terminal)
Activate
terminal.ActivateAsync(terminal)
terminal.Activate(terminal)
Transaction_Purchase
terminal.TransactionAsync(TransactionType.PURCHASE)
terminal.Transaction(TransactionType.PURCHASE)
Transaction_Credit
terminal.TransactionAsync(TransactionType.CREDIT)
terminal.Transaction(TransactionType.CREDIT)
Transaction_Reversal
terminal.TransactionAsync(TransactionType.REVERSAL)
terminal.Transaction(TransactionType.REVERSAL)
Deactivate
terminal.DeactivateAsync(terminal)
terminal.Deactivate(terminal)
Balance
terminal.BalanceAsync(terminal)
terminal.Balance(terminal)
Logout
terminal.LogoutAsync(terminal)
terminal.Logout(terminal)
terminal.DisconnectAsync(terminal)
terminal.Disconnect(terminal)
Disconnect
Dispose

Connect

Login

Activate (Shift open)

Transaction (e.g. purchase)
Transaction (e.g. reversal)
Transaction (e.g. credit)

Deactivate (Shift close = counting all transaction in this shift)

Activate (Shift open)

Transaction (e.g. purchase)
Transaction (e.g. reversal)
Transaction (e.g. credit)

Deactivate (Shift close = counting all transactions in this shift)

Balance (counting all transactions for all shifts this day)

Logout

Disconnect

Dispose

Legend	connect():	establish a connection between ECR and terminal
	login():	the ECR will login into the terminal (no other ECR can login to same terminal!)
	activate():	opens the shift (if shift was open before and not balanced, it will continue)
	deactivate():	closes the shift and creates counter for all transactions during last shift
	logout():	the ECR will logout from the terminal
	disconnect():	connection between ECR and terminal will disconnect
	dispose():	cleans up the terminal object from the memory

2 BUSINESS GUIDELINES

In this section, the business side requirements for a practical implementation are explained in more detail. To avoid corrections and/or mandatory changes at a later date please do not hesitate to contact us in case of uncertainties. This way the development can be done efficiently on your side and a quick aproval of the solution can normally be confirmed.

2.1 MANDATORY USECASE SET – BASIC RETAIL SOLUTIONS

The TIM API offers a very wide range of functions and information. It is not necessary to implement all offered functions. The minimum set is listed in the following table.

Operation Mode	TIM API Use Case Name	TIM API Settings
attended	Purchase Reversal (last transaction only) Credit Balance	DCC enabled AdditionalMerchantData used
attended Gastro	Purchase Reversal Credit Balance	DCC enabled AdditionalMerchantData used TIP activated
unattended Retail **(simple Vending solutions / SelfOrder systems)**	Purchase Balance	DCC activated AdditionalMerchantData used Guide.Unattended added

Attended / Unattended information

- **Attended ECR**: the merchant is present, initiates the transaction by an ECR and is able to verify the card holder (e.g. signature, PIN, …).

- **Unattended ECR**: there is NO merchant present (e.g. EV Charging, Kiosks, …). For this setup, you have to configure the TIM API accordingly as UNATTENDED (guide) in the terminal object. Some functionalities like e.g. TIP are not allowed. An unattended ECR is only allowed in combination with an unattended Terminal Type. Please consult your Worldline POS Integration contact person for more details. Note that a terminal will automatically reboot every 24 hours between 22h00 and 06h00 (PCI regulation). You can prevent this, if the time slot is inconvenient for you, by setting a Reboot transaction in your application every 24 hours at a certain time of preference. By that way, it will prevent rebooting between 22h00 and 06h00.

2.1.1 ADDITIONAL DATA

The ECR system can add reference data in transaction requests. This can be very useful for investigation and reconciliation tasks in customer systems, such as accounting software or processes.
Simply add reference data in the TIM API with MerchantOptionType AdditionalMerchantData in the MerchantOptions.
All Worldline/PAYONE back office tools report this reference in any exported data format, e.g. online view in the merchant portal, downloaded CSV files, MRX etc.

Recommendation: A customer will enter this data in search fields. The data format should have a simple character set and a short length.

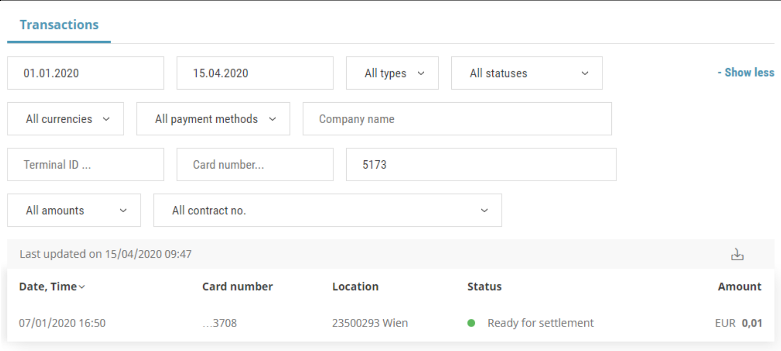
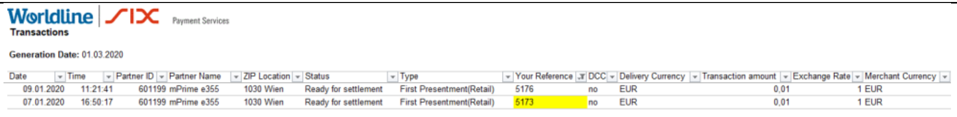
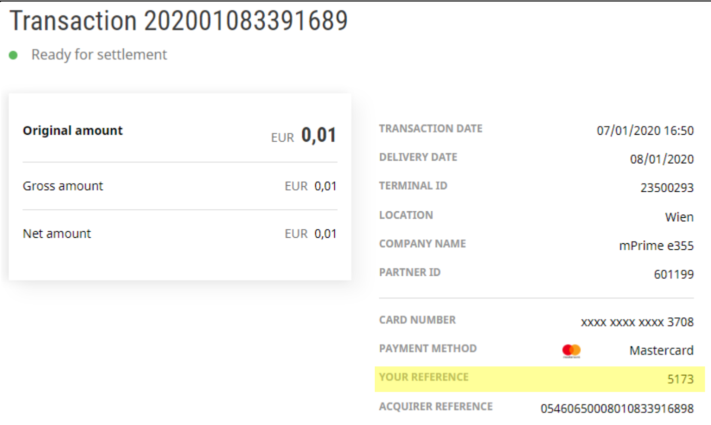
Example: A payment transaction should be done with AdditionalMerchantData = ‘5173’

```
TransactionRequest transactionRequest = new TransactionRequest();
transactionData.setDccAllowed(true);
transactionRequest.setTransactionData(transactionData);
transactionRequest.setAmount(new Amount(1.11, Currency.EUR));

List<MerchantOption> merchantOptions = new ArrayList<>();
merchantOptions.add(new MerchantOption(MerchantOptionType.ADDITIONAL_MERCHANT_DATA, "5173"));
transactionRequest.setMerchantOptions(merchantOptions);

TransactionResponse transactionResponse = terminal.transaction(TransactionType.PURCHASE, transactionRequest);
```

References:
https://six-tim.github.io/timapi/doc/cs/doc/api/SIX.TimApi.Terminal.html#SIX_TimApi_Terminal_MerchantOptions
https://six-tim.github.io/timapi/doc/cs/doc/api/SIX.TimApi.MerchantOption.html
https://six-tim.github.io/timapi/doc/cs/doc/api/SIX.TimApi.Constants.MerchantOptionType.html

search for reference data '5173' in portal:	
downloaded Excel/CSV data:	
detailed transaction view in portal	

Transactions

01.01.2020 | 15.04.2020 | All types | All statuses | - Show less

All currencies | All payment methods | Company name

Terminal ID ... | Card number... | 5173

All amounts | All contract no.

Last updated on 15/04/2020 09:47

Date, Time	Card number	Location	Status	Amount
07/01/2020 16:50	...3708	23500293 Wien	● Ready for settlement	EUR **0,01**

Worldline | SIX Payment Services

Transactions

Generation Date: 01.03.2020

Date	Time	Partner ID	Partner Name	ZIP Location	Status	Type	Your Reference	DCC	Delivery Currency	Transaction amount	Exchange Rate	Merchant Currency
09.01.2020	11:21:41	601199	mPrime e355	1030 Wien	Ready for settlement	First Presentment(Retail)	5176	no	EUR	0,01	1	EUR
07.01.2020	16:50:17	601199	mPrime e355	1030 Wien	Ready for settlement	First Presentment(Retail)	5173	no	EUR	0,01	1	EUR

Transaction 202001083391689

● Ready for settlement

Original amount	EUR **0,01**
Gross amount	EUR 0,01
Net amount	EUR 0,01

TRANSACTION DATE	07/01/2020 16:50
DELIVERY DATE	08/01/2020
TERMINAL ID	23500293
LOCATION	Wien
COMPANY NAME	mPrime e355
PARTNER ID	601199
CARD NUMBER	xxxx xxxx xxxx 3708
PAYMENT METHOD	Mastercard
YOUR REFERENCE	5173
ACQUIRER REFERENCE	05460650008010833916898

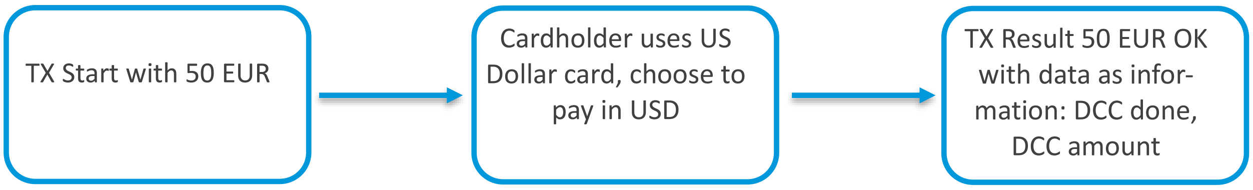
2.2 BALANCE – END OF DAY USECASE

The Balance function triggers the disbursement of the payments made to the merchant account. With the totals in the balance response, a comparison can - and should - be made with the cash register totals.
If possible, the balance function can be automated into a cash register cut or daily report function in the cash register system. This would also ensure **regular daily execution**.
In addition, it should also be possible to start the balance function manually for service purposes or if the automatic execution from ECR software is not possible or successful in individual cases.

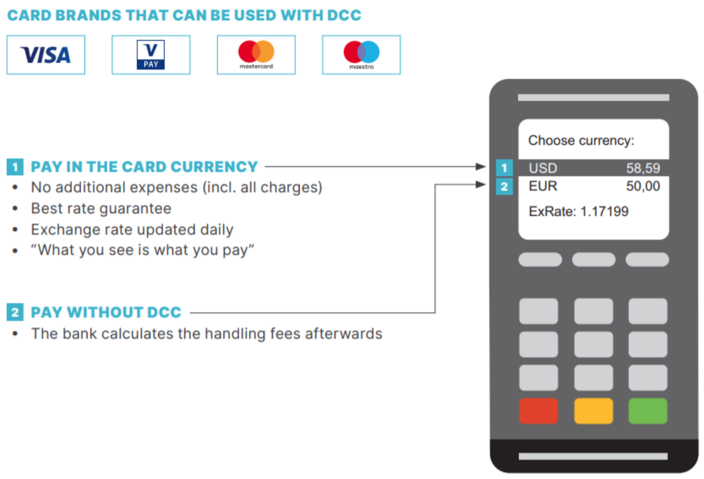
Do not parse or extract sums from receipt data as layout can change, use values from `CounterList` if needed in your software

2.3 DCC

With the Dynamic Currency Conversion (DCC) cardholders can easily pay in their home currency. The payment terminal recognises foreign cards automatically and gives the option of paying in usual card currency.
There is nothing to do on ECR side. Currency towards merchant and TIM API remains in started currency. Of course, in case of interest and report functionality the information about DCC data can be fetched from transactionInformation in response data.
Obligation for receipt data is automatically included in API receipt data, just print data 1:1 and all regulatory requirements are fulfilled.


TX Start with 50 EUR
Cardholder uses US Dollar card, choose to pay in USD
TX Result 50 EUR OK with data as information: DCC done, DCC amount

TransactionStatus will show activity with DCC in transaction flow already, eg.
if TransactionStatus reports value DccSelection.


CARD BRANDS THAT CAN BE USED WITH DCC
VISA
V PAY
mastercard
maestro
1 PAY IN THE CARD CURRENCY
No additional expenses (incl. all charges)
Best rate guarantee
Exchange rate updated daily
"What you see is what you pay"
2 PAY WITHOUT DCC
The bank calculates the handling fees afterwards
Choose currency:
1 USD 58,59
2 EUR 50,00
ExRate: 1.17199

2.4 ASYNC / SYNC INFORMATION

- **Async** (preferred): you have to work with event listeners and state requests to have a robust flow (e.g. the commit should be performed after a purchase transactionCompleted event).

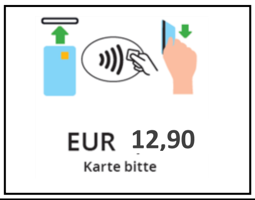
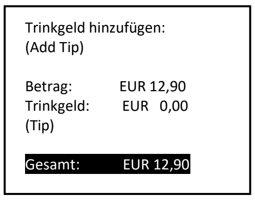
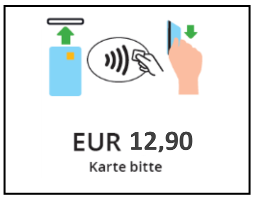
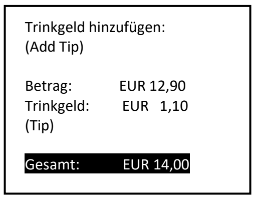
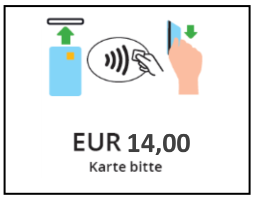
- **Sync**: this flow is sequential. An abort cannot be performed by the ECR after initiating a transaction and must be done by the EFT.

2.5 TIP HANDLING

TIM API offers a flexible usage of the TIP function on terminals. Please remeber to compare response data to determine TIP amounts in your system and reports accordingly.

Prerequisite for using TIP function on terminal:

a. TIP Option has to be enabled from SIX on terminal. If your test kit is not TIP enabled, please contact our technical partner support to change configuration settings
b. TIM Guide Gastro added in terminal object
c. TIP enabled/allowed flags in TIM API settings <u>and</u> TransactionData before sending
   - tipAllowed in terminalSettings
   - tipAllowed in TransactionData

tipAllowed	Amount	AmountTIP	Terminal-Screens		
False	12,90	N/A	 EUR 12,90 Karte bitte		
True	12,90	0	 Trinkgeld hinzufügen: (Add Tip) Betrag: EUR 12,90 Trinkgeld: EUR 0,00 (Tip) Gesamt: EUR 12,90	---> *)	 EUR 12,90 Karte bitte
True	12,90	1,10	 Trinkgeld hinzufügen: (Add Tip) Betrag: EUR 12,90 Trinkgeld: EUR 1,10 (Tip) Gesamt: EUR 14,00	----> *)	 EUR 14,00 Karte bitte

*) total or tip amount can be changed by operator/cardholder on terminal after transaction start. Effective total amount and tip amount are returned in transaction response

3 BASIC FLOW

3.1 START APPLICATION

SIX recommends **manual link control**, TIM automatisms have their limitations (eg: no reconnect in idle mode if link drops)

In start phase of ECR application please use Connect(), Login(), Activate() to establish TIM – Terminal link and enter operational shift open state for doing card transactions.

3.2 CHECK TERMINALSTATUS

TIM API reports every status change with events, please check/use status value of ConnectionStatus to detect link interruptions.

Information: Terminal will perform a self-restart every 24h, scheduled within service call in maintenance time window (22pm – 6am)

TransactionStatus can be used to inform user about transaction progress (eg. WaitforCard, PIN entry, etc) For a detailed report you can use Terminal display text information displayContent in an information window in your application. This is recommended, if a merchant can not check the display content itself (eg. protective barrier, …)

Check TransactionStatus before the start of a transaction. The Terminal may be busy for a short time if data transfer or service tasks are running. The status Idle or ApplicationSelection is recommended to start a new transaction request.

3.3 END APPLICATION

If cash register software will be closed, the link to the terminal must also be closed in a controlled manner. Otherwise the terminal link may remain busy and active.

Please use the Deactivate(), Logout(), Disconnect() command sequence to terminate the TIM terminal connection and then call the API Dispose functions.

Deactivate() is not applicable if *ManagementStatus* has value *closed* already.

In .net, an additional TIMApiDispose() is foreseen to terminate all API internal tasks on Windows platforms.

3.4 TIM STATES

After a basic login, the TIM API has 2 operating states:
* shift open: to run card-based use cases, eg. purchase, reversal, etc
* shift closed: to run administration functions, eg. balance, reconfig, etc.

actual states of terminal link are reported with terminal_status_changed event for example

TIM API Calls (App starting)	Connect() →	Login() →	Activate() →	
ConnectionStatus	**Disconnected**	**LoggedOut**	**LoggedIn**	**LoggedIn**
ManagementStatus	**Closed**	**Closed**	**Closed**	**Open**
TIM API Calls (App closing)	← Disconnect()	← Logout()	← Deactivate()	
shift status	-	-	Shift closed	Shift opened
possible commands			Balance, Administration Commands	Transactions, Commit

3.5 COMMIT HANDLING

Commit is the last message from the ECR to the terminal to finalize the transaction. The card processing has already been performed and a message of the terminal transaction authorisation has been sent to the ECR in advance. After the commit from ECR to the terminal (EFT) has been sent, the terminal signalizes the cardholder of the positive result of the transaction. If there is a printer on the terminal, the printer would now print the receipt or the ECR prints the receipt. In the setup of the ECR configuration to the terminal, there is the option to set the "autoCommit" flag.

- autoCommit=true: there is an automatic message to the terminals, that the transaction is completed by the ECR. In this case, the ECR cannot handle a possible error. There will be no waiting time for the ECR to permit the transaction completion.

- autoCommit=false: in difference to autoCommit=true, the system will wait for a specific message / method call commit() to complete the transaction. With this option, the receipt printing is saved and the control is covered by the ECR. This is the preferred setup. If no commit is sent from ECR to Terminal or being timed out, a technical Rollback of the Purchase will be executed by the Terminal!

4 RECEIPT HANDLING

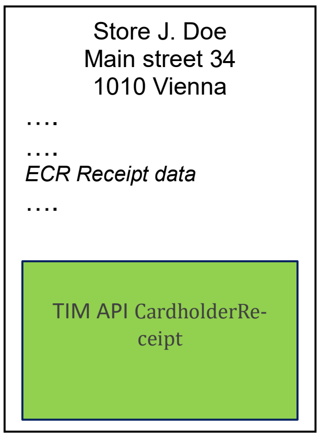
The receipt handling must be compliant with the Card Scheme Regulations. The authorization / payment receipts need to be treated carefully, so that these are available to the cardholder, merchant and acquirer. TIM API provides a 'ready-to-use' data format with CardholderReceipt and MerchantReceipt in TransactionResponse.PrintData.
Please use this print data as is. Card scheme regulations are covered automatically by terminal software. Otherwise ECR application has to take care of scheme compliant receipt data.
We recommend to integrate a print data block for customers in your standard ECR receipt. There is no need to print an own paper slip for card transaction receipt.

An ECR application can adjust preformatted text with PrintOptions:

- ➔ use TIM API PrintFlag SuppressHeader in PrintOptions to suppress header in CardholerReceipt if own header is on receipt anyway
- ➔ use PrintWidth in PrintOptions when default value does not fit with used paper width.

Check if you have the text ***** Customer Receipt ***** on top of your receipt for the cardholder (in the correct language).

Example ECR customer receipt with integrated TIM data:



The merchant must store the merchant receipts for at least 3 months (ECR or back office system). This can be in electronic form such as database, file, …., so he can provide it, if asked by the acquirer or customer.

Provide in which form the Merchant receipt is stored:

- **paper / database / file / PDF/ other …**
- Location, if electronic = **…**

Provide in which form the Cardholder receipt is issued:

- **paper / email / other …**

In which format is the receipt:

- **integrated into the store receipt / extra receipt / other …**

5 LOGGING AND DATA STORAGE

5.1 TIM LOG DATA

1. use Log Level FINEST of TIM API
2. Logfiles should be available at least for 90 days. SIX support will request in some cases those files for technical analysis of issues at customer location.
3. Logfiles should be accessible for your support via remote/service access

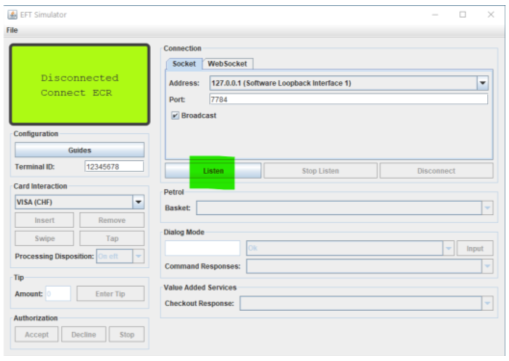
5.2 RECEIPT / TRANSACTION DATA

The TIM API MerchantReceipt data must be stored for at least 120 days on ECR systems or in back office systems. A user-friendly reprint or export function is strongly recommended. In case of chargebacks or customer complaints, a card-issuing bank may ask the merchant for a receipt copy.

6 TESTING

6.1 BASICS TEST / IMPLEMENTATION ANALYSIS

The TIM SDK includes a terminal simulator software in the directory TimSdk\Tools\EftSimulator. With it, you can develop your first lines with the TIM API and its functions.



Disconnected
Connect ECR

Address: 127.0.0.1 (Software Loopback Interface 1)
Port: 7784
☑ Broadcast
Terminal ID: 12345678
VISA (CHF)
Processing Disposition: On eft
Amount: 0
Basket:
Ok
Command Responses:
Checkout Response:

Short-intro:

- start simulation with ‘Listen’ to receive connect requests from TIM API
- connect from your application to localhost (127.0.0.1).
- do a login, activate on your ecr system
- start a purchase transaction
- in simulator you can do a card transaction with card reading
  (buttons ‘Insert’ or ‘Swipe’ or ‘Tap’ to simulate card reading)
- select authorization result with button Accept or Decline or Stop Button
- remove button if card interaction was ‘insert’

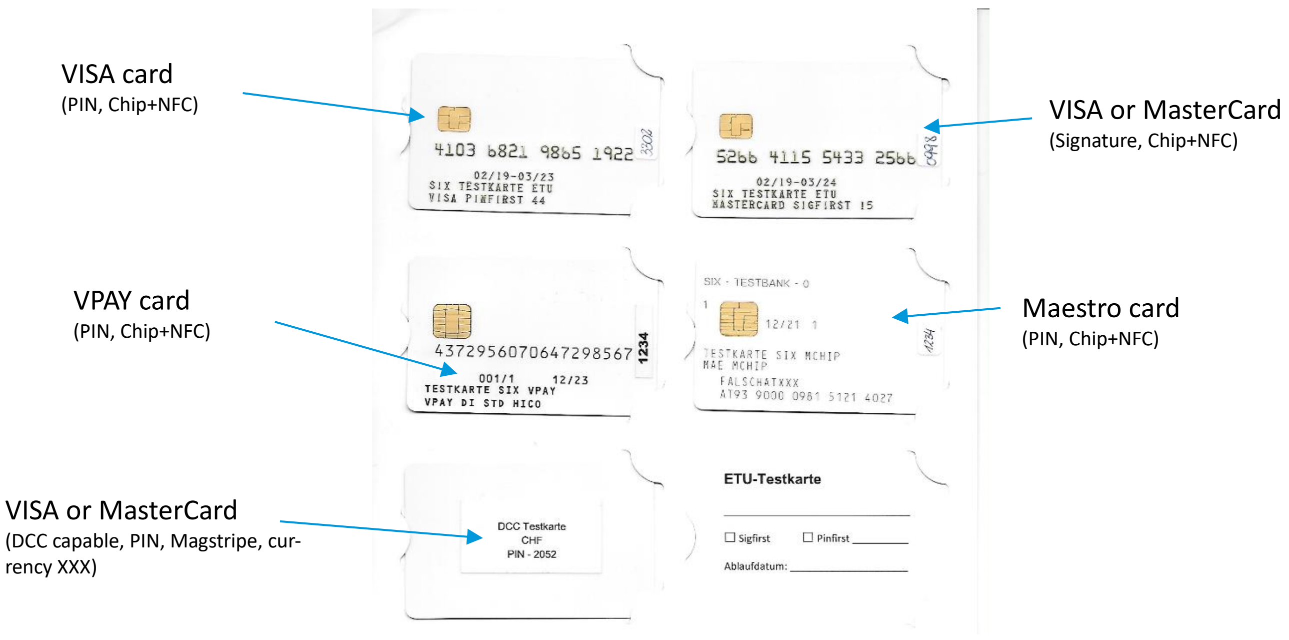
7 TEST KIT

7.1 ORDER

Please order a test kit (terminal + cards) for your integration phase. Certification test script requires a test kit.

7.2 TESTCARDS

Our Austrian testcard set comes with 5 different cards (brands, profiles, etc).



Important to know: Use testcard with label '…SigFirst' via chip card reader to test CVM Signature and a printout of MerchantReceipt in this case – first in right column.
DCC card is required to trigger a DCC transaction on terminal (brand and currency may differ in your set).

7.3 LICENSES

The TIM Software Development Kit is free of any license for the integrator.

Test cards use is unlimited, except too many false PIN entries were performed. Contact your Partnermanager to unlock a card, if this is the case.

The merchant who uses the terminal has to acquire a terminal license with his Worldline Sales Partner.

This allows the merchant to use the TIM module for a specific terminal.

8 RELEASE INFORMATION AND UPDATES

TIM API is updated regularly. Please check the Newsletter or Partner Extranet for new versions and release notes.