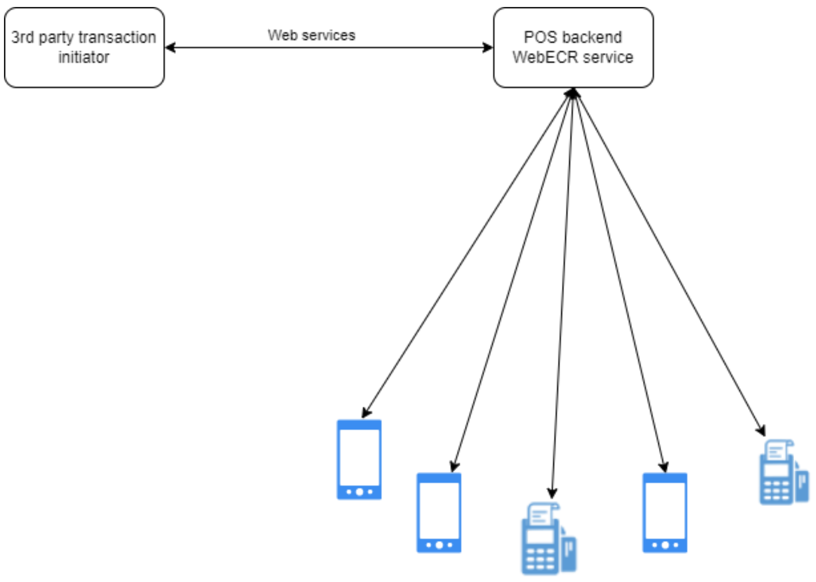
EFT POS - WebECR

Version 2.5.7 - Published 22/04/2024

1. Overview

The webECR service is a device concentrator service that provides a way for 3rd party services to initiate transactions with and receive results from managed EFT-POS devices. All device management and communications are handled by webECR, thus negating the need for manual device management.



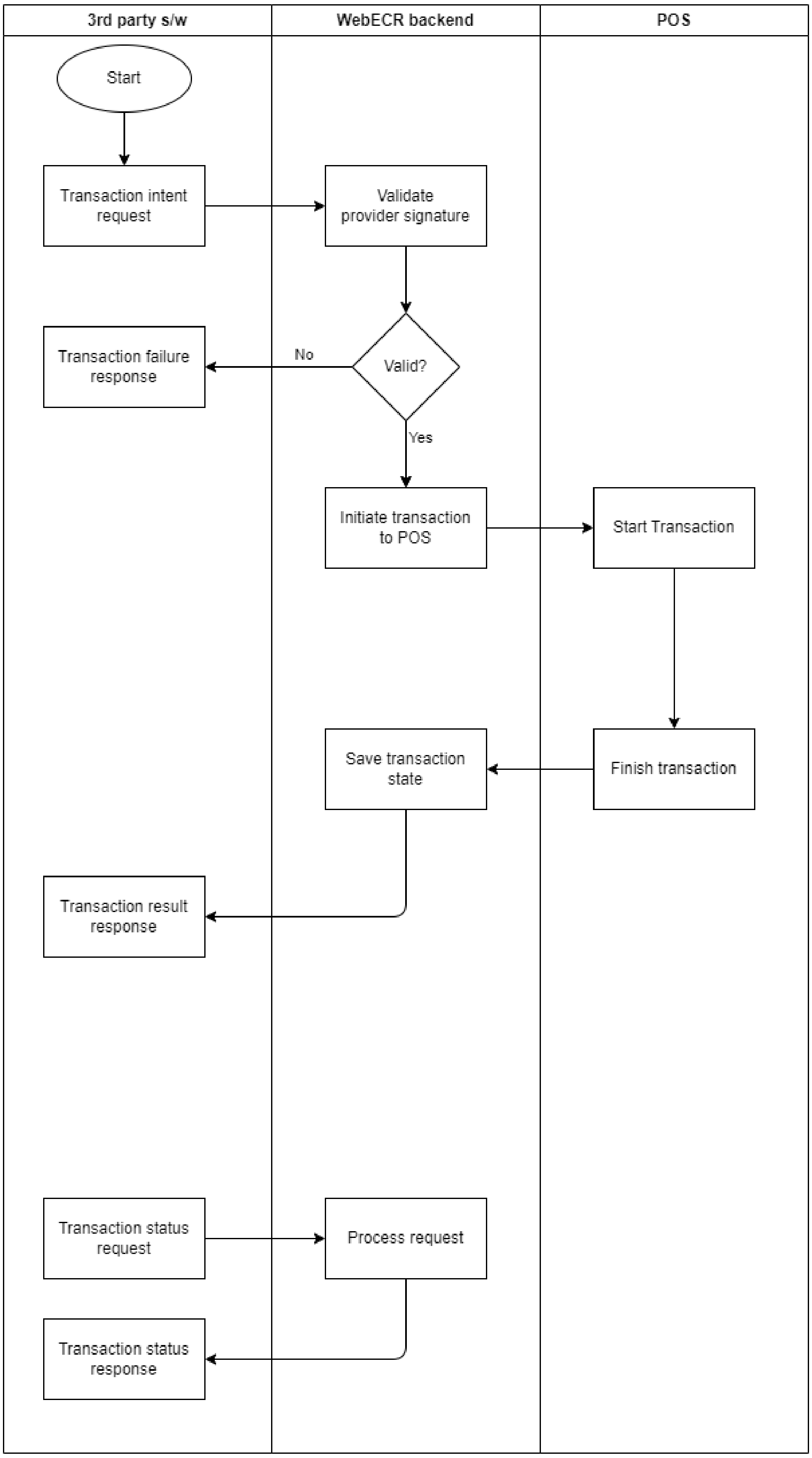
2. Workflows

2.1. API workflows:

The webECR service has 2 main workflows:

2.1.1. Polling workflow

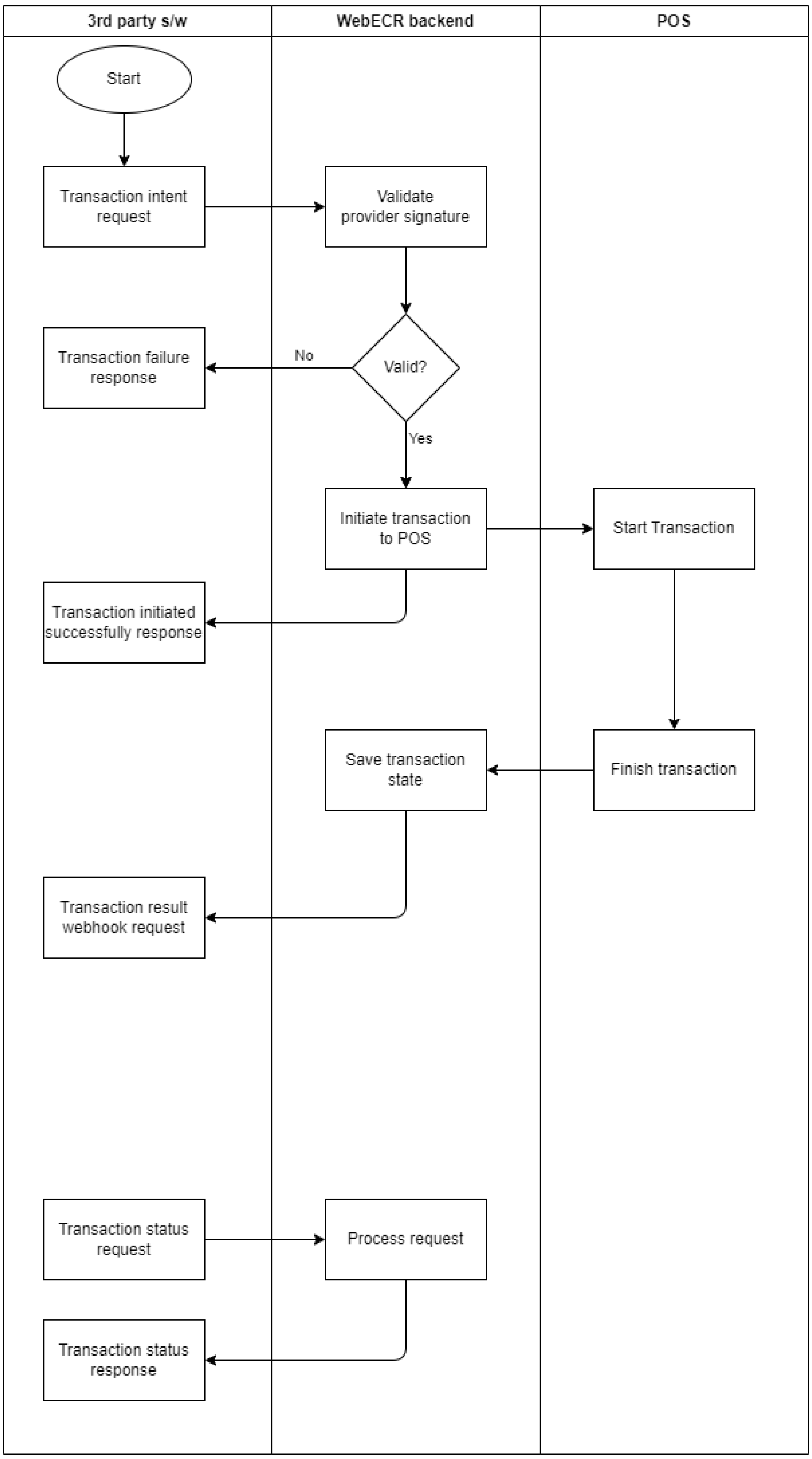
When initiating a transaction, the call is handled synchronously and the caller has the ability to wait for the transaction to complete. If after the maximum wait time, the request has not completed yet, the caller has the ability to call the results endpoint in regular intervals to check the transaction status.


3rd party s/w
WebECR backend
POS
Start
Transaction intent request
Validate provider signature
Transaction failure response
No
Valid?
Yes
Initiate transaction to POS
Start Transaction
Finish transaction
Save transaction state
Transaction result response
Transaction status request
Process request
Transaction status response

2.1.2. Webhook callback workflow

When initiating a transaction, the call is acknowledged instantly. The caller provides a callback URL (webhook) that will be called by the WebECR backend when the transaction intent changes status.

The callback URL is configured once at the account initiation process and no need to provide a callback URL on each request is required.


3rd party s/w
WebECR backend
POS
Start
Transaction intent request
Validate provider signature
Transaction failure response
No
Valid?
Yes
Initiate transaction to POS
Start Transaction
Transaction initiated successfully response
Save transaction state
Finish transaction
Transaction result webhook request
Transaction status request
Process request
Transaction status response

2.2 Authentication workflows

WebECR allows for two workflows on how a client software is authenticated and on what EFT-POS devices it is allowed to initiate transasctions.

2.2.1 Direct permissions workflow

In this workflow, the client software is directly controlled by the merchant. Thus, a merchant account is created in the WebECR server and the merchant's EFT-POS devices are directly assigned to this account.

In order to achieve this authentication, a bearer token is required, as described in chapter 3.6.1

2.2.2 3rd party servicer permissions workflow

In this workflow, the client software is not controlled by the merchant but by a 3rd party servicer (ex. SaaS invoicing solution provider). Thus the client software manages multiple merchants and has the capability to start transactions on behalf of them.

In this workflow, the client software is assigned with a client account which has no direct EFT-POS device assignments. Each EFT-POS device provides the capability for a merchant to grant or revoke permissions towards the 3rd party servicer to start transactions from the device menu.

In order to achieve this authentication, a bearer token is required (chapter 3.6.1) as well as an API key as described in chapter 3.2

3. API

All API calls are REST based and accept/respond with valid JSON. For all endpoints, both the WebECR service as well as the client have the ability to send additional fields not included in this document to allow for flexiblity in customisations.

It is therefore mandatory for all API implementations to:

- Ignore any fields in requests or responses that they do not recognise
- Not implement any mandatory fields not included in this document

The following calls are applicable to the webECR service:

3.1. Authentication

All calls require authentication & authorization. This is achieved through JWT bearer token authentication (access/refresh). The access token shall be used on all subsequent calls in the "Authorization" header as follows:

```
Authorization: Bearer <JWT access token>
```

The refresh token can be used to renew the access token without the usage of a username/password combination.

3.2. Authentication on behalf of user

In case the client software has acquired additional merchant permissions (3rd party servicer permissions workflow), these permissions are granted with the usage of an Api key. The API key needs to be included in the "X-Api-Key" header on all calls.

Example:

```
Authorization: Bearer <JWT access token>
X-Api-Key: <API key>
```

In order for the client software to retrieve the X-Api-Key, the following workflow is to be followed:

1. The user turns on the EFT-POS or opens the softPOS application on their phone
2. The user navigates to the relevant POS menu option named "Link with 3rd party services"
3. The EFT-POS shows an authentication code. The user enters it in the client software.
4. The client software exchanges (redeems) the authentication code with the API key, using the authentication code redeem API call as described below
5. The client software saves the API key and uses it for the rest of the integration lifetime

The client software can save/cache the API key and use it to start transactions at any point in time. The merchant has the ability to revoke the access permissions at any time, so the Api key may become invalid. In this case, the client software should handle this as an error condition.

3.3. ECR Token based data validation

In case the client software is using a local device for signature generation (ΦΗΜΑΣ) instead of an electronic invoicing provider, an alternative workflow needs to be used in order to adapt to the capabilities and mode of operation of these devices.

1. The calling software needs to call the terminal bind endpoint (section 3.4.9) in order for the POS to be registered as connected to the ECR device. This needs to be called at least once.
2. The calling software needs to call the terminal key exchange (section 3.4.10) in order to exchange session keys with the ECR device. This needs to be called at least once, and is strongly recommended to be called periodically after a predefined number of transactions.
3. The calling sofware is able to start transactions normally, using the start new transaction intent endpoint and filling the EcrTokenData data structure.

This workflow is also described in the document "A1155 - ERP to EFTPOS protocol proposal" which is the socket based protocol variant.

3.4. Pagination

Some calls that typically return multiple results are paginated. The pagination scheme is common for all paginated responses and is as follows:

parameter	type	required	comment
count	integer	yes	the total results count
next	string	no	url that should be called to retrieve the next page
previous	string	no	url that should be called to retrieve the previous page
results	list	yes	the list that contains the actual results

Example paginated response

```
{
    "count": 10816,
    "next": "https://uat.mreceipts.com/api/.../?limit=100&offset=100",
    "previous": null,
    "results": [
        {
            ...
        },
        {
            ...
        },
        {
            ...
        }
    ]
}
```

3.5. Filtering

The calls that return multiple results support filtering of results. The filtering works as GET query parameters as in the following format:

Example GET filtering which searches for an terminal entity with TerminalID = 01234567 AND Acquirer = 11

```
GET https://[baseURL]/?TerminalID=01234567&Acquirer=11
```

3.6. Error handling

WebECR responds with these standard HTTP response codes in case of error:

- **200**: Indicates a successful execution of the request.
- **400**: Indicates that the request was invalid. Where applicable, the respose contains details of the error
- **401**: Indicates that the caller has not authenticated successfully
- **403**: Indicates that the caller has authenticated successfully but has insufficient permissions to access the specific endpoint or call.
- **404**: Indicates that an API call endpoint is invalid. In case the endpoint contains an entity id, this entity does not exist or is inaccessible to the caller.
- **500**: Indicates that there was a server error

3.7. API

3.7.1 Get access token

Request type: POST
Request URL: [base url]/token/

Request parameters:

parameter	type	required	comment
username	string	yes	
password	string	yes	

Response parameters:

parameter	type	required	comment
access	string	yes	The access token in JWT form
refresh	string	yes	The refresh token in JWT form

Example request:

```
{
    "username" : "test_username",
    "password" : "test_password"
}
```

Example response:

```
{
    "refresh":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTY3
OTA2MTQwMSwianRpIjoiOWNkZDE4YWQwZjllNGE4OGIxOWE4ZGRkNzRkYmYyODAiLCJ1c2VyX2lkIjoxfQ
.B7HJ0c5EAJ_E4T0jauSDADQTNIoa6jbXRRuMC9tsit0",
    "access":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjc2
NDk4MjAxLCJqdGkiOiI3NmUyNjliYzZkN2Y0NWRhODgwZDIwNTQwYWQwNTBkMyIsInVzZXJfaWQiOjF9.S
bEq_oLk8Jwxq2Mg4v1NyVFW3gT1jdbDqrbnoQ47Q3c"
}
```

3.7.2 Refresh access token

Request type: POST
Request URL: [base url]/token/refresh/

Request parameters:

parameter	type	required	comment
refresh	string	yes	The previously acquired refresh token

Response parameters:

parameter	type	required	comment

parameter	type	required	comment
access	string	yes	The access token in JWT form

Example request:

```
{
    "refresh" :
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTY3
OTA2MTQwMSwianRpIjoiOWNkZDE4YWQwZjllNGE4OGIxOWE4ZGRkNzRkYmYyODAiLCJ1c2VyX2lkIjoxfQ
.B7HJ0c5EAJ_E4T0jauSDADQTNIoa6jbXRRuMC9tsit0"
}
```

Example response:

```
{
    "access":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjc2
NDk4MjAxLCJqdGkiOiI3NmUyNjliYzZkN2Y0NWRhODgwZDIwNTQwYWQwNTBkMyIsInVzZXJfaWQiOjF9.S
bEq_oLk8Jwxq2Mg4v1NyVFW3gT1jdbDqrbnoQ47Q3c"
}
```

3.7.3 Redeem authentication code

The endpoint is used to redeem an authentication code that is presented by a POS in order to receive an API Key. The implementation of this endpoint is required only when implementing the 3rd party servicer permissions workflow as described in sections 2.2.2 and 3.2.

Request type: POST
Request URL: [base url]/authorization/redeem/

Request parameters:

parameter	type	required	comment
Type	string	yes	The type of authorization code. Always set to "webecr"
Code	string	yes	The authorization code as shown by the POS and entered by the user

Response parameters:

parameter	type	required	comment
Type	string	yes	The type of authorization code. Always set to "webecr"
Id	string	yes	the API key that should be used for all subsequent API calls for this user

Example request:

```
{
    "Type" : "webecr",
    "Code" : "8413239096"
}
```

Example response:

```
{
    "Id": "Zg6xxfx_R1eXTOs39BPzqQ",
    "Type": "webecr"
}
```

3.7.4 Terminals list

The endpoint is used to retrieve a list of all the terminals that are available to the user. It can be used as a first step to search for a specific terminal by its name, merchant or type and retrieve its id for further processing

Request type: GET
Request URL: [base url]/terminal/

Response parameters:

parameter	type	required	comment
id	string	yes	The unique identifier of the terminal for the POS backend
TerminalID	string	yes	The terminal identifier
Merchant	string	yes	Reference to the unique identifier of the merchant of the POS
Acquirer	string	yes	Reference to the unique identifier of the acquirer of the POS

Filter parameters:

parameter	comment
TerminalID	
Merchant	
Acquirer	

Example response

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": "358046",
            "TerminalID": "64999964",
            "Merchant": "11719",
            "Acquirer": "11"
        }
    ]
}
```

3.7.5 Start new transaction intent

The endpoint is used to start a new transaction by creating an intent.

Request type: POST
Request URL: [base url]/terminal/{id}/txninit/

Request parameters:

parameter	type	required	comment
TxnType	integer	yes	The transaction type. See transaction type table below for possible values
Amount	integer	yes	The payable transaction amount in integer form. Ex. 1.00 eur is 100
TipAmount	integer	no	The tip amount in integer form. Ex. 1.00 eur is 100
CashbackAmount	integer	no	The cashback amount in integer form. Ex. 1.00 eur is 100
CurrencyCode	integer	yes	The ISO 4217 numerical currency code . 978 for EUR
Instalments	integer	no	The number of instalments
IsTaxFree	boolean	no	Is the transaction eligible to tax-free regulation
PreloadTransaction	boolean	no	The transaction should be preloaded to the POS device for asynchronous execution. Used for cases such as restaurants, deliveries, coffee shops etc.
PreloadExpiration	integer	no	The expiration of a transaction that is preloaded to a POS, in minutes. The field is mandatory if PreloadTransaction is true
OnBehalfCollection	boolean	no	The payee collects the transaction funds on behalf of another entity, ex. courier payment on delivery, food delivery platforms
CustomerReference	string	yes	A caller defined reference which can be used to reference the intent and eventually the transaction. Maximum length of 50 characters

parameter	type	required	comment
CustomerEmail	string	no	If provided and supported by the acquirer, a transaction receipt will be sent to this email address
CustomerPhone	string	no	If provided and supported by the acquirer, a transaction receipt will be sent to this phone
InitialTransaction	string	no	Required for pre-auth completion and refund transactions and it should include the TransactionId field of the original transaction
ProviderData	ProviderData	no	A data object representing the relevant data required by Greek law to accompany a provider signature based transaction request.
EcrTokenData	EcrTokenData	no	A data object representing the relevant data required by Greek law to accompany an ECR token MAC based transaction request.
Timeout	integer	no	If timeout is 0 or not present, the transaction will be initiated asynchronously. If present, the service will wait up to "Timeout" seconds before returning to the caller. Max timeout is 180s

Response parameters:

parameter	type	required	comment
Id	string	yes	The unique identifier of the intent for the POS backend
Status	integer	yes	The intent status. See below for values
Result	integer	yes	The intent result. See below for values
TxnType	integer	yes	The transaction type. See transaction type table below for possible values
Amount	integer	yes	The payable transaction amount in integer form. Ex. 1.00 eur is 100
TipAmount	integer	yes	The tip amount in integer form. Ex. 1.00 eur is 100
CurrencyCode	integer	yes	The ISO 4217 numerical currency code. 978 for EUR
Instalments	integer	no	The number of instalments
CustomerReference	string	yes	The caller defined reference which can be used to reference the intent and eventually the transaction. Maximum length of 50 characters
Terminal	string	yes	Reference to the terminal which this intent was sent to
CustomerEmail	string	no	Email that a receipt was/is going to be sent

parameter	type	required	comment
CustomerPhone	string	no	Phone that a receipt was/is going to be sent
Transaction	transaction	no	Contains the transaction result details as described in section 3.4.8 - only applicable if result = APPROVED
InitialTransaction	transaction	no	Contains the initial transaction details as described in section 3.4.8 - only applicable if TxnType is refund or pre-authorization completion
TransactionId	string	no	The unique transaction id required in the Greek market as mandated and defined by law.

Transaction type

The possible values for the transaction type are:

- 0: sale
- 1: refund
- 2: Pre-authorisation
- 3: Pre-authorisation completion
- 4: Mail order/Telephone order
- 5: Cash advance
- 6: Card payment
- 7: Bill payment
- 8: Other payment
- 9: Pre-payment

Transaction types from 0 to 49 are reserved for future use. Values over 50 are allowed to be used in custom implementations.

Intent status

The possible values for the intent status are:

- 1: PENDING - Intent has been registered to the backend and is pending to be sent to the device
- 2: SENT - Intent has been sent to the device
- 3: COMPLETED - Intent has been successfully completed by the device and has registered the results

Intent result

The possible values for the intent result are:

- 1: APPROVED - The transaction has been completed and approved by the authorization system
- 2: DECLINED - The transaction has been completed and declined by the authorization system
- 3: CANCELLED - The transaction has been cancelled by the POS user before reaching completion
- 4: FAILED - The transaciton has failed to complete
- 5: UNKNOWN - The transaction result is unknown. Only possible if the device hasn't responded with results

- 6: BUSY - The transaction has failed because the POS is currently unavailable for transactions (either processing another transaction or under maintenance)
- 7: MAX_TRANSACTIONS - The POS device has reached its transaction limit for the specific batch. Batch closing should be performed on the device before continuing transactions

Provider Data

The provider data is an object that contains the provider signature and the information needed to validate the signature as defined by the relevant Greek law. It contains the following information

parameter	type	required	comment
Uid	string	yes	The UID of the invoice
Mark	string	no	
SignatureTimestamp	string	yes	The generation timestamp of ProviderSignature in the same format as in the signature itself, namely YYYYMMDDhhmmss in Greece local time
NetAmount	string	yes	The price of goods without any VAT applied to it
VatAmount	string	yes	The amount of VAT that is applied to the net amount
TotalAmount	string	yes	The amount of VAT that is applied to the net amount
ProviderId	integer	yes	The id of the electronic invoicing provider. It is required for public key lookup.
Signature	string	yes	The electronic invoicing provider signature. Required when a provider signature validation process is used.

The WebECR server uses the data from the following fields in order to perform the validation:

- ProviderData.Uid
- ProviderData.Mark
- ProviderData.SignatureTimestamp
- Amount
- ProviderData.NetAmount
- ProviderData.VatAmount
- ProviderData.TotalAmount
- TID (not included in request but referenced in the URL)

ECR Token Data

The ECR Token data is an object providing the same functionality as provider data and used in cases where symmetric cryptography is to be used for transaction validation. It contains the following information

parameter	type	required	comment
AmountCommand	string	yes	The AMOUNT command as received and signed by a compliant device according to Greek law

In order to provide parity with socket based implementations, in the case of validation via ECR token a valid AMOUNT command is expected, as defined in the document "A1155 - ERP to EFTPOS protocol proposal". The command is checked for protocol level validity, for data correctness against the main data structure of the /txninit endpoint, as well as for cryptographic validity.

In order for ECR token data transaction initiation to succeed, the caller must first complete the process as described

Examples

Example request:

The following request is a simple request that does not include signature validation. For merchants & transaction types that require validation, this request will fail with an HTTP 400.

```
{
    "TxnType" : 0,
    "Amount" : 100,
    "TipAmount" : 0,
    "CurrencyCode" : 978,
    "CustomerReference" : "b7775c92-0be9-4005-9450-e45769e593e2",
    "Timeout" : 180
}
```

Example response:

```
{
    "Id": "1060",
    "Status": 3,
    "Result": 1,
    "TxnType": 0,
    "Amount": 100,
    "TipAmount": 0,
    "CurrencyCode": 978,
    "Instalments": 0,
    "CustomerReference": "b7775c92-0be9-4005-9450-e45769e593e2",
    "Terminal": "358154",
    "CustomerEmail": null,
    "CustomerPhone": null,
    "Transaction": {
        "Id": "8091750",
        "TxnType": 0,
        "Timestamp": "2023-11-13T10:45:24Z",
        "VoidTimestamp": null,
        "CardPAN": "516732******6411",
        "Approved": true,
        "Voided": false,
        "STAN": 1,
        "BatchNumber": 1,
        "Acquirer": "75",
        "TID": "90000001",
        "MID": "9000000001",
        "Amount": 100,
        "TipAmount": 0,
        "CashbackAmount": 0,
        "LoyaltyRedemptionAmount": 0,
        "Instalments": 1,
        "RRN": "001003",
        "AuthCode": "104525",
        "OriginalRRN": null,
        "OriginalAuthCode": null,
        "CurrencyCode": 978,
        "CustomerReference": "b7775c92-0be9-4005-9450-e45769e593e2"
    },
    "InitialTransaction": null
}
```

Example request (Greek market with Provider Data validation):

```
{
    "TxnType" : 0,
    "Amount" : 100,
    "TipAmount" : 0,
    "CurrencyCode" : 978,
    "CustomerReference" : "b7775c92-0be9-4005-9450-e45769e593e2",
    "Timeout" : 180,
    "ProviderData" : {
        "Uid" : "D4F6A5F5C6123658F78369E5191ED5C9D73CB7AC",
        "Mark" : null,
        "SignatureTimestamp" : "20231114100000",
        "NetAmount" : 100,
        "VatAmount" : 24,
        "TotalAmount" 124,
        "ProviderId" : "012",
        "ProviderSignature" : "12345"
    }
}
```

Example response (Greek market with Provider Data validation):

```
{
    "Id": 1060,
    "Status": 3,
    "Result": 1,
    "TxnType": 0,
    "Amount": 100,
    "TipAmount": 0,
    "CurrencyCode": 978,
    "Instalments": 0,
    "CustomerReference": "b7775c92-0be9-4005-9450-e45769e593e2",
    "Terminal": "358154",
    "CustomerEmail": null,
    "CustomerPhone": null,
    "Transaction": {
        "Id": "8091750",
        "TxnType": 0,
        "Timestamp": "2023-11-13T10:45:24Z",
        "VoidTimestamp": null,
        "CardPAN": "516732******6411",
        "Approved": true,
        "Voided": false,
        "STAN": 1,
        "BatchNumber": 1,
        "Acquirer": "75",
        "TID": "90000001",
        "MID": "9000000001",
        "Amount": 100,
        "TipAmount": 0,
        "CashbackAmount": 0,
        "LoyaltyRedemptionAmount": 0,
        "Instalments": 1,
        "RRN": "121702285176",
        "AuthCode": "315144",
        "OriginalRRN": null,
        "OriginalAuthCode": null,
        "CurrencyCode": 978,
        "CustomerReference": "b7775c92-0be9-4005-9450-e45769e593e2"
    },
    "InitialTransaction": null
    "TransactionId" : "075121702285176315144"
}
```

Example request (Greek market with Ecr token data validation):

```
{
    "TxnType" : 0,
    "Amount" : 100,
    "TipAmount" : 0,
    "CurrencyCode" : 978,
    "CustomerReference" : "b7775c92-0be9-4005-9450-e45769e593e2",
    "Timeout" : 180,
    "EcrTokenData" : {
        "AmountCommand" :
"ECR0110A/S000011/F100:978:2/RABC00111222/D20240412124019/M0/H0/T11/QD93EDE48"
    }
}
```

Example response (Greek market with Ecr token data validation):

```
{
    "Id": 1060,
    "Status": 3,
    "Result": 1,
    "TxnType": 0,
    "Amount": 100,
    "TipAmount": 0,
    "CurrencyCode": 978,
    "Instalments": 0,
    "CustomerReference": "b7775c92-0be9-4005-9450-e45769e593e2",
    "Terminal": "358154",
    "CustomerEmail": null,
    "CustomerPhone": null,
    "Transaction": {
        "Id": "8091750",
        "TxnType": 0,
        "Timestamp": "2023-11-13T10:45:24Z",
        "VoidTimestamp": null,
        "CardPAN": "516732******6411",
        "Approved": true,
        "Voided": false,
        "STAN": 1,
        "BatchNumber": 1,
        "Acquirer": "75",
        "TID": "90000001",
        "MID": "9000000001",
        "Amount": 100,
        "TipAmount": 0,
        "CashbackAmount": 0,
        "LoyaltyRedemptionAmount": 0,
        "Instalments": 1,
        "RRN": "121702285176",
        "AuthCode": "315144",
        "OriginalRRN": null,
        "OriginalAuthCode": null,
        "CurrencyCode": 978,
        "CustomerReference": "b7775c92-0be9-4005-9450-e45769e593e2"
    },
    "InitialTransaction": null
    "TransactionId" : "075121702285176315144"
}
```

3.7.6 Transaction Void Intent

The following endpoint is used to start a new intent to void a transaction. The caller needs to reference the transaction that needs to be voided and has the ability to either reference it via the original transaction intent, or the tranasction id. Both values can be obtained either by the response of the original successful intent, or by querying the the intent or transaction endpoints.

When a transaction is voided, the original intent and the void intent are 2 separate entities, however the transaction is object remains one and has a single transaction id.

Request type: POST
Request URL: [base url]/terminal/{id}/txnvoid/

Request parameters:

parameter	type	required	comment
OriginalIdentifier	string	yes	The identifier of the original intent
OriginalIdentifierType	integer	yes	The type of identifier contained in the OriginalIdentifier field
CustomerEmail	string	no	If provided and supported by the acquirer, a transaction receipt will be sent to this email address
CustomerPhone	string	no	If provided and supported by the acquirer, a transaction receipt will be sent to this phone

Response parameters:

parameter	type	required	comment
id	string	yes	The unique identifier of the intent for the POS backend
Status	integer	yes	The intent status.
Result	integer	yes	The intent result.
Terminal	string	yes	Reference to the terminal which this intent was sent to
CustomerEmail	string	no	Email that a receipt was/is going to be sent
CustomerPhone	string	no	Phone that a receipt was/is going to be sent
Transaction	transaction	no	Contains the transaction result details as described in section 3.4.8 - only applicable if result = APPROVED
InitialTransaction	transaction	no	Contains the initial transaction details as described in section 3.4.8 - only applicable if TxnType = refund or pre-authorization completion

OriginalIdentifierType

For quick reference of the original transaction, the API allows for multiple ways to identify the original transaction, depending on the OriginalIdentifierType. Possible values:

- 1: INTENTID - The field contains the field "Id" of the original intent to be voided
- 2: TRANSACTIONID - The field contains the field "TransactionId" of the original intent to be voided.

3.7.7 Transaction intent list

The endpoint is used to list all transaction intents that have been created and accessible to the user. It should be used if the "start transaction intent" call has timed out or was async in order to resolve the final status of a specific intent

Request type: GET
Request URL: [base url]/transactionintent/

Response parameters:

parameter	**type**	**required**	**comment**
id	string	yes	The unique identifier of the intent for the POS backend
Status	integer	yes	The intent status
Result	integer	yes	The intent result
TxnType	integer	yes	The transaction type. See transaction type table for possible values
Void	boolean	yes	Is this a transaction intent or a transaction void intent?
Amount	integer	yes	The transaction amount in integer form. Ex. 1.00 eur is 100
TipAmount	intege	yes	The tip amount in integer form. Ex. 1.00 eur is 100
CurrencyCode	integer	yes	The ISO 4217 numerical currency code. 978 for EUR
Instalments	integer	no	The number of instalments
CustomerReference	string	yes	The caller defined reference which can be used to reference the intent and eventually the transaction. Maximum length of 50 characters
Terminal	string	yes	Reference to the terminal which this intent was sent to
CustomerEmail	string	yes	Email that a receipt was/is going to be sent
CustomerPhone	string	yes	Phone that a receipt was/is going to be sent
Transaction	string	no	Reference to the transaction details of the completed transaction - only applicable if result = APPROVED
InitialTransaction	string	no	Reference to the transaction details of the initial transaction - only applicable if transaction type is refund or pre-auth completion
TransactionId	string	no	The unique transaction id required in the Greek market as mandated and defined by law.

Filter parameters:

parameter	**comment**
Status	
Result	
TxnType	
Timestamp_min	

parameter	comment
Timestamp_max	
Amount	
Amount_min	
Amount_max	
TipAmount	
TipAmount_min	
TipAmount_max	
Instalments	
CustomerReference	
Terminal	
TID	
Transaction	
InitialTransaction	

Response example

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": "5",
            "Status": 3,
            "Result": 1,
            "TxnType": 0,
            "Amount": 1000,
            "TipAmount": 0,
            "CurrencyCode": 978,
            "Instalments": 0,
            "CustomerReference": "e95d0038-6e5f-4874-84fe-e102ee6b87ca",
            "Terminal": "14833",
            "CustomerEmail": null,
            "CustomerPhone": null,
            "Transaction": null,
            "InitialTransaction": null,
            "TransactionId": null
        }
    ]
}
```

3.7.8. Transaction list

The endpoint is used to list all transactions that have been successfully performed and accessible to the user. It is referenced by the "Transaction" and "InitialTransaction" fields of a transaction intent and can be used to provide more details about the transaction.

Request type: GET
Request URL: [base url]/transaction/

Response parameters:

parameter	type	required	comment
Id	string	yes	The unique identifier of the transaction for the POS backend
TxnType	integer	yes	The transaction type. See transaction type table for possible values
Timestamp	string	yes	The transaction completion timestamp in iso-8601 format
VoidTimestamp	string	no	The transaction void completion timestamp in iso-8601 format
CardPAN	string	no	The truncated PAN of the card
Approved	boolean	yes	True if transaction is Approved, false otherwise
Voided	boolean	yes	True if transaction is Voided, false otherwise
STAN	integer	yes	The System trace audit number of the transaction as set by the authorisation system
BatchNumber	integer	yes	The number of the batch this transaction belongs to
TID	string	yes	Terminal ID of the terminal that performed the transaction
MID	string	yes	Merchant ID of the terminal that performed the transaction
Amount	integer	yes	The actual transaction amount in integer form. Ex. 1.00 eur is 100. Does not include any tip or cashback
TipAmount	integer	yes	The transaction tip amount in integer form. Ex. 1.00 eur is 100
CashbackAmount	integer	yes	The transaction cashback amount in integer form. Ex. 1.00 eur is 100
LoyaltyRedemptionAmount	integer	yes	The transaction loyalty redemption amount in integer form. Ex. 1.00 eur is 100
Instalments	integer	no	The number of instalments

parameter	type	required	comment
RRN	string	yes	The retrieval reference number of the authorization system
AuthCode	string	yes	The authorization code of the authorization system
OriginalRRN	string	no	The RRN of the initial transaction - only applicable if TxnType = REFUND
OriginalAuthCode	string	no	The authorization code of the initial transaction - only applicable if TxnType = REFUND
CurrencyCode	integer	yes	The ISO 4217 numerical currency code . 978 for EUR
CustomerReference	string	no	The customer defined reference used to reference the transaction in 3rd party systems. Maximum length of 50 characters

The total (payable) amount of a transaction is calculated as follows:

Amount + TipAmount + CashbackAmount + LoyaltyRedemptionAmount

The amount charged to the card does not include the loyalty amount which is redeemed from a loyalty system:

Amount + TipAmount + CashbackAmount

Filter parameters:

parameter	comment
TxnType	
Timestamp_min	
Timestamp_max	
Timestamp_max	
Amount	
Amount_min	
Amount_max	
TipAmount	
TipAmount_min	
TipAmount_max	
CashbackAmount	
CashbackAmountt_min	
CashbackAmount_max	

parameter	comment
Acquirer	
Terminal	
TID	
Merchant	
MID	
RRN	
AuthCode	
RRN	
OriginalRRN	
OriginalAuthCode	
CustomerReference	

Example response:

```
{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "Id": "8083099",
            "TxnType": 0,
            "Timestamp": "2023-02-15T15:35:16Z",
            "VoidTimestamp": null,
            "CardPAN": "491791******7889",
            "Approved": true,
            "Voided": false,
            "STAN": 13,
            "BatchNumber": 143,
            "Acquirer": 11,
            "TID": "64999942",
            "MID": "1234814",
            "Amount": 200,
            "TipAmount": 0,
            "CashbackAmount": 0,
            "LoyaltyRedemptionAmount": 0,
            "Instalments": 0,
            "RRN": "304610013134",
            "AuthCode": "892148",
            "OriginalRRN": null,
            "OriginalAuthCode": null,
            "CurrencyCode": 978,
            "CustomerReference": null
        }
    ]
}
```

3.7.9. ECR - Terminal bind

This call is required only when EcrTokenData workflow is used (section 3.3).

Request type: POST
Request URL: [base url]/terminal/{id}/ecrbind/

Request parameters:

parameter	type	required	comment
EcrId	string	yes	The unique ECR id of the device to be bound

Response parameters:

parameter	type	required	comment
EcrId	string	yes	The unique ECR id as reflected from the request

If a terminal is already bound and the endpoint is called with the same EcrId, the call has no effect. If the endpoint is called with a different EcrId, the previous EcrId is discarded and the device is bound with the new one.

Example request:

```
{
    "EcrId" : "ABC00111222"
}
```

Example response:

```
{
    "EcrId": "ABC00111222"
}
```

3.7.10. ECR - Terminal session key exchange

This call is required only when EcrTokenData workflow is used (section 3.3)

The session key provided must always be encrypted under the pre-shared master key that the ECR and POS has exchanged with ESEND.

Request type: POST
Request URL: [base url]/terminal/{id}/ecrkeyexchange/

Request parameters:

parameter	type	required	comment
EcrId	string	yes	The unique ECR id of the device that was bound with the POS
SessionKey	string	yes	The new sesssion key to be used for all future authentication. The previous session key (if it exists) is deleted
Kcv	string	yes	The key check value generated with this key. Used to validate the correctness of the session key

Response parameters:

parameter	type	required	comment
EcrId	string	yes	The unique ECR id of the device that was bound with the POS- reflected from the request
SessionKey	string	yes	The session key that was loaded - reflected from the request
Kcv	string	yes	The key check value of the key that was loaded - reflected from the request

The following example assumes that the pre-shared master key is "ABCDEF01234567899876543210ABCDEF" and the cleartext session key that is transmitted is "7E0269C147A7C328F46E497841CD5823"

Example request:

```
{
    "EcrId" : "ABC00111222",
    "SessionKey" : "8FF79564430DD13EF06B5F199278FADC",
    "Kcv" : "4C8166"
}
```

Example response:

```
{
    "EcrId" : "ABC00111222",
    "SessionKey" : "8FF79564430DD13EF06B5F199278FADC",
    "Kcv" : "4C8166"
}
```