Direct Debit Integration API

Application and Web Interfaces

Version 3.1.5

Release (4)

eDebit Pty Ltd

Support and Development Office

P O Box 164, Ulverstone, TAS 7315.

Tel 03 6425 9600

Version History

3.1.0 05/04/2012. Provisional document

3.1.1 10/05/2012. Draft Release Document

Adjusted for use of HTTPS POST for all functions – RM – Suitable for release for initial development consideration

3.1.2 27/05/2012 Initial Release

Modification to requirements for edPI, addition to edTStat example, modification to edPW as user name and password combination changed. Minor changes to edKI following removal of user name and password combination from edPW. Corresponding changes to example information.

3.1.3 19/06/2012 Initial Release (2)

Addition of edTRBulk to allow for the uploading of bulk transaction records for debit.

3.1.4 10/08/2012 Initial Release (3)

Modification to edTStat to allow for failure/dishonour reason.

3.1.5 15/05/2015 Release (4)

Addition of edTrAmend to allow for schedule based payment amendment.

Introduction

The eDebit Direct Debit Integration API has been developed to facilitate direct debit integration for software developers/vendors and web developers from differing industries and market segments. It should be noted that the API is not intended to process real time payments (to act as a payment gateway). This API is for the purpose of debiting funds from a nominated bank account/card on a regular basis.

The amount to be debited can vary from period to period and may be controlled by the integrated software. Where the amount does not vary then it may be useful to utilise the Schedule function detailed in this API as this instructs our process to deduct a fixed amount for either a fixed period of time and then to discontinue debiting or until further notice.

The eDebit Direct Debit Service may only be used to facilitate debits in Australia and New Zealand. International rules and regulations with regards to debits from accounts and cards stipulate that the business for whom we conduct the debit (the end user of the integrated software) must be conducting their business in Australia or New Zealand. The API CANNOT be used to undertake a debit from a US issued card for a US based business, the API CAN be used to debit a US issued card for an Australian/NZ based business. Bank accounts held by Australian or New Zealand banks are the only bank accounts that may be debited.

This document has been produced to provide software vendors and developers with the required information to enable them to integrate direct debit functionality into their software/web application. It should be noted that this document supersedes all prior versions. This version of the API introduces downstream compliance with PCIDSS requirements. Please refer to the section on PCIDSS Compliance below.

The API is subject to change and amendment and development teams are advised to obtain the latest copy of the API and to discuss projected amendments with eDebit prior to commencing development.

The API consists of a series of HTTPS calls that can be used to manage information in the eDebit Direct Debit System and to instruct our processes to deduct funds from nominated accounts/cards. Version 3 of the API introduces a tokenisation process to store confidential bank and card information. This function has been added in to provide compliance with PCI DSS.

Return values, where noted in this document will be a string value or key value pair/s using standard formatting, namely spaces are replaced with + and key value pairs are separated by &.

Prerequisites

Developers and software vendors are advised of the following prerequisites;

1. The business using the integrated application must have registered with eDebit and must have been provided with an account and eDebit number. The account number we provide is unique to the business and is required to be made accessible to the application when calling functions. Whether this is stored in a database or a configuration file is at the discretion of the software vendor.
2. The integrated software must have on demand access to the web and have the capacity to send HTTPS POST requests using SSL(3)
3. The development environment must provide for a 'web browser control' that can display a URL that will be provided.
4. Where paperless sign up and authorisation is required then the integrated software must store or gather information with regards to the drivers licence of the account/card holder and this must be included in the 'passed in' values to the relevant API call. There is no requirement for the integrated software to store these values.
5. Provision for a unique account holder reference number, this may be a reference, membership or internal account number. However it must be unique and cannot be reused at any time. Reuse will overwrite all existing information.
6. Provision for a unique debit reference that identifies each debit for each account holder. This may be the primary key in a table for example. This will be used by the eDebit system to provide return path information on the debit status and permit the software to access daily updates on the progress of a debit.
7. The integrated software must be capable of temporarily storing a token (up to 20 character string). This token value must be passed in when sending transactions or account information.
8. The integrated software must be able to determine the type of account to be debited, card or bank, prior to registration of the account. This does not need to be stored locally.
9. Where the schedule function is not utilised then the integrated software must provide the data instructing us to debit a given client on the relevant date.
10. Please be advised that the usage of the Schedule function is end user dependent. An end user of the integrated software may choose to use or not the Schedule function based on their settings agreed with eDebit. However this setting applies to all account holders to be debited by that end user. An end user cannot use the Schedule function for some account holders and not others. In principle eDebit advises that each software application provides an internal billing or invoicing function to create and transmit the data to eDebit. The Schedule function is best suited to business owned web sites offering a fixed price product or service directly debiting their own clients and where there is an integration/tokenisation requirement.
11. All string values should be validated. Any control characters should be removed. Empty strings must be passed in where the value is optional and maximum length adhered to. The eDebit system will drop invalid characters and truncate strings that exceed the required length and this may lead to unexpected behaviour or data entry.

12. The API uses HTTPS POST functionality in conjunction with provided URLs. These URLs are dynamically provided based on usage, server load, down time etc. Software developers should not capture and 'hard code' the URLs as this may lead to connection issues. Integrated software should call for and utilise the returned URLs.
13. Where a value is to be passed in as part of the POST then this should be appended to the URL as a query string ensuring that the key is the same as that listed. Note keys are case sensitive and you should ensure that any casing is correct. Refer to Appendix A for examples. In addition standard formatting should be used. Replace spaces with + and delimit key value pairs with &.
14. We advise that POSTing should be undertaken using dynamic/programmatic web requests rather than setting the POSTBack URL on form submission to the provided URL. The latter option will inhibit the ability of the software to capture and utilise the return values, in addition information will be displayed with no formatting or adherence to the web site CSS/styles.

PCIDSS Compliance

eDebit in conjunction with its banking partners has been able to provide a PCIDSS compliant solution that has been extended to software that is integrated using this API as a result of the tokenisation functionality.

eDebit recognises that integrated software may have conflicting requirements with regards to storage of card or account information and that end users who conduct their own debits or use alternate service providers may require this information to be stored locally and this may provide a challenge. Storage of the masked number (refer below) in combination with the expiry date and name on the card is permitted under the PCI requirements where this is a requirement of the software, as a result database integrity may not be compromised.

'Downstream' compliance is dependent upon a number of key factors as follows;

1. Software integrated using this API must not store credit card or bank account numbers in full. Our API provides a masked number that can be used for verification; this consists of the first 6 and last 3 digits of the card number or the last 3 digits of the bank account number.
2. Software vendors intending to utilise our API are required to confirm in writing that when using this API that full card/account numbers are not stored or saved at any time.
3. Card or account information is to be pre-registered with eDebit and a 'token' is provided, all references to the account or card after issuance of the token must be referred to by the token.

Procedure

Any account or card holder who is to be debited using this API must exist on the eDebit Data Store and must have their account information preregistered.

Integrated software should request the URLs that will be used to send or provide information. These URLs are provided in simple key value pair string as a result of a HTTPS POST.

- POST the required information to the URL (https://www.edebit.com.au/IS/edEP.ashx). This URL is not dynamic (it may therefore be 'hard coded') and provides the entry point for the eDebit integration. You should read the return key value pair string and use the provided values. Each 'function' listed below has a corresponding key/value pair in the return string.

The procedure is to first insert or update the personal information held by the eDebit data store.

- Personal information requires a POST to the edPI URL. This POST will determine whether an account holder exists in the system and therefore requires insertion or whether this is an update to currently held data. Integrated software need only program a call to a single function to ensure information is either inserted or updated.

Once personal information has been inserted or updated then the account information must be preregistered as follows;

- POST to the edPW URL to obtain the cd_community and cd_supplier combination.
- POST to the edKI URL using user ID, password and the unique account holder reference to obtain a key to insert information into the PCI Compliant Data Store. Note the key is one time use only and is valid for 60 minutes.
- POST to the edReg URL using the key value obtained from edKI. Pass in the values required, note that some of these values are obtained from prior POSTings.
- Display the edPage URL in a browser control appending the required value to the URL. The end user inserts the relevant account information into the displayed URL. The user must then click Save or Cancel.
- Clicking Save will validate and pre-register the account. The browser control may be closed down after the form has been submitted. Note that a failed preregistration will require a new key. We would advise that a new process is started as the dynamically provided URLs may change. Keys are one time use only.
- POST to edTKI to obtain the information in relation to the token to be stored in the software. This call will provide confirmation of the preregistration and return the masked card/account value, name on the card/account and type of card (where applicable) or the reason for the failure. eDebit will record and store each attempt to preregister the card/account.
- Failed calls/updated information require that the preregistration process is completed as required. Note it is not possible to query the full account information for any card or bank account.

After an account has been successfully preregistered you may (optionally) request our server to create a schedule of payments by POSTing to edSched. Note edSched should not be used under the following circumstances;

- Where amounts will vary from period to period or over the duration of the debit contract.
- Where the integrated software will provide the amount to be debited.
- Where frequent adjustment of the schedule payment is required.

For example a debit contract for a subscription to an online service is ideally suited to using the Schedule function as the amount will not change over the duration of the contract other than a general price rise, debits due as a result of the ad hoc usage of a product or service which are time or usage based and therefore subject to frequent change are not suited to use the Schedule function.

The debit schedule function requires that you determine whether debits are to be 'until further notice' or not. Until further notice contracts will create a rolling forward schedule for 6 months. The only way to cancel a client on this type of contract is to cancel them by calling edSchCxl. Calling edSchCxl will remove all forward debits transactions and mark the account holder as cancelled preventing further debits.

Cancelled account holders may be reactivated by calling edSched for this account holder. Calling edSched for an existing account holder with a schedule in place will replace all forward dated debits with the new information.

Where the Schedule function has not been used the integrated software/web site will need to generate data to send to eDebit instructing the eDebit system to debit a card or bank account on a given date. This data is to be transmitted on the due date.

Data and debit information should be sent to eDebit on the date due to be debited prior to 1pm AEST/AEDST. The relevant data should be compiled into a query string and POSTed to the relevant eDebit URL. The following POSTs are required to be completed;

- edTS to obtain the URL to POST to
- POST to edTR/edTRBulk appending the query string with the debit information or attaching the file to the post for edTRBulk. Each transaction should be POSTed to the URL independently for edTR but bulk transactions can be sent via edTRBulk as a file attachment. A POST to these 'functions' will return a value indicating whether the post was successful or not (S or F) and the reason for the failure. Failed POSTings should be retried or discarded based on the error. This is to be under the control of the software/web site.
- Where edTRBulk is used then users are notified that once the file has been posted to the function it is not possible to stop the processing of the file and the included transactions.

Note: - POSTing duplicate transactions for the same account holder will result in 'double debits'. POSTing credit value transactions, i.e. those for an amount less than $0.00 is not permitted and these will be 'dropped' by the eDebit system with an error explanation. You must ensure that the amount transmitted is correct. Late adjustment is possible but only by direct access via our web site.

The status of a submitted transactions/debit may be determined by a call to edTStat. This will return the status of the debit. The following values apply;

- PLANNED – still to be submitted to the bank and is dated beyond the date of enquiry
- PENDING – submitted to the bank – outcome unknown.
- SUCCESSFUL – debit was successful and the funds have been remitted to the end user
- FAILED – the bank dishonoured the payment and the end user has been notified with the reason. The reason for the dishonour is included within the return values.

API Calls

- Entry Point (www.edebit.com.au/IS/edEP.ashx)
    - Values to be passed in
        - eDebit Number (End User Account No) – 6 digit number – must be stored locally – **key name = edNo**
        - Account Type, whether paying by bank account or credit card – **key name = accountType, only permitted values empty string (for card) or DD (for bank account)**
    - POST will return a series of key value pairs. Each key is the name of a function and the corresponding value is the URL to post the information to using a dynamic web request. The key value pairs will be;
        - edPI – URL to POST to
        - edPW – URL to POST to
        - edKI – URL to POST to
        - edReg – URL used by eDebit Account Registration Process
        - edPage – URL to display in browser control/web site
        - edTKI – URL to obtain token and masked information from for local database storage.
- edPI (user to insert or update personal information on an account holder)
    - Key/Value Pairs to be passed in
        - eDebit Number (End User Account No) 6 digit number – must be stored locally (Required) – **key name = eDNo**
        - Unique Reference ID. Up to 11 character string which must not contain punctuation, spaces or symbols. These will be dropped if present providing unexpected results. (Required) – **key name = clNo**
        - First name, maximum 45 character string. Punctuation will be escaped where present. (Optional – empty string if not present) – **key name = cl1stName.**
        - Last name or business name – as for first name – **key name = cl2ndName.**
        - Address, the street address of the account holder, 45 character string (optional) – **key name = clAddr.**
        - Suburb, the name of the suburb – as for address – **key name = clCity.**
        - State, the state or province of the account holder, 3 character string (optional) – **key name = clState.**
        - Postcode, the postcode for the account holder, 4 character string (optional) – **key name = clPCode.**
        - Telephone number. This MUST be the full 10 digit number including the area code, no () or spaces (optional) – **key name = clTel.**
        - Email. The fully qualified email address, please note where passed this must resolve back to a valid email account. 100 character string (optional, where on line signing is required then this value is required) – key **name = clEmail.**
        - Drivers licence name. The name on the driver licence, 45 character string (required for paperless sign up, optional otherwise) – **key name = clDlName.**

        - Drivers licence number. The number on the driver licence, 20 character string (required for paperless sign up, optional otherwise) – **key name = clDlNo.**
        - Drivers licence state of issue. The state of issue for the drivers licence, 3 character string (required for paperless sign up, optional otherwise) – **key name = clDlState.**
        - Account Type, whether paying by bank account or credit card – **key name = accountType, only permitted values empty string (for card) or DD (for bank account)**
        - Linked Marketing Service Provider. Where the end user is contracted to a linked marketing provider and payment is deducted from source on a percentage basis then this key value must be used. It must contain either the eDebit Account number for the linked marketing service provider or an empty string. **Key name = clMktNo.**
    - Return Value – String. Either S for successful insertion/update or F and a string detailing the error (up to 200 characters)
- edPW (used to obtain the password, user ID to start account registration).
    - Key/Value Pair to be passed in
        - eDebit Number (End User Account No) 6 digit number – must be stored locally (Required) – **key name = edNo.**
    - Return Value – key value pair string.
        - 1. Cd_community = cd_community
        - 2. Cd_supplier_business = cd_supplier_business
- edKI (used to obtain an encrypted key to instruct the eDebit system to pre-register an account)
    - Values to be passed in (in order)
        - cd_crn = Unique customer reference ID for this account holder. This must consist of the eDebit Number plus hyphen plus up to 11 character string. No spaces or punctuation. Eg 100100-smithj or 100100-12345678901
    - Return Values
        - Token, this is an encrypted string of variable length and Error description. If the error value is empty/empty string then you may proceed to next step. If not then the process must be halted until the error is corrected.
- edReg (used to set environment for preregistration of an account – bank or card)
    - Values to be passed in
        - Company code from edPW – **key name = cd_community**
        - Business code from edPW – **key name = cd_supplier_business**
        - Unique account holder reference – **key name = cd_crn must be same value as cd_crn under edKI**
        - Account Type, must be DD for bank accounts for credit cards it should be an empty string – **key name = accountType**
        - Security code from edKI – **key name = token**
    - NO Return Values
- edPage

- Browser Control should now display the URL returned in edEP for edPage key. Note you must append the cd_crn used in edKI as a query string.
- User Must click Save or Cancel on displayed URL
  - Cancel will void the process and return nothing.
  - Save will register the account.
  - The browser control may be closed
- A successful save or a cancel will attempt to close the web page in the browser control. Invalid data will stop the save and highlight the invalid entry/ies.

- edTKI
  - Values to be passed in
    - Unique account holder reference – **key name = cd_crn – must be the same value as used under edKI and edReg**
  - Return values
    - The token value to refer to this account by in the future for transaction processing etc – **key name = token**
    - Account Holder, the name on the card/account as provided to us via the web page entry – **key name = acc_holder**
    - Account Alias. This is the masked version of the full card or account information. For credit cards this will be the first 6 and the last 3 digits of the card and for bank accounts will be the last 3 digits only – **key name = acc_alias**
    - Other account information. For credit cards this will be the expiry date in the form mm/yy for bank accounts this will be the BSB number in the form 000-000 – **key name = acc_info**
    - Error. This will either contain S for a successful registration or F followed by a hyphen and a description if the registration failed – **key name = error**
- edSched
  - Values to be passed in
    - eDebit Account No – **key name = edNo**
    - Account holder reference number – **key name = clNo**
    - Debit Frequency, **key name = clFreq.** Must be one of the following values
      - 7D = Weekly
      - 14D = Fortnightly
      - 28D = 4 Weekly
      - 1M = Calendar Monthly
      - 3M = Quarterly
    - Debit Amount, must be a currency amount expressed in dollars and cents with 2 decimal places and NO currency symbol, e.g. 10.00 for $10.00 – **key name = clAmt.**
    - No of Debits to complete. This must be a valid number greater than 1. – **key name = clDebitCnt**
    - Until further notice flag. If the client is to be debited until further notice then this key must have a value of 1, if not until further notice then the value is 0. **Key name = clUFN.**

        - Date to start the debit schedule on, This will be the first debit date and the debits will occur at the frequency stated on the anniversary of this date. Must be a date in the format DD/MM/YY. **Key name = clDebitDate**
        - Add Transaction Fee. Flag to add the transactions fee to the debit amount. Must be either 0 or 1, where 1 sets the flag to add the fee. **Key name = clTFee.**
        - Add Merchant Service Fee, set the flag to add a the merchant service fee for credit card transactions only. 0 or 1 where 1 sets the flag. **Key name = clMSF.**
    - Return Values
        - Error. This will either contain S for a successful process or F followed by a hyphen and a description if the process failed – **key name = error**
- edSchCxl
    - Values to be passed in
        - eDebit Account No – **key name = edNo**
        - Account holder reference number – **key name = clNo**
    - Return Values
        - Error. This will either contain S for a successful process or F followed by a hyphen and a description if the process failed – **key name = error**
- edTS
    - Values to be passed in
        - eDebit Account No – **key name = edNo**
    - POST will return a key value pair.
        - edTR – URL to POST to transactions/debits to
- edTR
    - Values to be passed in
        - eDebit Account No – **key name = edNo**
        - Account holder reference number – **key name = clNo**
        - Token number provided for this account holder – **key name = token.**
        - Amt. The amount expressed as a decimal to 2 decimal places – no currency symbol specifying the amount to deducted from the tokenised account – **key name = amt**
        - Description, a description of the debit – **key name = desc.**
        - A unique reference for this debit. This must be a unique id on a local data store. This value will be passed back when updating the status of the debit – **key name = id.**
    - Return value
        - Error. This will either contain S for a successful process or F followed by a hyphen and a description if the process failed – **key name = error**
- edTRBulk
    - Values to be include in each line of the file. The file must be in csv format with no column headings and attached to the post. Each transaction must be on a discrete line. It is possible to aggregate transactions from different eDebit Numbers using this process as each line is processed independently. Once the file has been posted to

the server it is not possible to prevent processing of the file or insertion of the transaction lines.

- eDebit Account No – **key name = edNo**
- Account holder reference number – **key name = clNo**
- Token number provided for this account holder – **key name = token.**
- Amt. The amount expressed as a decimal to 2 decimal places – no currency symbol specifying the amount to deducted from the tokenised account – **key name = amt**
- Description, a description of the debit – **key name = desc.**
- A unique reference for this debit. This must be a unique id on a local data store. This value will be passed back when updating the status of the debit – **key name = id.**

  - Return value
    - Error. This will either contain S for a successful process or F followed by a hyphen and a description if the process failed – **key name = error**

- edTStat
  - Values to be passed in
    - eDebit Account No – **key name = edNo**
    - Debit ID – ID submitted under edTR – **key name = id**
  - Return value
    - status, either PLANNED, PENDING, FAILED or SUCCESSFUL – **key name = status**
    - the reason for the dishonour for failed transactions only, otherwise it will be an empty string – **key name = desc**
    - error – a description of the error – **key name = error**
- edTrAmend
  - Values to be passed in
    - eDebit Account No – **key name = edNo**
    - Account Holder Reference No – **key name = clNo**
    - Either the single date to amend a transaction for or the lower date value in a date range – **key name = stDate**
    - Either the upper date value in a date range or an empty string where stDate refers to a single transaction date – **key name = endDate**
    - The new value to be used to overwrite the current value. This may be $0 or greater – **key name = amt**
  - Return value
    - String value indicating success (S) or failure (F) followed by the error description

Appendix A

Samples (Do NOT use sample information provided to hard code URLs)

- Entry Point -edEP
  - POST to
    - https://www.edebit.com.au/IS/edEP.ashx?edNo=100100
  - Return key value pair string
    - edPI=https://www.edebit.com.au/IS/edPI.ashx&edPW=https://www.edebit.com.au/IS/edPW.ashx&edKI=https://www.edebit.com.au/IS/edKI.ashx
- edPI
  - POST to
    - https://www.edebit.com.au/IS/edPI.ashx?edNo=100100&clNo=1234&cl1stName=Fred&cl2ndName=Smith&clAddr=1+The+Street&clCity=Melbourne&clState=VIC&clPCode=3012&clTel=0312341234&clEmail=fredsmith@domain.com.au&clDlName=Fred+Smith&clDlNo=12345&clDlState=VIC&accountType=DD&clMktNo=200200
  - Return key value pair string
    - Result=S&Error=
    - Result=F&Error=Missing value clNo
- edPW
  - POST to
    - https://www.edebit.com.au/IS/edPW.ashx?edno=100100
  - Return key/value pair string
    - Cd_community=business+name&cd_supplier_business=45632345
- edKI
  - POST to
    - https://www.edebit.com.au/IS/edKI.ashx?cd_crn=100100-12345
  - Return key/value pair string
    - token=cp6FPyPZTI458mHIG9b4tRmz0E123gwh&error=
- edPage
  - Display the URL from edEP in the key value pair for edPage. Ensure you append the cd_crn to the URL as a query string – see example
    - https://www.edebit.com.au/IS/edPage_Name.htm?cd_crn=100100-12345
- edTKI
  - POST to
    - https://www.edebit.com.au/IS/edTKI.ashx?cd_crn=100100-12345
  - Return key value string
    - token=abcdf34gH!2&acc_holder=Fred Smith&acc_alias=456412*********345&acc_info=11/14&error=S (credit card token) or
    - token=abcdf34gH!2&acc_holder=Fred Smith&acc_alias=****345&acc_info=012-012&error=S (bank account token) or

        - token=&acc_holder=&acc_alias=&acc_info=&error=F-eDebit Account No not valid
- edSched
    - POST to https://www.edebit.com.au/IS/edSched.ashx this URL is not dynamic and may be hard coded.
        - https://www.edebit.com.au/IS/edSched.ashx?edNo=100100&clNo=12345&clFreq=14D&clAmt=25.00&clDebitCnt=26&clUFN=0&clDebitDate=01/01/12&clTFee=0&clMSF=0
    - Return Value
        - error=S or
        - error=F-Invalid eDebit No
- edSchCxl
    - POST to https://www.edebit.com.au/IS/edSchCxl.ashx this URL is not dynamic and may be hard coded.
        - https://www.edebit.com.au/IS/edSchCxl.ashx?edNo=100100&clNo=12345
    - Return Value
        - error=S or
        - error=F-Invalid eDebit No
- edTS
    - POST to https://www.edebit.com.au/IS/edTS.ashx this URL is not dynamic and may be hard coded.
        - https://www.edebit.com.au/IS/edTS.ashx?edNo=100100
    - Return Value
        - error=S or
        - error=F-Invalid+eDebit+No
- edTR
    - POST to
        - https://www.edebit.com.au/IS/edTR.ashx?edNo=100100&clNo=12345&token=asdfbg$rer66&amt=25.00&desc=debit+no+25&id=123667
    - Return Value
        - error=S or
        - error=F-Invalid+eDebit+No
- edTRBulk
    - POST to
        - https://www.edebit.com.au/IS/edTRBulk.ashx?edNo=100100 + attached file contents in post
    - Return Value
        - error=S or
        - error=F-Invalid+eDebit+No
- edTStat
    - POST to
        - https://www.edebit.com.au/IS/edStat.ashx?edno=100100&id=34556
    - Return Value
        - status=PENDING&desc=&error=
            - Status options are;

        - PLANNED
        - PENDING
        - FAILED
            - Failed transactions will carry the reason for the failure in the desc key.
            - eg. status=FAILED&desc=Refer to Customer&error=
        - SUCCESSFUL
    - Or status=&desc=&error=invalid+edebit+no
- edTrAmend
    - POST to
        - https://www.edebit.com.au/IS/edTrAmend.ashx?edNo=100100&clNo=12345&stDate=01/01/2013&endDate=01/02/2013&amt=25.00
        - The above will modify all transaction to the stated amount for the stated client between 01/01/2013 and 01/02/2013 but NOT including 01/02/2013, that is we use greater than or equal to stDate and less than endDate. OR
        - https://www.edebit.com.au/IS/edTrAmend.ashx?edNo=100100&clNo=12345&stDate=01/01/2013&endDate=&amt=25.00
        - The above will modify all transaction to the stated amount for the stated client on 01/01/2013 ONLY.
        -
    - Return Value
        - error=S or
        - error=F-Invalid+eDebit+No
-