PREPARED BY:

Pillars
Amman – Jordan.
+962 6 222 4545

Pillars Receive API

PREPARED FOR:

WORLDWIDE CASH EXPRESS™ MONEY TRANSFER

Confidentiality

The information contained herein is considered confidential and proprietary and is intended solely for use in the evaluation of **Pillars** services. The information contained herein shall not be disclosed, in whole or part, to any third party including other employees not participating in the evaluation of this proposal. That data shall be maintained with the same degree of care the above-named company uses to maintain its own confidential information. **Pillars** reserves the right to ask for the return, at its discretion, of any and all materials furnished in conjunction with this proposal.

Table of Contents

Definitions, Acronyms, and Abbreviations

Definition	Description
API	Application Programming Interface

Introduction

PILLARS Receive API is a collection of remittance APIs intended to manage the communication between Pillars and partner systems as part of the funds remittance process. The APIs included are:

- ## Validate Account
  Verifies / validates customer Account details with the partner.

- ## Load
  Pushes funds to the partner by providing the Account and the transaction details so the partner can deposit funds to the account.

- ## Check Balance
  Check Balance to get the current balance with currency ISO3.

- ## Check Status
  Check Status for the Transaction.

- ## Reconciliation Report
  Check Status for the Transaction.

Purpose

This document is aimed at providing details of the PILLARS Receive API to developers and technical personnel of partners who will be implementing this API to enable them to Send remittance transactions Received by PILLARS.

Validation Account Process

Pillars Received transaction is generated at partner side. Pillars makes an **HTTP POST** call to the Sender Partner.

Pillars validates the Account number for the transaction based on internal business rules. If the Account are valid, the API responds back with status 1.

If invalid, the API provides the appropriate response code. Pillars interprets the code and provides an appropriate message at the partner side. Based on the response code received, the transaction may be resubmitted after corrections have been made.

Validation Data Process

Pillars Received transaction is generated at partner side. Pillars makes an **HTTP POST** call to the Sender Partner.

Pillars validates the certain values for the transaction based on the internal business rules. If the transaction data are valid, the API responds back with status 1.

If invalid, the API provides the appropriate response code. Pillars interprets the code and provides an appropriate message at the partner side. Based on the response code received, the transaction may be resubmitted after corrections have been made.

Load Process

A successful validation results in the generation of a payment load by initiating an **HTTP POST** call to the API with transaction reference number and other transactional information.

If the transaction is valid, the API responds back with status **1** with an optional Success Response object. Partner will mark the transaction as Pending or Sent based on the API response.

If the transaction is not successful, the API provides appropriate response code. Partner interprets the code and marks the transaction as rejected or for repost.

Check Balance Process

Pillars Received transaction is generated at partner side. Pillars makes an **HTTP POST** call to the Sender Partner.

Pillars check the partner balance based on the internal business rules. The API responds back with status 1 and current Balance.

In case of error occur, the API provides the appropriate response code. Pillars

interprets the code and provides an appropriate message at the partner side. Based on the response code received, the transaction may be resubmitted after corrections have been made.

Check Status Process

Pillars Received transaction is generated at partner side. Pillars makes an **HTTP POST** call to the Sender Partner.

Pillars validates the transaction Number based on their internal business rules. If the transaction is valid, the API responds back with status 1 and string message illustrate the actual status of transaction.

If invalid, the API provides the appropriate response code. Pillars interprets the code and provides an appropriate message at the partner side. Based on the response code received, the transaction may be resubmitted after corrections have been made, it will Return The status of Transaction if it is repeated.

Reconciliation Report Process

Pillars Received transaction is generated at partner side. Pillars makes an **HTTP POST** call to the Sender Partner.

Pillars will support you in transaction Reconciliation. API that's take a specific Date

And will retrieve all transactions made in that day, the API responds back with

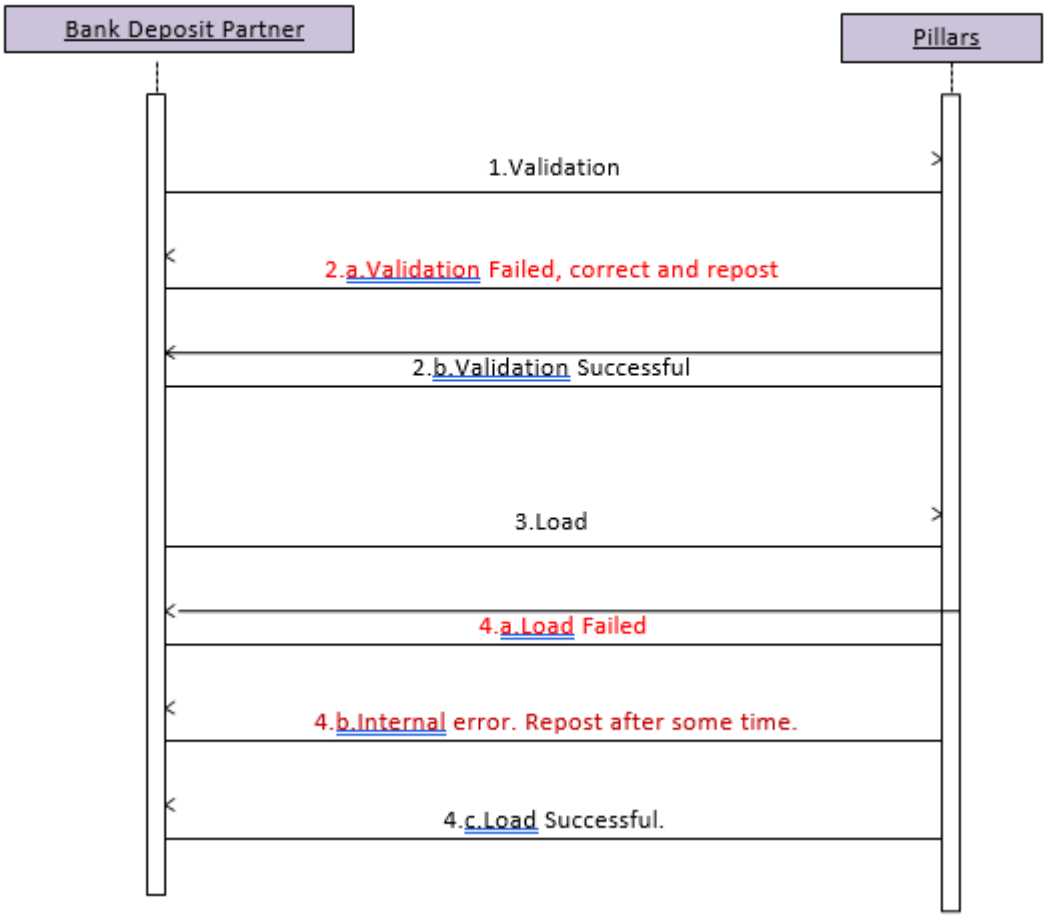
the appropriate response code,

Security Configuration for Pillars Receive API:

Pillars IT security policy allows the following security protocol for Receive API:

- IP whitelisting: At the firewall level, Pillars will whitelist partner IPs to allow SSL connection only for these approved IPs.
- HTTP Basic Authentication with SSL:
- Mutual Authentication: Pillars and the partner share the public certificate with each other. The partner presents this certificate in the initial SSL Handshake.

Sequence Diagram



Validate Account

RESOURCE URI:

Method	API URL	Description
POST	https://ce-staging.alalamifs.com:4030/api/Transactions/ValidateAccount	Validate Transaction

Validate Data

RESOURCE URI:

Method	API URL	Description
Post	https://ce-staging.alalamifs.com:4030/api/Transactions/ValidateData	Validate Data

Load Account

RESOURCE URI:

Method	API URL	Description
Post	https://ce-staging.alalamifs.com:4030/api/Transactions/LoadTransaction	Load funds into customer Account

Check balance

RESOURCE URI:

Method	API URL	Description
Post	https://ce-staging.alalamifs.com:4030/api/Transactions/CheckBalance	Check The balance for the Customer

Check Status

RESOURCE URI:

Method	API URL	Description
Post	https://ce-staging.alalamifs.com:4030/api/Transactions/CheckStatus	Status for the Transaction

Reconciliation Report

RESOURCE URI:

Method	API URL	Description
Post	https://ce-staging.alalamifs.com:4030/api/Transactions/ReconciliationReport	Status for the Transaction

REQUEST ATTRIBUTES:

Note:

- This message includes a request header.
- Optional fields are populated on the message based on the agreement between Pillars and the partner. Required fields are part of the message.

Attributes	Description	Type	Optional/Required/Not Required
ProviderId	Unique Id provided by pillars	globally unique identifier	Required
UserId	Unique user Id provided by pillars	globally unique identifier	Required
SecurityInfo	Encrypted data	string	Required
SecretKey	Encrypted Key	string	Required

To Encrypt the Security Info:

1. Generate AES Key with the following options:
   - **Key size:** 256
   - **Block size:** 128
   - **Mode:** CBC
   - **Padding:** PKCS7
   - **IVSTR:** IV
2. Fill the request with the following parameters:
   - **Provider Id:** Provided by **Pillars.**
   - **User Id:** Provided by **Pillars.**
   - **Security Info:** Encrypted data on the AES Key generated before.
   - **Secret Key:** The AES Key generated before must be encrypted on RSA algorithm by public key provided by **Pillars.**

In general, this is the Request format:

```
{
  "ProviderId": "",
  "UserId": "",
  "SecretKey": "sample string 3",
  "SecurityInfo": "sample string 4"
}
```

Security Sample Code (C#):

```
        public string GenerateAESKey()
        {
            RijndaelManaged aesEncryption = new RijndaelManaged();
            aesEncryption.KeySize = 256;
            aesEncryption.BlockSize = 128;
            aesEncryption.Mode = CipherMode.CBC;
            aesEncryption.Padding = PaddingMode.PKCS7;
            aesEncryption.GenerateIV();
            string ivStr = Convert.ToBase64String(aesEncryption.IV);
            aesEncryption.GenerateKey();
                string keyStr = Convert.ToBase64String(aesEncryption.Key);
            string completeKey = ivStr + "," + keyStr;

            return Convert.ToBase64String(ASCIIEncoding.UTF8.GetBytes(completeKey));
        }


        public string EncryptOnAES(string PlainText, string EncodedKey)
        {
            RijndaelManaged aesEncryption = new RijndaelManaged();
            aesEncryption.KeySize = 256;
            aesEncryption.BlockSize = 128;
            aesEncryption.Mode = CipherMode.CBC;
            aesEncryption.Padding = PaddingMode.PKCS7;
            aesEncryption.IV =
Convert.FromBase64String(ASCIIEncoding.UTF8.GetString(Convert.FromBase64String(EncodedKey)).Split(',')[0]);
            aesEncryption.Key =
Convert.FromBase64String(ASCIIEncoding.UTF8.GetString(Convert.FromBase64String(EncodedKey)).Split(',')[1]);
            byte[] plainText = ASCIIEncoding.UTF8.GetBytes(PlainText);
            ICryptoTransform crypto = aesEncryption.CreateEncryptor();
            byte[] cipherText = crypto.TransformFinalBlock(plainText, 0, plainText.Length);
            return Convert.ToBase64String(cipherText);
        }


public string EncryptOnRSA(string PlainText, string PublicKey)
        {
            RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
            rsa.FromXmlString(PublicKey);

            byte[] byteText = Encoding.UTF8.GetBytes(PlainText);
            byte[] byteEntry = rsa.Encrypt(byteText, false);
            return Convert.ToBase64String(byteEntry);
          }
```

Request Formats for Validate Account:


```
{"Header”: {
"UserName":"",
"Password":"",
"RequestDate":"2022-10-16T09:31:12.7531268+03:00"
},
"Data”: {
"AccountType":1,
"AccountNumber":"07902567052"
}
}
```


Request Formats for Validate Data:


```
{"Header”:
{
"UserName":"",
"Password":"",
"RequestDate":"2022-10-16T10:29:38.3623173+03:00"},
"Data”: {
"ResonOfTransferCode":"20",
"RelationshipCode":"01",
"SourceOfFundCode":"1"
}
```

Request Formats for Load Account:


```
{"Header":
{
"UserName":"",
"Password":"",
"RequestDate":"2022-10-16T10:37:13.4986537+03:00"},

"Data":
{"TransactionData":
{
"AccountType":2,
"AccountNumber":"0791150624",
"TransactionId":"2",
"RelationshipCode":"",
"PurposeOfTransferCode":"20",
"SendCountryISO3":"JOD",
"SendAmount":200.0,
"SendCurrencyISO3":"USD",
"ReceiveAmount":200.0,
"ReceiveCurrencyISO3":"JOD",
"SenderInfo":
{
"FirstName":"lena",
"SecondName":"omar",
"ThirdName":"moh",
"LastName":"albayed",
"SourceOfFundCode":"",
"Adress":"Amman",
"IdentityTypeCode":"1",
"IdentityNumber":"194400",
"NationalNumber":"9982014665",
"NationalityISO3":"JOD",
"DateOfBirth":"1992-10-16T10:37:13.5142531+03:00",
"CountryOfBirthISO3":"Jodran",
"MobileNumber":"0791111111",
"IdentityExpiryDate":"2022-10-16T10:37:13.5295487+03:00",
"IdentityIssuePlace":"amman"
},
"ReceiverInfo":
{
"FirstName":"moh",
"SecondName":"omar",
"ThirdName":"moh",
"LastName":"albayed",
"Adress":"amman",
"NationalityISO3":"JOD",
"MobileNumber":"0784509560"
},"Additional Data”: [ ]   }   }      }
```

Account Number Format:

- Wallet Number:
  - 009627xxxxxxxx
  - 9627xxxxxxxx
  - 07xxxxxxxx
- IBAN:
  - JO94 CBJO 0010 0000 0000 0131 0003 02
  - JO00 XXXX 0000 0000 0000 0000 0000 00

Request Formats for Check Status:

```
{"Header": {
"UserName":"",
"Password":"",
"RequestDate":"2022-10-16T10:40:19.7568167+03:00"},
"Data": {
"TransactionId":"20"
}
}
```

Request Formats for Check Balance:

```
{"Header":{
"UserName":"",
"Password":"",
"RequestDate":"2022-10-17T12:17:59.0622652+03:00"},
"Data":{
"MTO_ProviderId":""
}
}
```

Request Formats for Reconciliation:


```
{
  "Header": {
    "UserName": "",
    "Password": "",
    "RequestDate": "2022-12-13T10:13:03.0475341+03:00"
  },
  "Data": {
    "MTO_ProviderId": "",
    "ReportDate": "2022-12-09T00:00:00"
  }
}
```


Response Attributes:

Note:

- For success validation response, response body is optional. HTTP code 200 maps to success response by default.
- If the validation fails, the API responds back with an error object in the response body.

Attributes	Description	Type	Optional/Required/Not Required
Response Result	Response result contain the response header Details	**Type:** ResponseHeader	Required

Response Header: Object

Attributes	Description	Type	Optional/Required/Not Required
StatusCode	Status Code contain the status code Details	**Type:** ResultType	Required
StatusDesc	Status Description Contain the Status Description details	**Type:** string	Required
MessageReason	Message Reason	**Type:** string	Required
MessagesList	Message List contain message list Description	**Type:** Collection of MessagesListBTO	Required

Result Type: Object

Attributes	Description	Value
Info	The Info code	**0**
Successfully	Successfully Code	**1**
Warning	Warning Code	**2**
Error	Error Code	**3**
Unauthorized	Unauthorized Code	**4**

Messages List BTO: Object

Attributes	Description	Type	Optional/Required/Not Required
MessageCode	Message Code contain the Message Code Details	**Type:**string	Required
MessageDesc	Message Description Contain the Message Description details	**Type:** string	Required

Response Formats for Validate Account:

```
{
  "ResponseResult": {
    "StatusCode": 0,
    "StatusDesc": "sample string 1",
    "MessageReason": "sample string 1",
    "MessagesList": [
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      },
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      }
    ]
  },
  "TargetAccount": "sample string 1"
}
```

Response Formats for Validate Data:


```
{
  "ResponseResult": {
    "StatusCode": 0,
    "StatusDesc": "sample string 1",
    "MessageReason": "sample string 1",
    "MessagesList": [
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      },
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      }
    ]
  }
}
```


Response Formats for Load Transaction:


```
{
  "ResponseResult": {
    "StatusCode": 0,
    "StatusDesc": "sample string 1",
    "MessageReason": "sample string 1",
    "MessagesList": [
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      },
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      }
    ]
  },
  "TargetAccount": "sample string 1",
  "TransactionId": "sample string 2"
}
```

Response Formats for Check Status:


```
{
  "ResponseResult": {
    "StatusCode": 0,
    "StatusDesc": "sample string 1",
    "MessageReason": "sample string 1",
    "MessagesList": [
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      },
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      }
    ]
  },
  "TransactionStatusId": 1,
  "TransactionStatusDescription": "sample string 2"
}
```


Response Formats for Check Balance:


```
{
  "ResponseResult": {
    "StatusCode": 0,
    "StatusDesc": "sample string 1",
    "MessageReason": "sample string 1",
    "MessagesList": [
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      },
      {
        "MessageCode": "sample string 1",
        "MessageDesc": "sample string 2"
      }
    ]
  },
  "AvailableBalance": 1.1
}
```

Response Formats for Load Transaction:


```
{
    "ResponseResult": {
        "StatusCode": 1,
        "StatusDesc": "Successfully",
        "MessageReason": "",
        "MessagesList": [
            {
                "MessageCode": "20",
                "MessageDesc": "Success"
            }
        ]
    },
"Reports": [
        {
            "TransactionNumber": "438935",
            "TransactionDate": "2022-12-09T13:21:28.093",
            "Amount": 455.69,
            "Fees": 0.0,
            "CurrencyISO3": "JOD",
            "TransactionStatus": 0,
            "TransactionStatusDsc": "Entered",
            "AccountNumber": "JO19IIBA11000000011000012844510",
            "AccountType": "2",
            "AccountTypeDsc": "Bank IBAN"
        },
        {
            "TransactionNumber": "438927",
            "TransactionDate": "2022-12-09T12:10:07.89",
            "Amount": 305.5,
            "Fees": 0.0,
            "CurrencyISO3": "JOD",
            "TransactionStatus": 11,
            "TransactionStatusDsc": "Rejected Not Updated",
            "AccountNumber": "JO71TEST0000000000897849710795",
            "AccountType": "2",
            "AccountTypeDsc": "Bank IBAN"
        }
 ]
}
```

Appendix A

Note:

- Predefined error codes available for partner

Code	Message
01	Invalid Transaction Id
02	Duplicate Transaction
03	Wallet Number Syntax Incorrect
04	Invalid Account Number
05	Valid Account Number
06	Valid Data
07	Invalid Reason Of Transfer
08	Reason Of Transfer Is Required
09	Invalid Relationship
10	Relationship is Required
11	Invalid Sender Id Type
12	Sender Id Type is Required
13	Sender Id Number is Required
14	Sender Address is Required
15	Transaction Already In Progress
16	Transaction in AML Hold
17	Transaction Already Paid
18	Transaction Rejected
19	Funds Deposited
20	Success
21	Internal Server Error
22	Transaction Not Exist
23	Sender Frist Name Is Required
24	Sender Last Name Is Required
25	Receiver Frist Name Is Required
26	Receiver Last Name Is Required
401	Unauthorized

Appendix B

Note:

- Predefined Purpose of Transaction codes available for partner

---

Code	Description
01	Education and Training
02	Personal
03	Legal Obligation (alimony, court fee, etc.)
04	Investment/Savings
05	Business Expense / Employee Remittance
06	Loan
07	Gift
08	Donation or Financial Aids
09	Family Support
10	Purchase of Goods
11	Travel Expenses
12	TRANSFER TO NRE ACCOUNT
13	INVESTMENT IN MUTUAL FUND/INSURANCE
14	INVESTMENT THROUGH BANK
15	EDUCATION/TUITION/BOARDING
16	HOTEL ACCOMODATIONS
17	TRAVEL AGENT
18	TRADE TRANSACTION
19	UTILITY PROVIDER
20	TAX PAYMENT
21	HOSPITAL OR MEDICAL INSTITUTION
22	PURCHASE OF PROPERTY

Appendix C

Note:

- Predefined Relationships codes available for partner

Code	Description
01	Close Friend
02	Business Partner
03	Client
04	Acquaintance
05	Family Member
06	Employee
07	My Self
08	Contractor
09	Employer
10	Online Friend
11	Third Party
12	Vendor

Appendix D

Note:

- Predefined Identity Types codes available for partner

Code	Description
01	National ID
02	Military Card
03	Refugee Card
04	Passport
05	Resident Card ID
06	Work Permit
07	Trade license - Corporate
08	Trade license - Financial Institution

Appendix E

Note:

- Predefined Source of Funds available for partner

Code	Description
01	Salary
02	Sale of Property
03	Family Funds
04	Loan
05	Savings
06	Inheritance
07	Pension