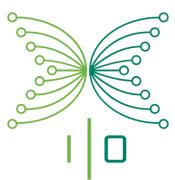
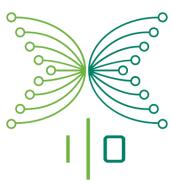
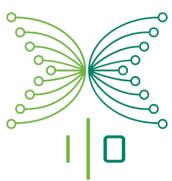


Overview





Stakeholders	3
Host	3
Distributor	3
Buyer	3
System modules	4
Frontend	4
Backend	4
Json Generator	4
AVVM	4
Volumes	5
Overview	5
Frontend - Business Logic	5
Frontend - System Setup	6
Frontend - Technical Documentation	8
Frontend - 3rd party Software and Services	9
Backend - System Setup	9
Backend - Transaction Fees Structure	10
Backend - Authentication	10
Backend - Api Doc	11
Backend - Api Troubleshooting	12



Stakeholders

The application is considered to have three stakeholders, the host, distributors and buyers. All of them are needed to carry out a sale.

Host

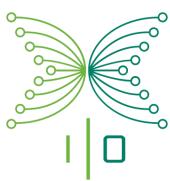
The sales have a host responsible for launching sales, inviting partners and audit members. The host is setting the frames for the sale by selecting trusted partners and gate keep any invited buyer.

Distributor

The distributors are responsible for expanding the network by recruiting more distributors or inviting buyers to purchase. Commissions are earned based on purchases in their downline. The top distributors are recruited directly by the host and are called partners.

Buyer

The buyers are the ones that actually make purchases in the system. By getting invited by a distributor and later approved by the host the buyers will be enabled to make their purchases.



System modules

This section explains the different kinds of systems used in the application.

Frontend

This module represents the visible face of the Sales App, and its main purpose is to provide an user interface for the actors, and manage all their actions. It also manages the business rules for distributing and ordering ADA.

Backend

Backend supplies a layer of abstraction to the sales app from the Bitcoin Payment network and processing. It's structured as a service API that Sales app can use to create payment address, make transactions, pay commission, etc.

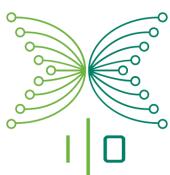
Json Generator

The JSON Generator module consists of an API that allows the Sales Frontend application to communicate with the AVVM, and is responsible for the interaction between the application and the exposed AVVM methods.

AVVM

The AVVM (Ada Voucher Vending Machine) is one of the most important and secure modules. It is a service that exchanges the paid invoices for ADAs.

In order for the Frontend application to interact with the AVVM, it must use the JSON Generator interface.



Volumes

Overview

This current document provides a quick summary of the Sales App documentation files. You will find a brief summary of their contents and its main highlights, explaining its most important sections.

Frontend - Business Logic

The business logic documentation shows how a user of the system would navigate through the application. It explains the different types of users and how they interact with the system. The different email types are explained as well.

- **Actors**

This part explains which users (actors) are in the system and what their role is.

- **Supported Languages and Regions**

All the languages and regions supported by the application are listed here.

- **Account Creation**

The account creation section explains the different ways on how to create the necessary accounts needed to use the application. It also explains how an admin and an external actor can create an account.

- **Enrollment**

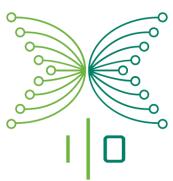
The enrollment section explains how partners and distributors can create urls to enroll buyers. The enrollment process itself is explained in detail by the use of images.

- **Approval**

This section explains how a compliance officer can approve a distributor or buyer. The whole process is shown step by step. It also shows how a CO can find users and filter them by different search criterias.

- **Invoice Tickets**

Describes the most important aspects of an Invoice ticket; how a user can reorder ADA, when the receipt is sent, and how the commissions are distributed.



- **Monitor the Sale**

This section explains when the sale starts and how the admin can check the current status.

- **User Support**

Describes the actions that a customer service officer can take in order to do client support. Also describes the investigation process when a distributor has a **suspicious** activity.

- **System Operation**

This section is about how the sysop user can monitor the sales and application workers.

- **Self Administration by External Actors**

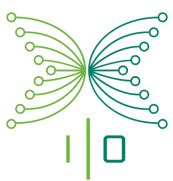
Describes the actions that buyers and distributors can perform in order to manage their account and personal data.

- **Misc**

This part contains miscellaneous topics like; The bitcoin rate used by the App, how users log in, etc.

- **Emails**

Describes all types of emails that the application can send.



Frontend - System Setup

The System Setup document explains what is necessary to configure your environment correctly to work with this app. Here you will find four main sections, which are the following:

- **System Requirements**

List of the requirements or standards that the system must meet in order to run the app. For example you must have installed a specific Node version.

- **System Configuration**

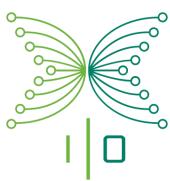
Specifies how to properly configure settings that are critical for the correct system behaviour.

- **Deployment**

This part of the document specifies what software is needed to deploy, why it's needed and how to deploy.

- **Known Quick Fixes**

This part describes the fixes that are used when the application runs into some unforeseen problems, these mostly originate from scaling up the application.



Frontend - Technical Documentation

The technical documentation is oriented to future developers or any person with a technical profile that wants to understand the development process taken, which frameworks and tools were used, and what the current status of the development is. In order to do so, the document is divided into the following sections:

- **Development System Requirements**

Describes the hardware and software that is required to develop the app.

- **Code Structure**

Explains the naming convention for files and the folder structure. It also introduces the most important files for guiding a developer through the project.

- **Code Base Metrics**

This section contains metrics about the code and provides stats about the developing process.

- **Development Process**

Describes the workflow throughout the development cycle and the different methodologies used.

- **API**

This section explains the Sales application API and which systems use it.

- **Advanced Features**

This section explains the most important aspects of the technical challenges and solutions used to develop the Sales application.

- **Languages, Standards and Frameworks**

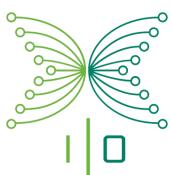
Describes the programming languages, standards and frameworks adopted to develop the system.

- **Improvement Suggestions**

Lists the improvements to the system suggested by developers.

- **Known Bugs**

Lists the known bugs that currently exist and provides links to detailed information about them.



Frontend - 3rd party Software and Services

- **Services**

Summary of 3rd party services which the sales application is relying on.

- **Packages**

For the application we use many 3rd party modules for either Meteor or Node. Any pieces of software used from sources other than our own will be listed here.

- **Software**

Summarizes the tools and plugins that the developers of the app used in order to improve the development process.

Backend - System Setup

- **System Requirements**

List of the requirements or standards that the system must meet in order to run the app. For example you must have installed a specific Node version.

- **System Configuration**

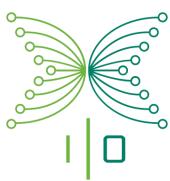
Specifies how to properly configure settings that are critical for the correct system behaviour.

- **Deployment**

This part of the document specifies what software is needed to deploy, why it's needed and how to deploy.

- **Recommendations**

Tips to ease up deployment and avoid problems.



Backend - Transaction Fees Structure

Although Bitcoin Transaction fee represent a small amount of the transaction, from an accounting perspective it's very important to have a clear understanding by who and how are they taken off.

- **Backend Transactions Types**

Summary of the types of transactions.

- **Fees Calculation**

Formula for calculating fees per kilobyte.

- **Fees Application**

Summary about fees for each transaction type.

Backend - Authentication

The authentication is a key security feature of the process, this document explains in detail how it's managed for every endpoint, the protocol, the use of tokens and payload signature.

- **Authentication**

Describes what kind of endpoint there are and how the authentication process is done between sales application and backend.

- **Revoke Access**

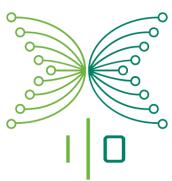
Describes how to revoke access of certain token.

- **Security Concerns**

Describes what kind of concerns should you be aware of when the backend gets compromised.

- **Access Token Generation**

Describes how to generate access token for the backend.



Backend - Api Doc

Here you'll find a detailed explanation of every endpoint from a technical perspective, the method names, parameters it receives or requires and possible workflow interactions.

- **Distributor Named Invoice Wallet**

Describes the process of creating a new distributor wallet.

- **Get New Invoice Addresses**

Describes the process of generating a new invoice address.

- **Holding Wallet Funds Received**

Describes the process when holding wallet receives funds.

- **Bitcoin Received on Invoice Address**

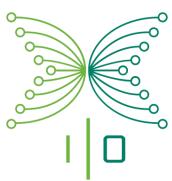
Describes the process when invoice ticket receives a payment in bitcoins.

- **Commission Wallet Funds Received**

Describes the process when commission wallet receives funds.

- **Fix Wrong Amount and Invoice Refunds**

Describes the process when buyer sends in too small or too big amount.



Backend - Api Troubleshooting

None regular workflow and troubleshooting are explained in this separate document, they usually involved side endpoints, setting edition or manual processing via the Bitgo console client. For example, the process of updating the exodus address can be found in this section.

- **General Considerations**

Describes things that need to be aware of when facing some problems in the backend.

- **Funds are Stucked in an Address**

How funds can be stuck in an address and what can be done to solve them.

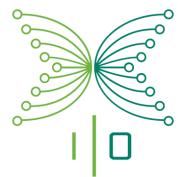
- **Bitgo API Token Max Spend Exhausted**

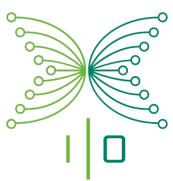
Bitgo allows a certain amount of total BTC to be sent. This sections explains how to solve this issue.

- **Exodus Address Needs to be Changed**

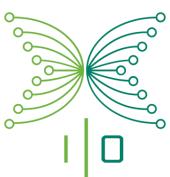
How to change the exodus address when needed.

Frontend - Business Logic

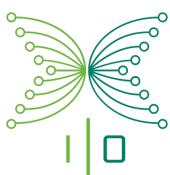




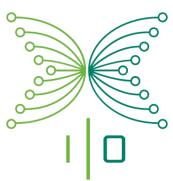
Actors	6
Internal	6
Head Compliance Officer	6
Bank Transfer Management	6
Head Invoice Manager	6
Bank Manager	6
Bank Checker	6
Exporter	7
Compliance Approval	7
Compliance Officer	7
Chief Compliance Officer	7
Customer Service	7
Investigator	8
Admin	8
Sysop	8
External	8
Partner	8
Distributor	9
Buyer	9
Supported Languages	10
Account Creation	11
Direct Database	11
Admin	11
Chief Compliance Officer	13
Head Compliance Officer	15
Investigator	17
Bank Manager	19
Bank Checker	21
Exporter	23
Head Invoice Manager	25
Sysop	27
Admin Dashboard	28
Compliance Officer	28
Customer Service	29
Enrollment	29
Partner	29



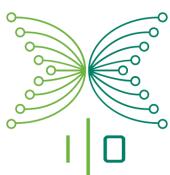
Distributor	29
Buyer	29
Enrollment	30
Generate Links	30
Generate Partner Link	30
Generate Distributor Link	31
Generate Buyer Link	32
View Enrollment Links	33
Enroll	34
Sign Up	38
Log On	39
Approval	40
Filter Users	40
Review User	41
Approve or Reject User	47
Send to Chief Compliance Officer	48
Invoice Tickets	50
Reorder	50
Reorder ADAs from Login Page	50
Reorder ADAs from Buyers Dashboard	52
Compliance Standard	53
Tiers A-D	53
Tier D+	54
Cancel Order and Expiration	54
Waiting for Sale to Start	55
Forced ADA Reservation	56
Bank Tickets	57
Value Calculations	57
Receiving the Invoice	58
Paying the Invoice	58
Bank Ticket Management	59
Entering Bank Transfer Information	60
Double Check Transfer Information	61
Bundle Bank Tickets	62
Transfer Funds to the Holding Wallet	65
Btc Tickets	65



Value Calculations	65
Receiving the Invoice	66
Paying the Invoice	66
Receiving Ada	67
Invalid Funds Received	67
Btc Orders	69
Receipt Sent	71
Distributor's Commissions	71
Monitor the Sale	72
Start the Sale	73
Check the Status of the Sale	74
User Support	74
Find User as Customer Service	75
Searching by the Exact Email Address	75
“Fuzzy” Email Search	75
Send to Head Compliance Officer	77
Find User as Investigator	77
Put a User Under Investigation	79
System Operation	80
Monitor Sale	80
Monitor Workers	81
Self Administration by External Actors	83
Password Reset	83
Change Email Address	84
Once email address change is confirmed, all buyer's orders will regenerate their Add Passcode, and a new email will be delivered to the new address with the new invoice Passcodes.	88
Change Btc Address	88
Resend Ada Passcodes	89
Misc	90
Log In	90
Logging in as Internal Actor	91
Logging in as External Actor	91
Bitcoin Pricing	93
Value Calculations Graphs Reference	93



Emails	93
Email templates	94
Email Naming and Numbering Conventions	98
EU - User Actions	100
EU1 - Non-Logged in User's Actions	100
EU2 - Compliance Officer Actions	100
EU3 - Distributor Actions	100
EU5 - ADA Pass-Code Regeneration	101
ET - Invoice Ticket States	102
ET1 - Non-Reserved Invoice Ticket States	102
ET2 - Reserved Invoice Ticket States	102
ET3 - Paid Invoice Ticket States	103



Actors

Internal

Head Compliance Officer

The Head Compliance Officer (from now on HCO) is the highest in hierarchy amongst the compliance officers. Not only does HCO have access to the personal information of the user, but the HCO can also edit almost all of the personal information except for emails.

The HCO can only see users that have been referred by Customer Service officers.

Bank Transfer Management

Head Invoice Manager

The Head Invoice Manager (HIM) is the highest in the hierarchy amongst the invoice officers. The HIM is able to perform every single operation inside the Invoice Management cycle.

The HIM has access to every action that the Bank Checker, Bank Manager and Exporter performs, and the HIM also has access to some special actions, which due to their sensitiveness, they cannot be performed by the roles mentioned before, and must be handled by a specialist in the subject.

The HIM can see all the invoices that have received an invalid amount of funds, and can either refund the invoice to the buyer, or mark it as approved so it moves on with the regular invoice flow.

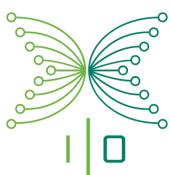
The HIM can also see all invoices awaiting for the sale to start, and force them to start the sale process.

Bank Manager

The Bank Manager is the responsible for notifying the Sales Application that there's been a Bank Transfer for a specific invoice, stating the bank which has received the transfer, and the amount in JPY that has been paid.

Bank Checker

The Bank Checker is the person responsible for confirming the information that the Bank Manager has provided, related to the Bank Transfer Payment.



Exporter

The Exporter is responsible for adding invoices to a bundle. Once they have enough invoices in the bundle, they will export it so then they can trade that USD amount in the bank for BTC in an Exchange Office. These BTC are transferred to the Holding Wallet address previously assigned to this Bundle, where it's split between every buyer's Invoice Ticket to continue the regular bitcoin payment process.

Compliance Approval

Compliance Officer

A Compliance Officer (CO) is the person in charge of reviewing users, checking whether a user is allowed to use the application or not.

Any time a buyer or a distributor enrolls, or if a buyer applies for a new tier (see buyer section), they are put into a 'PENDING' state. COs will see all users in that pending state, and has to review them in order to APPROVE or REJECT them, this action will either allow or ban them from the application.

If the CO cannot decide if that user should be approved or rejected from the application due to the complexity of the case or any other reason, they can send that user to the Chief Compliance Officer, which is higher in hierarchy.

Once the CO sends a user to the CCO, they are no longer able to APPROVE or REJECT that user.

Chief Compliance Officer

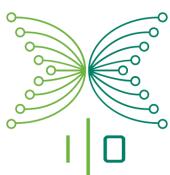
A Chief Compliance Officer (CCO) is an officer, who is higher in hierarchy than a regular Compliance Officer. Whenever there's an unreviewed user that CO cannot decide whether they should be approved or banned from the application, they send that user to the CCO queue.

Functionality-wise, CCOs use the same queue page as COs, but they only see users that have been sent to CCOs in that queue.

Customer Service

The CS officer is a role that caters to customers calling in. The officer will search for that customer based on their exact email-address to do specific actions. These actions include:

- Adding new personal files
- Leaving notes for the HCO so they can do a more specific action (changing personal information)
- Sending that user to the HCO for further action.



Investigator

The Investigator operates independently from the whole sales process. They proactively search for suspicious users or get hints from other officers of the system. When they find a suspicious user, they will investigate this user and note what has been suspicious. They also have direct contact with clients to warn them about any problems they have.

They personally(in person) contact the Customer Service who sends that user to the HCO, which in turn takes action on that user.

Admin

The admin is the most important role in the system. Admin can create all the partners, adds Compliance Officers and Customer Service Officers accounts. Admin also has access to the sale overview and sales simulator.

Sysop

System Operator is a role created to monitor over live application status from a technical perspective. This person has no access to the selling process itself, but only to the workers and some key parameters that allows the aforementioned to inspect the application's health.

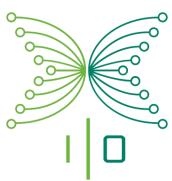
There are basically two sections a sysop user has access to:

- **SALE OVERVIEW** is a section where - like an admin - they can watch over the basic sales settings (*current Tranche, sale status, amounts, etc*) and simulate a sale scenario with the Simulator.
- **JOB MONITOR** is a section where the application's workers are displayed to check on each status. This means, if a Job has stalled, the sysop can check it directly without needing DB access.

External

Partner

Partner is a special type of distributor, that can only be invited by admin. Partners are at the very top of the distributor hierarchy and are getting the highest profits from ADA sales. They can also invite distributors ranging from tier 1 and below. In the code, it is sometimes referenced as tier 0.



Distributor

Distributor is a role that is in charge of recruiting other distributors and looking for potential ADA buyers. Distributors are divided in 3 tiers. Tier 1 is the highest and tier 3 is the lowest. It is only possible to invite distributors of a lower tier and for every buyer they or their invited distributors enroll, they will get a commission, which is a percentage of the purchase amount, in their btc wallet.

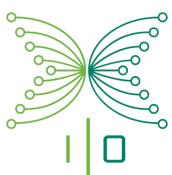
Buyer

Buyer is a user who purchases ADA. They are recruited by a Distributor or Partner. They can order ADA with the initial enrollment and by reordering. Once enrolled, they will be sent to Compliance Officers to be rejected or approved. When the account has been approved, they have the ability to

- Change their email address.
- Change account password.
- Look up on their purchase history.
- Regenerate ADA passcodes
- Reorder ADA.

Each of these processes are described in the section below.

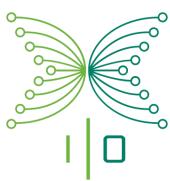
They will also be sent to Compliance Officer when exceeding the amount of ADA purchase limit of their Compliance Tier. In order to be approved, they will need to submit an application form required by the Compliance Officers.



Supported Languages

There are 4 different languages supported namely; Japanese, English, Korean and Chinese. However Japanese and English are the only languages supported for officers, while the sysop only has the English language available to them.

Role	Language available
Admin	English, Japanese
Bank Checker	English, Japanese
Bank Manager	English, Japanese
Buyer	English, Japanese, Korean, Chinese
Chief Compliance Officer	English, Japanese
Compliance Officer	English, Japanese
Customer Service	English, Japanese
Distributor	English, Japanese, Korean, Chinese
Exporter	English, Japanese
Head Compliance Officer	English, Japanese
Head Invoice Manager	English, Japanese
Investigator	English, Japanese
Partner	English, Japanese, Korean, Chinese
Sysop	English
Pages without a role (ex: login page)	English, Japanese, Korean, Chinese



Account Creation

This section will explain how to create certain types of accounts. These are split into 3 separate sections with each their own way of how to create them.

Direct Database

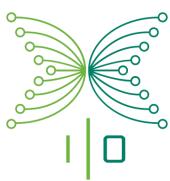
The accounts in this section have to be manually inserted into the database. The application itself can't create these users. You will need a tool that connects to the database to insert these.

To create your own password you'll need a "bcrypt" hash generator and replace the hash inside "services -> password -> bcrypt" with your own.

Admin

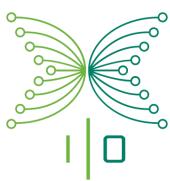
To add an Admin, use the following code as an example:

```
var admin = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+admin@iohk.io"
    }
  ],
  "roles" : [
    "admin"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqgq"
    }
  },
}
```



```
"fuzzySearchEmails" : [
  {
    "normalized" : "testemail@iohk.io",
    "alias" : "admin"
  }
],
"personalInformation" : {
  "postaddress" : {},
  "name" : "Thomas",
  "surname" : "Lastname",
  "oldOriginatorIds" : [ ],
  "language" : "ja"
},
"oldEmails" : [
  {
    "value" : null,
    "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
  },
  {
    "value" : "test-email+chiefcompliance@iohk.io",
    "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
  }
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};

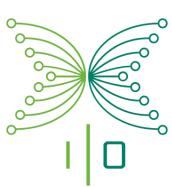
db.users.insert(admin);
```



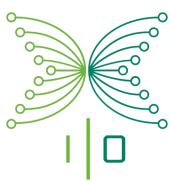
Chief Compliance Officer

To add a CCO, use the following code as an example:

```
var cco = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+chiefcompliance@iohk.io"
    }
  ],
  "roles" : [
    "chiefcompliance"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqq"
    }
  },
  "fuzzySearchEmails" : [
    {
      "normalized" : "testemail@iohk.io",
      "alias" : "chiefcompliance"
    }
  ],
  "personalInformation" : {
    "postaddress" : {},
    "name" : "Thomas",
    "surname" : "Lastname",
    "oldOriginatorIds" : [ ],
  }
}
```



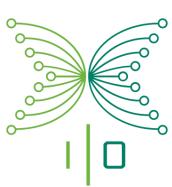
```
"language" : "ja"
},
"oldEmails" : [
{
  "value" : null,
  "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
},
{
  "value" : "test-email+chiefcompliance@iohk.io",
  "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
}
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};
db.users.insert(cco);
```



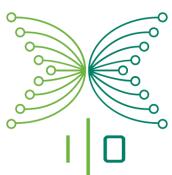
Head Compliance Officer

To insert a HCO, use the following code as an example:

```
var hco = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+hco@iohk.io"
    }
  ],
  "roles" : [
    "headCompliance"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqgq"
    }
  },
  "fuzzySearchEmails" : [
    {
      "normalized" : "testemail@iohk.io",
      "alias" : "hco"
    }
  ],
  "personalInformation" : {
    "postaddress" : {},
    "name" : "Thomas",
    "surname" : "Lastname",
    "oldOriginatorIds" : [ ],
  }
}
```



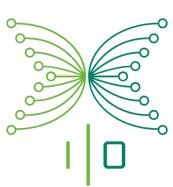
```
"language" : "ja"
},
"oldEmails" : [
{
  "value" : null,
  "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
},
{
  "value" : "test-email+chiefcompliance@iohk.io",
  "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
}
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};
db.users.insert(hco);
```



Investigator

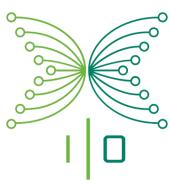
To insert a Investigator, use the following code as an example:

```
var investigator = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+investigator@iohk.io"
    }
  ],
  "roles" : [
    "investigator"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqgq"
    }
  },
  "fuzzySearchEmails" : [
    {
      "normalized" : "testemail@iohk.io",
      "alias" : "investigator"
    }
  ],
  "personalInformation" : {
    "postaddress" : {},
    "name" : "Investigator",
    "surname" : "Service",
    "oldOriginatorIds" : [ ],
    "language" : "ja"
  },
}
```



```
"oldEmails" : [
  {
    "value" : null,
    "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
  },
  {
    "value" : "test-email+ccs@iohk.io",
    "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
  }
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};

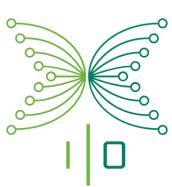
db.users.insert(investigator);
```



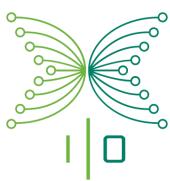
Bank Manager

To insert a Bank Manager, use the following code as an example:

```
var bankManager = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+bankManager@iohk.io"
    }
  ],
  "roles" : [
    "bankManager"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqq"
    }
  },
  "fuzzySearchEmails" : [
    {
      "normalized" : "testemail@iohk.io",
      "alias" : "bankManager"
    }
  ],
  "personalInformation" : {
    "postaddress" : {},
    "name" : "Thomas",
    "surname" : "Lastname",
    "oldOriginatorIds" : [ ],
  }
}
```



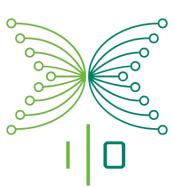
```
"language" : "ja"
},
"oldEmails" : [
{
  "value" : null,
  "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
},
{
  "value" : "test-email+bankManager@iohk.io",
  "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
}
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};
db.users.insert(bankManager);
```



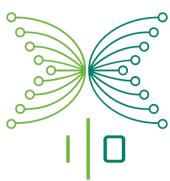
Bank Checker

To insert a Bank Checker, use the following code as an example:

```
var bankChecker = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+bankChecker@iohk.io"
    }
  ],
  "roles" : [
    "bankChecker"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqgq"
    }
  },
  "fuzzySearchEmails" : [
    {
      "normalized" : "testemail@iohk.io",
      "alias" : "bankChecker"
    }
  ],
  "personalInformation" : {
    "postaddress" : {},
    "name" : "Thomas",
    "surname" : "Lastname",
    "oldOriginatorIds" : [ ],
  }
}
```



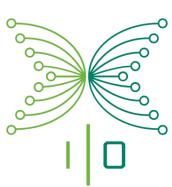
```
"language" : "ja"
},
"oldEmails" : [
{
  "value" : null,
  "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
},
{
  "value" : "test-email+bankChecker@iohk.io",
  "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
}
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};
db.users.insert(bankChecker);
```



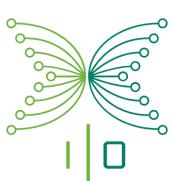
Exporter

To insert an Exporter, use the following code as an example:

```
var exporter = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+exporter@iohk.io"
    }
  ],
  "roles" : [
    "exporter"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqqq"
    }
  },
  "fuzzySearchEmails" : [
    {
      "normalized" : "testemail@iohk.io",
      "alias" : "exporter"
    }
  ],
  "personalInformation" : {
    "postaddress" : {},
    "name" : "Thomas",
    "surname" : "Lastname",
    "oldOriginatorIds" : [ ],
  }
}
```



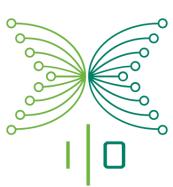
```
"language" : "ja"
},
"oldEmails" : [
{
  "value" : null,
  "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
},
{
  "value" : "test-email+chiefcompliance@iohk.io",
  "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
}
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};
db.users.insert(exporter);
```



Head Invoice Manager

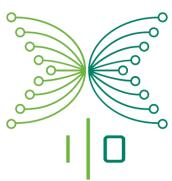
To insert a HIM, use the following code as an example:

```
var him = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+him@iohk.io"
    }
  ],
  "roles" : [
    "headInvoiceManager"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqgq"
    }
  },
  "fuzzySearchEmails" : [
    {
      "normalized" : "testemail@iohk.io",
      "alias" : "him"
    }
  ],
  "personalInformation" : {
    "postaddress" : {},
    "name" : "Thomas",
    "surname" : "Lastname",
    "oldOriginatorIds" : [ ],
    "language" : "ja"
  },
}
```



```
"oldEmails" : [
  {
    "value" : null,
    "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
  },
  {
    "value" : "test-email+him@iohk.io",
    "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
  }
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};

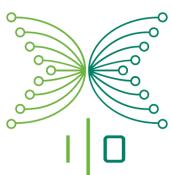
db.users.insert(him);
```



Sysop

To insert a Sysop, use the following code as an example:

```
var sysop = {
  "_id": ObjectId().valueOf(),
  "changelog" : [ ],
  "emails" : [
    {
      "verified" : true,
      "address" : "test-email+sysop@iohk.io"
    }
  ],
  "roles" : [
    "sysop"
  ],
  "createdAt" : ISODate("2015-09-13T08:56:29.324Z"),
  "services" : {
    "password" : {
      "bcrypt" : "$2a$10$sjIsfdD28n8JbE.zwNpv9.tRmsB6aVTNAYuqKPBgNhNtUOIhUlqgq"
    }
  },
  "personalInformation" : {
    "postaddress" : {},
    "name" : "Sysop",
    "surname" : "Sysop",
    "oldOriginatorIds" : [ ],
    "language" : "ja"
  },
  "oldEmails" : [
    {
      "value" : null,
      "changedAt" : ISODate("2016-01-14T11:32:07.717Z")
    }
  ]
}
```



```
},
{
  "value" : "test-email+sysop@iohk.io",
  "changedAt" : ISODate("2015-09-13T08:56:29.324Z")
}
],
"updatedAt" : ISODate("2016-09-26T01:38:09.551Z")
};
db.users.insert(sysop);
```

Admin Dashboard

These accounts are created by logging in as an admin and using the admin interface to create them.

Compliance Officer

In order to create a CO, an Administrator must login and select the option “Add Administrator” in the main dashboard menu. Then a form will be displayed, where the Name, Email and Password must be filled in.

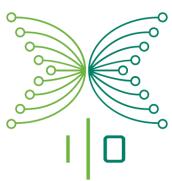
Once the button “Add Administrator” is clicked the user is created.

The screenshot shows the Admin Dashboard interface. On the left, there is a sidebar with the following menu items:

- attain admin
- test-email+admin@ionik.io
- SALE OVERVIEW
- GENERATE PARTNER LINK
- VIEW PARTNER LINKS
- ADD ADMINISTRATOR** (highlighted with a red arrow)
- ADD CS OFFICER
- BUYERS
- LOGOUT

On the right, a modal window titled "Add Compliance Administrator" is displayed. It contains three input fields: "Name", "Email Address", and "Password". Below the fields is a blue "Add Administrator" button. A red border surrounds the entire modal window.

All the other compliance related roles must be created through the Database.



Customer Service

In order to create a CS officer, an Administrator must login and select the option “Add CS Officer” in the main dashboard menu. Then a form will be displayed, where the Name, Email and Password must be filled in.

Once the button “Add user” is clicked the user is created.

The screenshot shows the Sales App dashboard. On the left, there is a sidebar with the following options: Admin, test-email-admin@lohk.io, SALE OVERVIEW, GENERATE PARTNER LINK, VIEW PARTNER LINKS, ADD ADMINISTRATOR, ADD CS OFFICER (which has a red arrow pointing to it), BUYERS, and LOGOUT. At the bottom of the sidebar is the Attain Corporation logo. The main area of the screen displays a modal window titled "Add Customer Service Officer". The modal contains three input fields: "Name", "Email Address", and "Password", each with a corresponding text input box. Below these fields is a blue "Add user" button. The entire modal is enclosed in a red border.

Enrollment

Enrolling is the only way external actors are able to enter the system. There are 3 different kinds of external actors who are able to enroll namely; Partner, Distributor and Buyer.

Partner

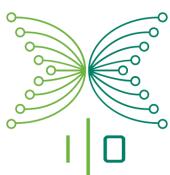
To enroll as a partner, you have to be invited by the Admin, this is the only way to get partnership.

Distributor

To enroll as a distributor, you will have to be invited by a partner or another distributor.

Buyer

To enroll as a buyer, you will have to be invited by a partner or another distributor.



Enrollment

Generate Links

Enrollment links can be generated by the Administrator, Distributor and Partner. Each of those roles have an option in their menu that generates an enrollment link to share with others.

To create a link choose one of the following menu items;

- *Generate Partner Link*
- *Generate Distributor Link*
- *Generate Buyer Link.*

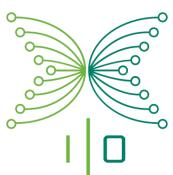
Generate Partner Link

When generating a partner link, you have the option to fill in the name and comments belonging to that link(these are optional). This link will only work once.

This can only be done by the Admin.

The screenshot shows the Sales App interface. On the left, there is a sidebar with a dark blue background. It displays the user's name "Admin" and email "test-email+admin@lohk.io". Below this are sections for "SALE OVERVIEW", "GENERATE PARTNER LINK" (which is highlighted with a red arrow pointing to it), "VIEW PARTNER LINKS", "ADD ADMINISTRATOR", "ADD CS OFFICER", "BUYERS", and "LOGOUT". At the bottom of the sidebar is the Attain Corporation logo. The main content area has a light gray background. A modal window titled "Generate Link" is open. It contains fields for "Link Name" and "Notes", both of which are currently empty. Below these fields is a section labeled "Link" with a blue border, containing the URL "http://localhost:3000/enroll/re4b36d3c1a0a48bab33". At the bottom of the modal is a blue "Generate Link" button. A red box surrounds the entire "Generate Link" modal.

After pressing “Generate Link”, the link will be displayed (Section A) so that the user can copy it, and share it to their enrollment contact.



Generate Distributor Link

When generating a distributor link, you can choose what tier that distributor will fall under which needs to be greater than your own; for example, a Partner can create Distributors of any tier, while a tier two Distributor would only be able to generate tier three or four links.

The One time link will only work once, meaning that it expires once it's used. The unlimited option will stay active as long you don't expire the link manually.

You can give the link a name as well to distinguish it from other links. The comment can be used for any extra information on this link.

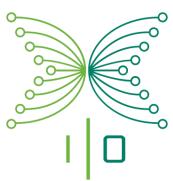
This can only be done by Distributors and Partners.

The screenshot shows the user interface of the Sales App. On the left, there is a sidebar with the following menu items:

- Howard Upton (User Name)
- test-email+Upton.Howard91@iohk.io (Email)
- MY ACCOUNT
- GENERATE DISTRIBUTOR LINK** (highlighted with a red arrow)
- GENERATE BUYER LINK
- BUYERS
- COMMISSIONS
- VIEW MY LINKS
- LOGOUT

On the right, a modal window titled "Generate Link" is displayed. This window contains the following fields:

- Tier Selection: Radio buttons for Tier 1, Tier 2, and Tier 3. Tier 1 is selected.
- Link Type: Radio buttons for One time and Unlimited. One time is selected.
- Link Name: An input field for the name of the link.
- Notes: A text area for additional comments.
- Generate Link: A blue button at the bottom of the modal.



Generate Buyer Link

When generating a buyer link, you have the option to select the type of link you want to create. The One time link will only work once expires. The unlimited option will stay active as long as you don't expire the link manually.

You can give the link a name as well, to distinguish it from other links. The comment can be used for any extra information on this link.

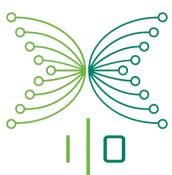
This can only be done by Distributors and Partners.

The screenshot shows the Sales App interface. On the left, there is a sidebar with the following menu items:

- OConnor Hayfa (User Name)
- test-email+Hayfa.OConnor18@iohk.io (Email)
- MY ACCOUNT
- GENERATE DISTRIBUTOR LINK
- GENERATE BUYER LINK** (highlighted with a red arrow pointing to the right)
- BUYERS
- COMMISSIONS
- VIEW MY LINKS
- LOGOUT

On the right, a modal window titled "Generate Link" is open. It contains the following fields:

- One time
- Unlimited
- Link Name: [Empty input field]
- Notes: [Empty input field]
- Generate Link** (Blue button)



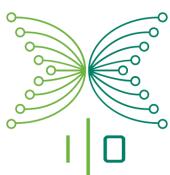
View Enrollment Links

To view the enrollment links created before, you can choose one of the following options on the side menu; View partner links or View my links. This page may look slightly different for partners.

The play icon expands the information and shows the created date and notes made on the link.

As viewed below, the different columns represent the different enrollment links. The distributor links on the left and the buyer on the right.

Link	Invite	Link
a4ea5	http://localhost:3000/enroll/r4971269893e9f9a	6ed7f
7dcdd	http://localhost:3000/enroll/rf4c802aafd007ee	fdd48
abbe7	http://localhost:3000/enroll/r37d3fea1fab2e25	



Enroll

To enroll, you have to follow certain steps, the enrollment starts with an url provided by a partner or distributor. After following the url, you will start at step 1 of the enrollment process.

Step 1:

The first step is selecting the *Language* and *Country of Residence*. The language selected here will be the language the interface will be in after creating your account as well.

For Vietnamese users, there will be a checkbox they are required to check. This checkbox makes sure the user understands that the agreements they sign will not be available in Vietnamese, but in English only. After selecting, click next for Step 2.

Please choose your residence

STEP 1 STEP 2 STEP 3 STEP 4

言語/ 사용 언어/ 使用語言/Your Language

English

居住国/거주 국가/居住地/Your Country of Residence

Japan

Next

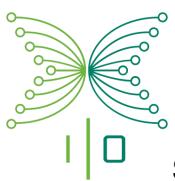
Step 2:

Here the enrollee has to provide their personal information. First they have to choose their account type; Personal or Corporation. If a corporation is chosen, the field “Corporation Name” will become available.

To verify the *Email Address*, a code will be send as soon as the user fills in the email and click “Receive Code”. After opening the email and copy and pasting the code in the box and clicking confirm code, the email address will be verified.

If entered *Zip Code* can't be automatically verified, a checkbox will show, having the user verify it manually.

If the enrollee is a buyer, option to input the amount of ADA (in USD) is available along with the payment options.



The user has to accept the policy at the bottom of the page. The policy file and language are defined by the settings file.

The image below shows the process for a distributor or partner.

Please fill out the enrollment form

STEP 1 STEP 2 STEP 3 STEP 4

Account Type <input type="text" value="Individual"/>	Zip Code <input type="text" value="1234"/> <input checked="" type="checkbox"/> Is this zip code correct?
Last Name <input type="text" value="Doe"/>	State <input type="text" value="1234"/>
First Name <input type="text" value="John"/>	City <input type="text" value="1234"/>
Email Address <input type="text" value="john+doe@asdasd.com"/>	Address <input type="text" value="1234"/>
Email Verification Code <input type="text" value="26cb6575f5fe40ec031"/>	Receive code Confirm code
Phone Number <input type="text" value="1234"/>	
Birthdate <input type="text" value="2017-01-31"/>	
Privacy Policy 日本語	
<input checked="" type="checkbox"/> I agree to the Privacy Policy	
Back Next	

Frontend
Business Logic

The image below shows the process for a buyer.

Please fill out the enrollment form

STEP 1 STEP 2 STEP 3 STEP 4

Account Type

Last Name

First Name

Email Address

Email Verification Code
 Receive code Confirm code

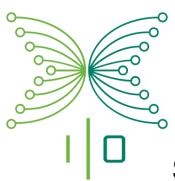
Phone Number

Birthdate

[Privacy Policy 日本語](#)

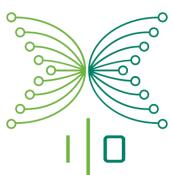
I agree to the Privacy Policy

Back Next



Step 3:

In this step, the user has to accept any policies listed before continuing on to the next step.



Step 4:

The user can provide legal documents here proving their existence and legitimacy. There are different kinds of documents which can be attached; *National Health Insurance*, *Driver's Licence* and *Passport* are the default choices. If the document provided doesn't match any of the categories, "Other" can be selected instead.

The user has to agree that no "My Number" document has been provided, as this is illegal in Japan.

Please upload a copy of your ID

STEP 1 STEP 2 STEP 3 STEP 4

Please upload a copy of your ID

Do not upload your My Number document

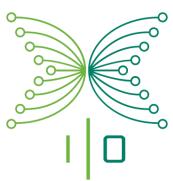
Select the type of your document.

National health insurance

ID Document

No files uploaded yet!

Check to confirm that you are not uploading a My Number document



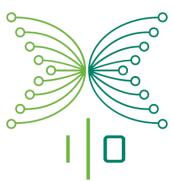
Sign Up

After you complete the enrollment form, you will receive a signup link in your email address (This only happens if you are a distributor. If you are a buyer, you'll have to reset your password in order to login). When you follow that link, a form will be displayed asking you to set a password, which you will need to log in to the application.

Create Password

Please set a password to log in:

Re-enter Password:



Log On

After you have an approved account, you can go to the login page and log on into the application. In the Section A you can complete the fields with your email and password and press the Login button.

If you already have a valid account but forgot your password, you can click the “Forgot password” link (Section B).

A

Welcome Back

Login

English

EMAIL ADDRESS

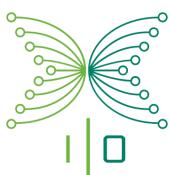
PASSWORD

Reorder

B

Forgot password?

Inquiry support@attaincorp.co.jp



Approval

Filter Users

Compliance Queue allows a user to search through different filters. In this section, these filters will be explained. For that, we decided to split them into different categories, according to the next screenshot.

Attain Dashboard LOGOUT

A

Buyer Distributor
Filter Corporation Individual
 (A) (B) (C) (D+)

B

Recent (0) Unreviewed (43) Watching (3)
Funds Received (0) CCO (5) All (1643)

Countries: Japan, South Korea, Others

D

From To

Surname Search

Surname Name Email

E

Results found: 47

Progress	Surname	Name	Email	Country of Residence	Account Type	Compliance Level (Old/New)	Indicators	Current Reviewer
■	中本	悟志	test-email+Odysseus.Fletcher5715@iohk.io	Japan	Individual, Buyer	- / B		<button>Review</button>
■	中本	悟志	test-email+Ella.Lancaster6795@iohk.io	Japan	Individual, Buyer	- / A		<button>Review</button>
■	中本	悟志	test-email+Tara.Terrell8023@iohk.io	Japan	Individual, Buyer	- / A		<button>Review</button>

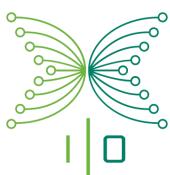
Section A:

These are togglable filters which means they can be turned on or off at will. These filters will return buyers (either corporation or individual) or distributors from the corresponding tier (A,B,C,D+).

Section B:

This contains the tabs on top center. This allows the CO to navigate between

- **Recent:** Users that have entered the compliance queue in the last week.
- **Unreviewed:** Users in pending state.
- **Watching:** Users that are being watched by the current CO.
- **Funds Received:** Users that have paid while being in pending state.
- **CCO:** Users sent to CCO, this only applies for the COO user.
- **All:** Every buyer and distributor in the application in pending state.

**Section C:**

Country filters allows to search users whose residence country is Japan, South Korea or Others.

Section D:

Date filters allows a CO to filter users who have entered compliance queue in a date range.

Section E:

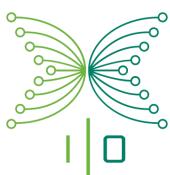
The search filter allows a CO to submit a search by Name, Surname or Email.

Review User

After you search for a user, a list of them will be displayed with some basic information of that user. If you want to see that user's information in more detail, you click the "review" button which will take you to the user summary page.

The screenshot shows the 'Account Approval Workflow' section of the Attain Dashboard. At the top, there are several filter buttons: 'Recent (7)', 'Unreviewed (29)', 'Watching (0)', and date range pickers for 'From' and 'To'. Below these are buttons for 'Funds Received (0)', 'CCO (0)', and 'All (30)'. A search bar includes a dropdown for 'Surname' and a 'Search' button. Underneath the filters, there are checkboxes for 'Countries': 'Japan' (checked), 'South Korea' (checked), and 'Others' (checked). A red arrow points from the text 'Current Reviewer' to a 'Review' button for the first user in the list. The table below shows results for 839 users, with columns for Progress, Surname, Name, Email, Country of Residence, Account Type, Compliance Level, Indicators, and Current Reviewer.

Progress	Surname	Name	Email	Country of Residence	Account Type	Compliance Level (Old/New)	Indicators	Current Reviewer
Green	Cobb	Whilemina	test-email+Whilemina.Cobb75@iohk.io		Individual, Distributor	- / C	Commission Wallet	Review
Green	Gonzalez	Kaden	test-email+Kaden.Gonzalez91@iohk.io		Individual, Buyer	- / C		Review
Green	Kirk	Justina	test-email+Justina.Kirk4@iohk.io		Individual, Buyer	- / A		Review



Review user page is divided in two main components, one that shows all the information about that user, and the second part that shows other users with the same birthday.

The section A displays a series of buttons:

- **Back:** Go to the previous page.
- **Approve:** Approve the user.
- **Reject:** Rejects the user.
- **Watch:** Keep track of that user under the “watching” tab in the search page.
- **Send to CCO:** Send the user to CCO.

Section B shows all the user's information to check, and section C displays the images that the user uploaded to the web. If you click on one of these images it will be displayed in the center of the page.

A: Buttons

B: User Information Table

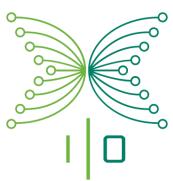
C: File Upload

ID	pf9Z2crcP7MyXkx7i
Account Type	Individual, Distributor
Name	Cobb Whilemina
Email	test-email+Whilemina.Cobb75@iohk.io
Status	Pending
Is the user under investigation?	
Birthdate	09/25/1947
Distributor level	Tier 1
Branch Partner	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Referred By	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Enrolled At	2月 7日 2016
Enrollment IP	
Phone	2055484514
Language	
Residence Country	

No document selected

Choose a file

Health Insurance Card



Sales App | Frontend - Business logic

You can rotate the image and also zoom in and out by scrolling with the mouse. If the image is inappropriate, you can delete it.

Attain Dashboard LOGOUT

Choose a file

Approve **Reject** **Watch** **Send to CCO**

ID	pf9Z2crcP7MyXkx7i
Account Type	Individual, Distributor
Name	Cobb Whilemina
Email	test-email+Whilemina.Cobb75@iohk.io
Status	Pending
Is the user under investigation?	
Birthdate	09/25/1947
Distributor level	Tier 1
Branch Partner	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Referred By	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Enrolled At	2月 7日 2016
Enrollment IP	
Phone	2055484514
Language	
Residence Country	

Delete **Rotate**

Health Insurance Card

Enrollee Name: **FIRST M LAST NAME**

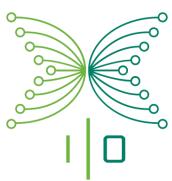
Enrollee ID: **2234567890**

Issuer: **987654321**

RxBIN: **444444** RxGrp: **55555555**

Enrollment Date: **01/01/2010**

This health plan is provided by ABC Health Insurance. While coverage remains in force, members are entitled to the benefits under the terms and conditions of the plan. This card is for identification purposes only and is not a guarantee of coverage. Definitions and conditions may apply. Call 800-222-1233 to talk to a ABC Health Insurance representative.



At the end of the user information, there is a section to add comments about that user, there are three types of comments:

- **Nikkei Telecom:** Nikkei Telecom is a company used by Attain to make a further analysis of the user. Any analysis they made is written here.
 - **Calls:** Logs that Compliance Officer takes when they call a user over telephone.
 - **Comment:** Comments for the Compliance Officer to leave a personal reminder, so they are the only ones who will be able to see it
- The Nikkei and Calls comments can be seen by any Compliance Officer.

Attain Dashboard

LOGOUT

Did user pass API zip check? This user has not done the check

Progress
 Reviewed

Nikkei Telecom

Nikkei Telecom Comment

Add comment

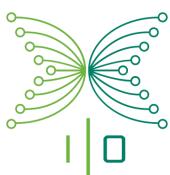
Calls

Call Comment

Add Call

Great Person

Save



At the bottom of the page, you will find the “Users with similar Birthday” section (Section A), this section is used to find users with duplicated accounts. A list of different users will be displayed and if you want to compare the information between them you can do click on the down arrow (Section B).

Attain Dashboard

LOGOUT

Add Call

Great Person

Save

A

Users with Same Birthday

Results found: 1

Progress	Surname	Name	Email	Country of Residence	Account Type	Compliance Level (Old/New)	Indicators	Current Reviewer
Green	Patton	Joel	test-email+Joel.Patton91@iohk.io		Individual, Buyer	A / -		B

B

Attain Dashboard

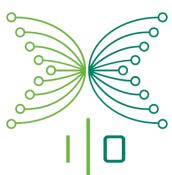
LOGOUT

Currently Reviewing

ID	pf9Z2crcP7MyXkx7i
Account Type	Individual, Distributor
Name	Cobb Whilemina
Email	test-email+Whilemina.Cobb75@iohk.io
Status	Pending
Is the user under investigation?	
Birthdate	09/25/1947
Distributor level	Tier 1
Branch Partner	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Referred By	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Enrolled At	2月 7日 2016
Enrollment IP	
Phone	2055484514
Language	
Residence Country	
Address	P.O. Box 120, 4007 Tortor. Av. Hastings, NI 123-1234
Did user pass API zip check?	This user has not done the check

User With Same Birthday

ID	33eK5mEqqCMSis5xY
Account Type	Individual, Buyer
Name	Patton Joel
Email	test-email+Joel.Patton91@iohk.io
Status	Approved
Birthdate	09/25/1947
Compliance Level (Old/New)	A / -
Branch Partner	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Referred By	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Enrolled At	3月 2日 2016
Enrollment IP	
Phone	2025004692
Language	
Residence Country	
Address	P.O. Box 669, 6731 Tristique Avenue Ede, Gelderland 123-1234
Did user pass API zip check?	This user has not done the check



When you hover over any attribute, a blue line will display where that information is for both users.

Attain Dashboard

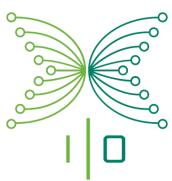
LOGOUT

Currently Reviewing

ID	pf9Z2crcP7MyXkx7i
Account Type	Individual, Distributor
Name	Cobb Whilemina
Email	test-email+Whilemina.Cobb75@iohk.io
Status	Pending
Is the user under investigation?	
Birthdate	09/25/1947
Distributor level	Tier 1
Branch Partner	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Referred By	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Enrolled At	2月 7日 2016
Enrollment IP	
Phone	2055484514
Language	
Residence Country	
Address	P.O. Box 120, 4007 Tortor. Av. Hastings, NI 123-1234
Did user pass API zip check?	This user has not done the check

User With Same Birthday

ID	33eK5mEqqCMSis5xY
Account Type	Individual, Buyer
Name	Patton Joel
Email	test-email+Joel.Patton91@iohk.io
Status	Approved
Birthdate	09/25/1947
Compliance Level (Old/New)	A / -
Branch Partner	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Referred By	Leroy Sexton <test-email+Leroy.Sexton91@iohk.io>
Enrolled At	3月 2日 2016
Enrollment IP	
Phone	2025004692
Language	
Residence Country	
Address	P.O. Box 669, 6731 Tristique Avenue Ede, Gelderland 123-1234
Did user pass API zip check?	This user has not done the check



Approve or Reject User

Once the buyer or distributor has enrolled, they will be sent to Compliance Officer(CO) and be reviewed. The CO will review the user and based on the information given and either *APPROVE* or *REJECT* them. In the case the CO is unsure, the CO can send the user to the Chief Compliance Officer who also has the option to *APPROVE* or *REJECT*. Approved users will be able to use application depending on their role. For more details about what user have access to, see the external actors section.

In more rare cases, when a user gets sent to the Head Compliance Officer, the HCO has the option to change the user's status regardless of their previous status.

Standard way of approving or rejecting by a CO or CCO.

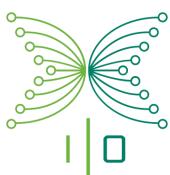
Attain Dashboard

LOGOUT

Back	Approve	Reject	Watch	Send to cco
ID	LdQCsJmptyhrASMu			
Account Type	Individual, Buyer			
Name	な子 て口			
Email	test-email+Baker.Kirkland1713@iohk.io			
Status	Pending			
Birthdate	03/18/1996			
Compliance Level (Old/New)	A / A			
USD Requested	\$1,500			
Branch Partner	んさ し雲 <test-email+Gary.Burton655@iohk.io>			
Referred By	んさ し雲 <test-email+Gary.Burton655@iohk.io>			
Enrolled At	2月 9日 2016			
Enrollment IP	111.22.333.444			
Phone	(+64-07-543-4558)			
Language	ja			
Residence Country	Japan			
Address	P.O. Box 248, 1697 Rutrum Av. Tokyo, MA 3112			
Did user pass API zip check?	This user has not done the check			

No document selected

Choose a file



For HCO, changing user's birthdate looks like the following, and demands him to enter his password.

Attain Dashboard LOGOUT

Back	Done
ID	Wcggn63wpZM5SK79jo
Account Type	Corporation, Buyer
Company Name	Good Times
Registration Date	11/10/2015
Name	逃子 蜜い
Email	test-email+Cassidy.Fry5214@iohk.io
Status	Approved
Birthdate	03/19/1954
<input type="button" value="Edit"/>	
Birthdate 1954-03-31	
アカウントを承認する	
<input type="password" value="Confirm Password"/>	
<input type="button" value="Update Birthdate"/>	
Compliance Level (Old/New)	A / -
USD Requested	\$2,245
Branch Partner	にに 波口 <test-email+Caldwell.Kidd3861@iohk.io>
Referred By	にに 波口 <test-email+Caldwell.Kidd3861@iohk.io>

No document selected Choose a file

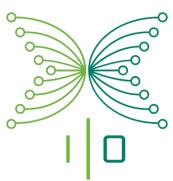
Send to Chief Compliance Officer

If there is a verification that exceeds the Compliance Officer's domain or they simply want the Chief Compliance Officer to take care of this review, the CO can send this user to the CCO. This user will be blocked for any other CO until the CCO completes the review.

Attain Dashboard LOGOUT

Back	Approve	Reject	Watch	Sent to CCO
ID	YetQkMyeZNzbtN6L2			
Account Type	Individual, Buyer			
Name	鳥日 美ビ			
Email	test-email+Joel.Bowers6391@iohk.io			
Status	Pending			
Birthdate	03/15/1998			
Compliance Level (Old/New)	- / A			
USD Requested	\$1,000			
Branch Partner	にき さ司 <test-email+Jordan.Ward7648@iohk.io>			
Referred By	にき さ司 <test-email+Jordan.Ward7648@iohk.io>			
Enrolled At	8月 7日 2016			
Enrollment IP	111.22.333.444			
Phone	08070036218			
Language	ja			
Residence Country	Japan			

No document selected Choose a file



After the “Send to CCO” button is clicked and the action confirmed, a message will indicate that the user is under CCO review, and the *Approve/Reject* options are not available.

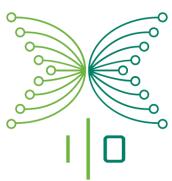
Attain Dashboard LOGOUT

[Back](#) [Watch](#)

This user is already under review by the CCO ←

ID	Yet0kMYeZNzbtN6L2
Account Type	Individual, Buyer
Name	鳥日 美ビ
Email	test-email+Joel.Bowers6391@iohk.io
Status	Pending
Birthdate	03/15/1998
Compliance Level (Old/New)	- / A
USD Requested	\$1,000
Branch Partner	にき さ司 <test-email+Jordan.Ward7648@iohk.io>
Referred By	にき さ司 <test-email+Jordan.Ward7648@iohk.io>
Enrolled At	8月 7日 2016
Enrollment IP	111.22.333.444
Phone	08070036218

No document selected Choose a file



Invoice Tickets

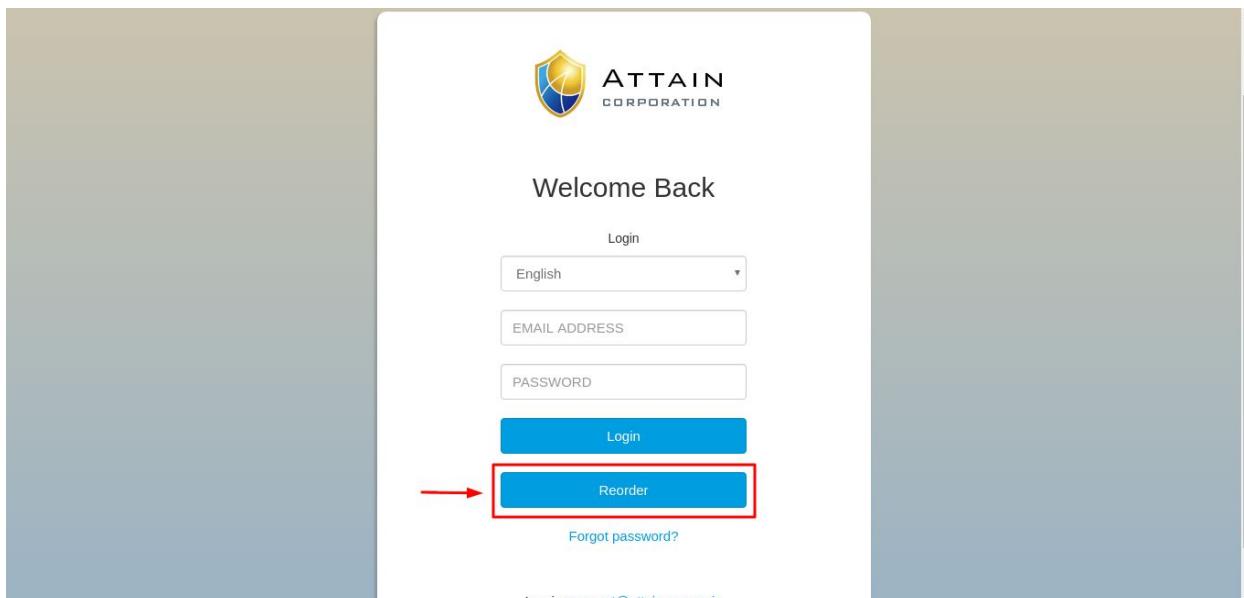
Reorder

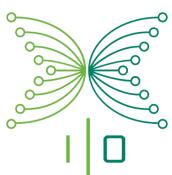
The reordering of ADA can be done in two ways:

- From the login page.
- From the reorder page (after a buyer is logged in).

Reorder ADAs from Login Page

On the login page, you will see a button to reorder. Once you click it, it will bring you to the reorder page. There you can select your language and you'll need to fill in your email which needs to be checked before you can reorder.



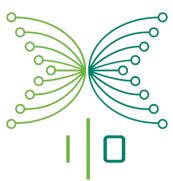


The screenshot shows a modal window titled "Reorder ADA". It contains two sections, A and B. Section A is labeled "Select Language" with a dropdown menu set to "English". Section B is labeled "Email Address" with an input field containing "kalyhav@hotmail.com". At the bottom are two buttons: "Order" (blue) and "Confirm Email Address" (green). A small letter "C" is positioned next to the green button.

Frontend
Business Logic

After your email is verified, more options will appear, the ones under Section A are the same for all the users but the “*Payment Method*” under Section B is only for users whose residence country is configured in a way that allows both bitcoin and Bank payments. When you complete all the fields and agree to the terms you can click the “Order” button to reorder ADAs.

The screenshot shows the same modal window as above, but with additional fields and sections. Section A now includes a required input field for "USD amount" with the placeholder "Please enter the USD amount you wish to exchange into ADA". Section B includes a dropdown menu for "Payment Method" with the placeholder "(Select One)". Section C contains two groups of checkboxes: "User Policy 日本語" (checkboxes for "I agree to the User Policy" and "I understand the Risks") and "Possible Risks 日本語" (checkbox for "I understand the Risks"). The "Order" and "Reset" buttons are at the bottom.

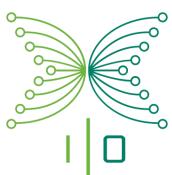


Reorder ADAs from Buyers Dashboard

The other way to reorder ADA is from the reorder page. After a buyer logs in and selects the “*Reorders*” button on the left it will take them to the reorder page.

This page is very similar to the reorder without login, but you don’t have to input your email. Just like the other reordering process, the Section A is equal for all the users, and Section B is only for the users whose residence country is configured in a way that allows both bitcoin and Bank payments.

The screenshot shows the 'Reorder ADA' page. On the left, there is a sidebar with the user's name 'Berger Leroy' and email 'test-email+Leroy.Berger95@iohk.io'. Below that are 'MY ACCOUNT', 'REORDERS' (which is highlighted with a red arrow pointing to the right), 'PURCHASES', and 'LOGOUT'. At the bottom of the sidebar is the 'ATTAIN CORPORATION' logo. The main content area has a title 'Reorder ADA' and two sections: 'A' and 'B'. Section A contains a text input field with placeholder 'Please enter the USD amount you wish to exchange into ADA'. Section B contains a dropdown menu labeled '(Select One)'. Below these sections is a red-bordered box containing 'User Policy 日本語' and 'Possible Risks 日本語'. Underneath this box is a blue 'Order' button. There are also two checkboxes: 'I agree to the User Policy' and 'I understand the Risks'.

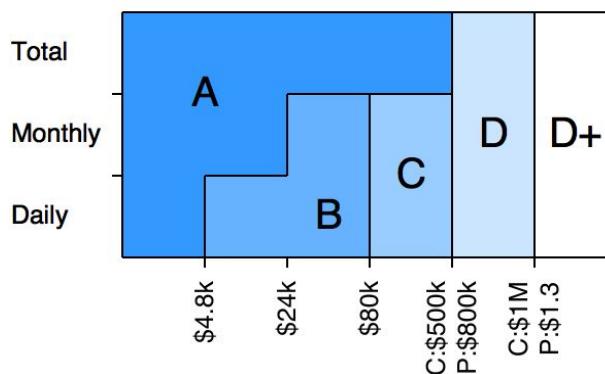


Compliance Standard

A Compliance Tier is a feature that limits the amount of Ada that a buyer can purchase. These limits are applied on the daily, monthly and total purchased quantity.

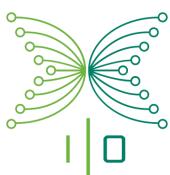
Compliance tier scales from A (lowest level) to D+ (highest level), and every time a buyer exceeds their current compliance tier limit, they must go through compliance approval again. When they are approved, they are set in the new compliance tier.

The tiers are calculated from the start of a new calendar day or the start of the new calendar month. If an order is made just before midnight and the next just after, the tier won't change unless it exceeds the monthly limit. Just as an order made on the first day of the month won't count towards the monthly limit of the previous month and won't increase the tier.



Tiers A-D

Account Type	Limit Type	USD Amount (up to)	Compliance Tier
Personal	Daily	4,800	A
	Monthly	24,000	B
		80,000	C
	Total	800,000	D
Company	Daily	4,800	A
	Monthly	24,000	B
		80,000	C
	Total	500,000	D



Tier D+

Once a buyer has reached D level, their compliance tier will no longer be recalculated, but they'll still have to go through compliance approval each time their total amount exceeds their last tier limit plus 500,000 USD. This behaviour will be repeated until the buyer has purchased a total amount of 5,800,000 USD (if the account type is Personal) or 5,000,000 USD (if the account type is Company). Once that amount has been exceeded, then they will no longer have to be approved by a Compliance Officer.

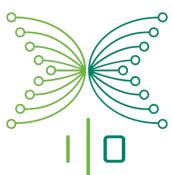
Cancel Order and Expiration

Both Bitcoin and Bank Transfer invoices have a time deadline to receive payment. If the invoice does not get paid before that deadline passes, it will be moved to an Expired state. Once that invoice has been expired, it cannot be paid. In that case, the user will have to create a new invoice. These tickets are shown in the “Expired” tab and only the Head Invoice Manager and the Bank Manager can see them.

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
黒蟹 難好	596566	Bank	L	9月 15日 2015	\$5,000	¥0	₿0	₿0
赤才 子は	546595	Bank	V	9月 16日 2015	\$1,000	¥0	₿0	₿0

If for some reason the COs consider that the invoice is suspicious (and still hasn't been paid), they have the authority to cancel it. In that state, it also cannot be paid or moved. These two states (Canceled and Expired) are considered final states, so they are dead ends in the invoice flow.

When a buyer gets rejected by the Compliance Officer, all of their unpaid invoices are automatically canceled.



Waiting for Sale to Start

When tickets are in “*waiting for sale to start*” state they are essentially paused. The tickets will not get processed and are in a queue until the next sale starts. Tickets on top of the queue will be processed first when the new sale starts.

These tickets can be seen it by Head invoice manager in the “Sale Pending” tab.

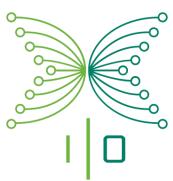
Atain Dashboard LOGOUT

Head Invoice Manager

Sale Pending (1) Unpaid Invoices (1) Confirm Bank Amount (1)
Sent to Bank Checker (1) Prepare Bundle (0) Current Bundle (2)
Bundles (1) Export (144) Expired (5622)
Invalid Funds Received (0) BTC Orders (4) Receipt Sent (7642)
Force Reserve ADA (0)

From To
Invoice Number Search

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
Hiroto Shioi	1008696	Btc	L	1月 25日 2017	\$100,000	¥0	฿0	฿0



Forced ADA Reservation

If for some reason the Head invoice manager determines that a stalled ticket can be sold, he can force the ADA reservation for that ticket. By doing this the ticket is moved to the next state and the buyer can continue the purchase process.

Attain Dashboard LOGOUT

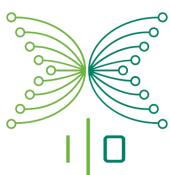
Are you sure you want to force this ticket to reserve ADA?

Cancel Confirm

Sale Pending (1)	Prepare Bundle (0)	Current Bundle (2)
Sent to Bank Checker (1)	Export (144)	Expired (5622)
Bundles (1)		
Invalid Funds Received (2)	BTC Orders (6)	Receipt Sent (7642)
Force Reserve ADA (1)		

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED	Actions
Davis Kemlers	1008699	Btc	L	1月 25 日 2017	\$99,999,999	¥0	฿0	฿0	(i)

Force reserve ADA

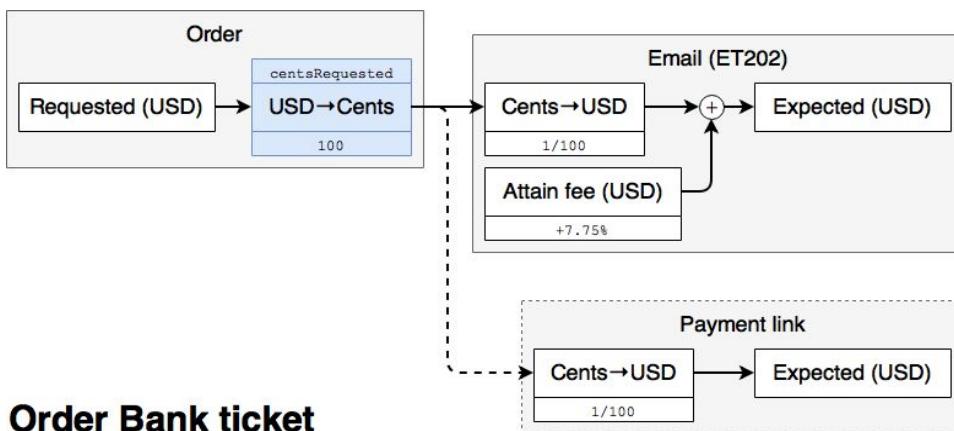


Bank Tickets

Value Calculations

When the buyer creates a Bank Invoice, he will manually enter the USD amount he wishes to convert to ADA. After he has transferred the corresponding USD (or YEN) to one of Attain's Bank accounts, the Bank Invoice Management flow starts.

In this stage, the system will calculate the expected USD, adding the Attain fee (7,75%) to the USD amount entered by the buyer.

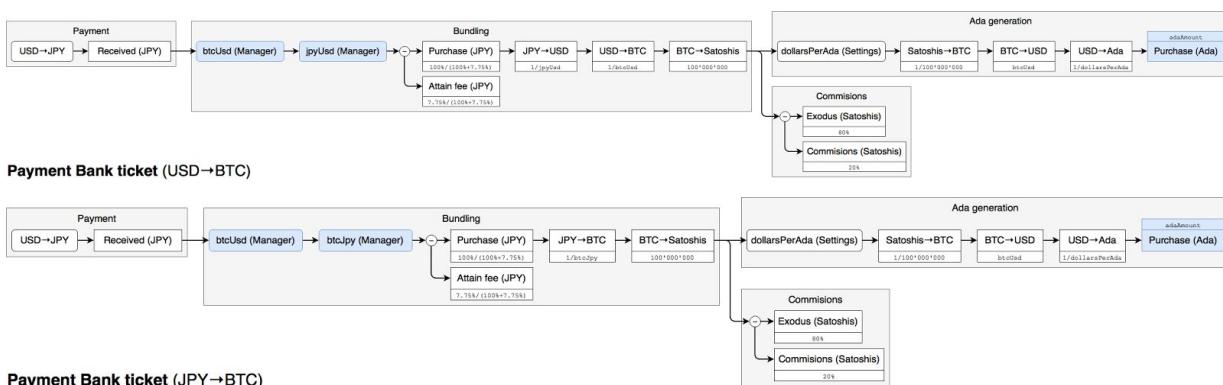


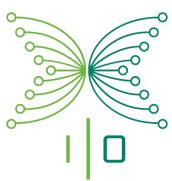
Order Bank ticket

At a certain point of the flow (see Bundle bank tickets section), that transferred money will be used to purchase BTC, along with some sibling orders bundled in one single buy.

There's a double check between the exchange rate at which the Attain's personnel (Exporter) has purchased the BTCs, and the exchange rate obtained from internet providers (Same as BTC Invoices, for more detail please address to *Misc/Bitcoin Pricing*).

The app can calculate the corresponding ADA in two different ways, according to the exchange rate at which the Exporter has purchased the BTCs.





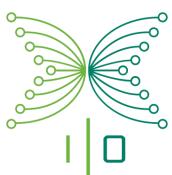
Receiving the Invoice

After a buyer creates a bank invoice, they will receive the invoice receipt in their email account with the information on how to pay that invoice. This means that the email will contain:

- The bank account where money can be transferred to in order to pay the invoice . Application supports two banks, Sumitomo Mitsui Banking Corporation and Kansai Urban Banking Corporation. Transfers can only be done to one of them.
- Amount in JPY
- A link to get the actual exchange rate from USD to JPY

Paying the Invoice

When the bank transfer is complete, the internal bank ticket management flow will begin. At this point in the process, the buyer will not have to do anything, except waiting for this step to be completed in order to receive the Ada.



Bank Ticket Management

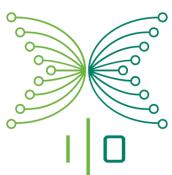
In each stage of the Bank Ticket Management, you will see that every screen has the same filters in the top right corner. These are common filters which can be used in each stage of the bank ticket flow.

The user will be able to search invoices by

- Creation date (from-to filters)
- Invoice number
- Buyer Id
- Invoice transaction Id
- Payout transaction Id
- Invoice wallet or bank (dropdown menu)

The screenshot shows the Attain Dashboard with the following interface elements:

- Header:** Attain Dashboard, LOGOUT
- Top Bar:** Prepare Bundle (1), Current Bundle (0), Bundles (2), Export (137)
- Section Title:** Exporter
- Table Headers:** NAME, INVOICE #, PAYMENT METHOD, STATE, CREATED AT, ORDERED AMOUNT, YEN REC
- Table Data:** John Doe, 1004400, Bank, ✓, 1月 17日 2017, \$1,234, ¥1,234
- Buttons:** Add to Export
- Filtering:** A dropdown menu is open, highlighted with a red box. It includes fields for From and To dates, a search bar, and a dropdown menu with options: Invoice Number, Buyer Id, Invoice Transaction Id, Payout Transaction Id, Invoice Wallet, and Bank. An arrow icon points to the Bank option.



Entering Bank Transfer Information

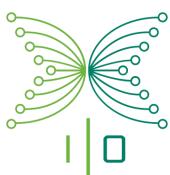
The first action that must be taken when a bank invoice is created, is to confirm the bank transfer information. This action is performed by the Bank Manager who will manually input the amount of JPY for the bank transfer, then select the bank from which the transfer was made.

Once the information has been filled in and after clicking Funds Received, the Bank Manager will see a popup to confirm that the information that was set is correct. When the Bank Manager confirms, the ticket moves on to the next stage which is the double confirmation by the Bank Checker.

The screenshot shows the Attain Dashboard with the title "Bank Manager". At the top, there are three buttons: "Unpaid Invoices (1)", "Sent to Bank Checker (0)", and "Expired (3119)". Below these are filters for "From" and "To" dates, and a search bar with a dropdown for "Invoice Number".

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
John Doe	1004400	Bank	✓	1月 17日 2017	\$1,234	¥0	฿0	฿0

At the bottom, there is a row with a currency selector (¥), a dropdown menu labeled "Select a bank", and a blue button labeled "Funds Received". A circled arrow icon is located at the top right of the table header.



In the other tabs, the Bank Manager can also see the expired invoices and the invoices that they already sent to the Bank Checker.

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
桃味 金雲	1008684	Bank	✓	1月 24日 2017	\$1,001	¥100,100	฿0	฿0

Frontend
Business Logic

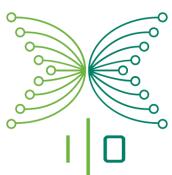
Double Check Transfer Information

The Bank Checker's job is to check if the information received from the Bank Manager is correct. The Bank Checker can either confirm the payment, or send it back to the Bank Manager, marking it as unpaid.

When the Bank Checker confirms that the invoice payment is correct, the Bank Checker will send the invoice to the Exporter, which is then responsible for creating the Bundle that will then be exchanged for Btc in order to get the Ada.

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
John Doe	1004400	Bank	✓	1月 17日 2017	\$1,234	¥1,234	฿0	฿0

Buttons at the bottom: Mark Unpaid (gray), Confirm Payment (blue).



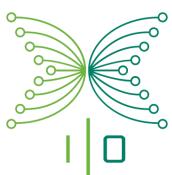
Bundle Bank Tickets

An Exporter is someone who is responsible of creating invoice bundles which they will use to trade in that bundle at an exchange for BTC, which will then be transformed into Ada.

The first step is to start adding the invoices to the current bundle. To do so, in the “*Prepare Bundle*” tab, the Exporter will have to click on the “*Add to Export*” button.

The screenshot shows the Attain Dashboard with the following interface elements:

- Header:** Attain Dashboard, LOGOUT
- Section Title:** Exporter
- Buttons:** Prepare Bundle (3), Current Bundle (0), Bundles (2), Export (137), From, To, Search (Invoice Number)
- Table Headers:** NAME, INVOICE #, PAYMENT METHOD, STATE, CREATED AT, ORDERED AMOUNT, YEN RECEIVED, SATOSHIS EXPECTED, SATOSHIS RECEIVED
- Data Rows:**
 - John Doe, 1004400, Bank, ✓, 1月 17日 2017, \$1,234, ¥1,234, \$0, \$0
 - John Doe, 1004401, Bank, ✓, 1月 17日 2017, \$1,000, ¥1,000, \$0, \$0
- Action Buttons:** Add to Export (highlighted with a red arrow), Delete (for each row), Edit (for each row)



After doing that, the invoices that were added to the bundle will be displayed in the tab “*Current Bundle*”. This is a temporary stage that allows the Exporter to add as many invoices as they want to that bundle. When the Exporter is finished adding invoices to the bundle, the button “*Bundle Payments for Export*” has to be pressed, which will then create an actual bundle where invoices can no longer be added or removed.

Attain Dashboard

EXPORTER

Prepare Bundle (1) Current Bundle (2) Bundles (2)

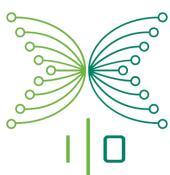
From To

Invoice Number Search

Currently you have 2 paid orders with the amount of ¥2,234 to be exported

Bundle Payments for Export

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
John Doe	1004400	Bank	✓	1月 17 日 2017	\$1,234	¥1,234	฿0	฿0
John Doe	1004401	Bank	✓	1月 17 日 2017	\$1,000	¥1,000	฿0	฿0



Sales App | Frontend - Business logic

After creating the Bundle, it will appear in the Bundles tab. Here, once the Exporter has purchased the BTC, he will have to manually set the exchange rates, either between YEN -> BTC or USD -> BTC before finalizing the payments bundle.

Note: The BTC must only be transferred to the invoice wallets after finalizing the payments bundle.

When the Bundle is finalized, it's moved to the Export tab, from where it can be downloaded into a csv file.

Attain Dashboard

LOGOUT

Prepare Bundle (1) Current Bundle (0) Bundles (3)

From To

Export (137) Invoice Number Search

BUNDLED ON EXPORT ID TOTAL YEN AMOUNT SALES

9月 15 日 2016 mrcLnSa9zEjYKsFBn ¥220,860 2

YEN -> BTC BTC/USD Finalize Payments Bundle

USD -> BTC BTC/JPY

INVOICE # BUYER ID CREATED AT YEN RECEIVED

1004244 22YmvvJh6R3MbK4W5 9月 15 日 2016 ¥110,430

1004213 MRbQvB9q33ZNwEQP3 9月 15 日 2016 ¥110,430

FINALIZED TIMESTAMP EXPORT ID HOLDING WALLET ADDRESS SALES TOTAL AMOUNT

10月 2 日 2015 25rBWMMnXv7K2FLwpC 2 \$0 BTC

Download

INVOICE # BUYER ID CREATED AT YEN RECEIVED

202437 4A3438yQFL4to9xD8 11月 16 日 2015 ¥5,000,000

301816 hGEBYTkHzoKTS2y3G 11月 16 日 2015 ¥2,500,000

Export page:

Attain Dashboard

LOGOUT

Exporter

Prepare Bundle (1) Current Bundle (0) Bundles (3)

From To

Export (137) Invoice Number Search

FINALIZED TIMESTAMP EXPORT ID HOLDING WALLET ADDRESS SALES TOTAL AMOUNT

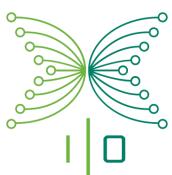
10月 2 日 2015 25rBWMMnXv7K2FLwpC 2 \$0 BTC

Download

INVOICE # BUYER ID CREATED AT YEN RECEIVED

202437 4A3438yQFL4to9xD8 11月 16 日 2015 ¥5,000,000

301816 hGEBYTkHzoKTS2y3G 11月 16 日 2015 ¥2,500,000



Transfer Funds to the Holding Wallet

Before finalizing the Bundle, the Exporter goes to an Exchange office where invoices can be traded in for BTC. These BTC are transferred to the Holding Wallet address, previously assigned to this Bundle. Once confirmed, it will be splitted again into each buyer's Invoice Ticket, and continue the regular bitcoin payment process from there.

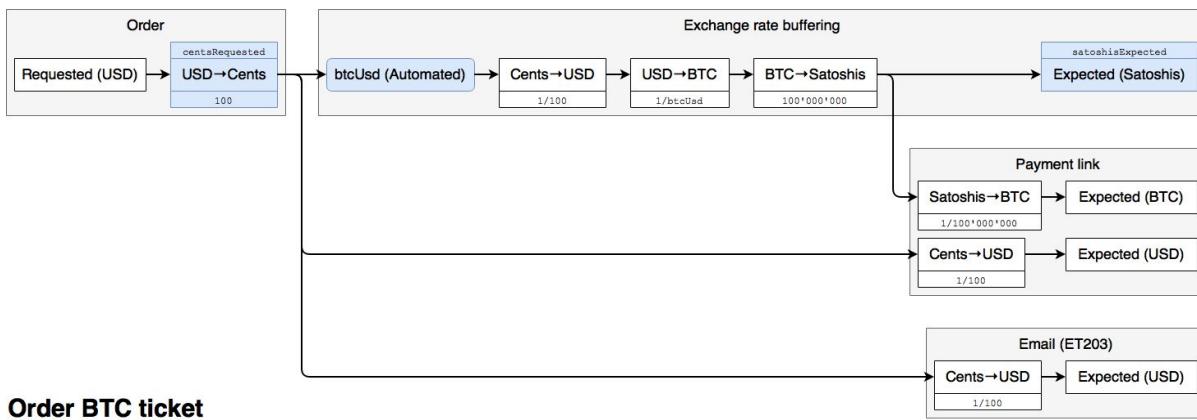
Receiving Ada

After the invoice management flow is completed, a receipt with a code named "Ada Passcode" will be send to the buyer's email account, which will be used to retrieve ADAs at a later date once the blockchain is active.

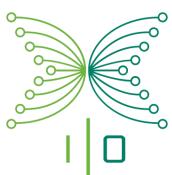
Btc Tickets

Value Calculations

When the invoice is created, the buyer must input the desired amount of USD he wants to exchange into ADA. This amount will be stored as cents in the DB. While the invoice is waiting to get the payment, the application will evaluate the exchange rate between BTC and USD in real time, and then calculate the amount of satoshis expected to make the payment.

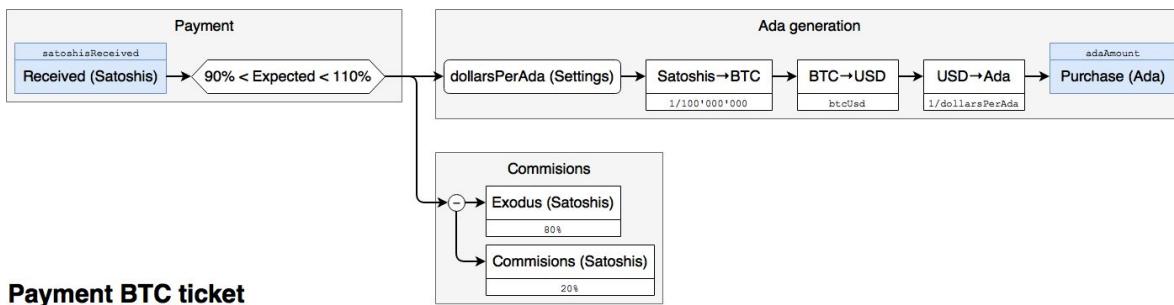


Order BTC ticket



For more details on the exchange rate between BTC and USD calculation, please address to *Misc/Bitcoin Pricing*.

The system will also apply some value calculation on the payment step, where it will convert the satoshis received into USD, in order to get the equivalent ADA amount using the USD-ADA conversion rate (known as dollarsPerAda in the code).

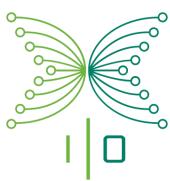


Payment BTC ticket

In order to provide an easier understanding of graphs like the following, we invite you to visit *Misc/Value Calculations Graphs Reference* section.

Receiving the Invoice

After the invoice has been created, and the sales have started, the buyer will receive an email with details of the invoice and a link to the payment page. (See next section).



Paying the Invoice

When following the payment link, the buyer will see the *USD -> BTC* exchange rate and the *Bitcoin address* (Invoice address) to execute the payment.

The buyer will also see a timer. Every time the timer reaches 0, the application will recalculate the exchange rate, getting the latest updated price.

The payment needs to be done on a single transaction, and each Invoice Ticket will be linked to a different bitcoin address, address cannot be reused in following orders.

Pay with Bitcoin

Send Bitcoins to `2N2GKpeYdUjWdXmxEVX66CuLX5ob8KUurrV`

Requested Amount (USD) `$1,000`

Bitcoin Price `$886.67`

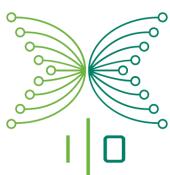
Bitcoins to Pay `$1.12781530`

You have **576** seconds left to pay the above price. After that the price will be automatically recalculated.

Warning Please complete the transaction with a single payment. Any additional payments will not be converted into ADA.

Receiving Ada

After the invoice has been paid, a receipt will be sent to the buyer's email account, with an "Ada Passcode" that will be used at a later date to retrieve the Adas once the blockchain is active.



Invalid Funds Received

When a buyer pays, but transfers a wrong amount (over the tolerance), the ticket will be placed in “*invalidFundsReceived*” state. If a ticket is in this state, it can be manually approved or marked as refunded by the Head Invoice Manager.

First the HIM should go to the tab “Invalid Funds Received”

Head Invoice Manager

Sale Pending (1)	Unpaid Invoices (0)	Confirm Bank Amount (1)
Sent to Bank Checker (1)	Prepare Bundle (0)	Current Bundle (2)
Bundles (1)	Export (144)	Expired (5622)
Invalid Funds Received (2)	BTC Orders (6)	Receipt Sent (7642)
Force Reserve ADA (1)		

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED	Actions
Hiroto Shioi	1008700	Btc		1月 26日 2017	\$10	¥0	฿0.01115088	฿0.01252572	

[Approve Ticket](#) [Mark as Refunded](#)

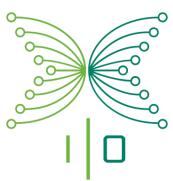
The HIM can then choose to approve the ticket with this new amount.

Attain Dashboard

Are you sure you want to approve this invoice with Invalid Funds?

[Cancel](#) [Confirm](#)

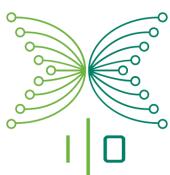
Sale Pending (1)	Prepare Bundle (0)	Current Bundle (2)
Sent to Bank Checker (1)	Export (144)	Expired (5622)
Bundles (1)	BTC Orders (6)	Receipt Sent (7642)
Invalid Funds Received (2)		
Force Reserve ADA (1)		



Or to mark the ticket as refunded, this is done by filling in the transaction id used to refund the ticket and the reason. The actual refund process has to be done manually.

The screenshot shows the Attain Dashboard interface. A modal window titled "Confirm ticket Refund" is open in the center. It contains two input fields: "Refund Transaction Id" and "Reason". Below the fields are "Cancel" and "Update" buttons. In the background, there's a list of ticket statuses: "Sale Pending (1)", "Sent to Bank Checker (1)", "Bundles (1)", "Invalid Funds Received (2)", and "Force Reserve ADA (1)". At the bottom of the dashboard, there are two buttons: "Approve Ticket" and "Mark as Refunded".

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
Hiroto Shioi	1008700	Btc	✓	1月 26日 2017	\$10	¥0	฿0.01115088	฿0.01252572



Btc Orders

In the “BTC Orders” tab the Head invoice Manager can see all the BTC tickets that are about to be paid.

For each ticket the HIM can add or edit a comment.

Attain Dashboard

LOGOUT

Invalid Funds Received (2) BTC Orders (6) Receipt Sent (7642)

Force Reserve ADA (0)

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
桃味 金雲	1008685	Btc	L	1月 24日 2017	\$1,001	¥0	฿1.09578544	฿0

No comments

Edit

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
桃味 金雲	1008687	Btc	L	1月 24日 2017	\$1,002	¥0	฿1.09688013	฿0

Editing a comment

Attain Dashboard

LOGOUT

Sale Pending (2) Unpaid Invoices (0) Confirm Bank Amount (1) From To

Sent to Bank Checker (1) Prepare Bundle (0) Current Bundle (2) Invoice Number Search

Bundles (1) Export (144) Expired (5627)

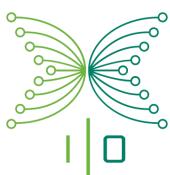
Invalid Funds Received (2) BTC Orders (1) Receipt Sent (7642)

Force Reserve ADA (0)

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
Davis Kemlers	1008699	Btc	L	1月 25日 2017	\$99,999,999	¥0	฿0	฿0

the ticket...| The comment goes here!

Cancel Save



Receipt Sent

In order to see which tickets (BTC or Bank) have been paid and the amount of satoshis that have been received, the Head invoice manager has a special tab called “Receipt Sent”.

The screenshot shows the 'Head Invoice Manager' interface. At the top, there are several buttons: 'Sale Pending (2)', 'Unpaid Invoices (0)', 'Confirm Bank Amount (1)', 'Sent to Bank Checker (1)', 'Prepare Bundle (0)', 'Current Bundle (2)', 'Bundles (1)', 'Export (144)', 'Expired (5622)', 'Invalid Funds Received (2)', 'BTC Orders (6)', and 'Receipt Sent (7642)'. On the right, there are date range filters ('From' and 'To') and a search bar ('Invoice Number'). Below this, a table lists invoices. One row is highlighted with a red box around the 'SATOSHIS RECEIVED' column, which contains '\$20'. The table has columns: NAME, INVOICE #, PAYMENT METHOD, STATE, CREATED AT, ORDERED AMOUNT, YEN RECEIVED, SATOSHIS EXPECTED, and SATOSHIS RECEIVED.

NAME	INVOICE #	PAYMENT METHOD	STATE	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED
カンン味	826252	Btc	(L)	9月 14日 2015	\$5,000	¥0	\$0	\$20

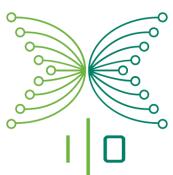
Distributor's Commissions

When an invoice is paid by a buyer, the Distributor will receive a commission paid by the Company as a reward for their Services. A percentage of the Bitcoin amount transferred will be used for this.

This percentage depends on the distributor's tier:

- If it's a Partner (compliance tier “0”) they receive 20% of the exchange transaction.
- If the Distributor has tier 1, they receive 15% and the Partner that invited them receives a 5%.
- If the Distributor has tier 2 they receive a 10%, the Distributor (with tier 1) that invited them receives 5% and the Partner that invited the Distributor with tier 1 receives 5%.
- Finally, if the Distributor has tier 3 they receive 5% and each Distributor in the upline receives 5% as well.

In order to keep track of the received commissions, Distributors have access to a special “COMMISSIONS” section where they can see each commission with its current status and a link to the transaction info on "blockchain.info". Also the top-right of the page has the total amount of bitcoins gathered.



Distributor commissions tab:

いば 鯨な
test.
email+Remedios.Rosario2942@iohk.io

MY ACCOUNT

GENERATE DISTRIBUTOR LINK

GENERATE BUYER LINK

BUYERS

COMMISSIONS

VIEW MY LINKS

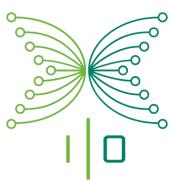
LOGOUT

 ATTAIN CORPORATION

Commissions

TOTAL: ₱2.08440879

AMOUNT	TRANSACTION	DATE	ORIGIN	INVOICE
₱1.29880706	View Transaction	2016-08-25	kjarai0726	1001323
₱0.26428745	View Transaction	2016-08-23	eddy.of.salaryman.investor	1000927
₱0.26274745	Pending	2016-08-20	shouta.shimo.24	1003143
₱0.25857125	View Transaction	2016-08-10	educe01	1000610



Monitor the Sale

Start the Sale

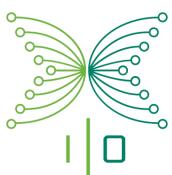
To start the sale, the application has to be (re)started with a settings file that sets the “*salesStarted*” to true.

```
{
  "public": {
    "migrations": true,
    "development": false,
    "salesStarted": true,
```

Furthermore, the sales limits have to be set for the tranche that will be active. The active tranche is set by “*currentTranche*” and the settings for that are set in “*tranches*”.

- **totalAmountAvailable:** The sales goal of the tranche, after this goal has been reached the sale will automatically stop..
- **overCapacityTolerance:** The highest amount of over sell allowed, where 0.25 is 25%. The higher this constant is, the more rapid the end sale will be at the cost of a potential over sell.

```
"salesLimits": {
  "currentTranche": "t3.5",
  "tranches": {
    "t1": {},
    "t2": {},
    "t3": {},
    "t3.5": { "totalAmountAvailable": 8000000, "overCapacityTolerance": 0.25 },
    "t4": {}
  }
},
```



To make the sales available to a region, the following setting has to be set to the regions that are wanted.

```
"residenceCountriesAllowedSale": ["KR", "JP"],
```

Check the Status of the Sale

To check the status of the sale, you have to be logged in as an Administrator or System Operator. After that you'll need to select the “Sale overview” tab on the main page. Here you can see all the information regarding the current sale:

- **Current tranche:** Indicates what tranche is currently active.
- **Sales status:** It shows if the sale is running or stopped.
- **Total amount available(TAA):** Allocated amount of USD before the sales goal has been reached.
- **Overcapacity threshold:** The largest potential overshoot of sales in percentage.
- **Amount invoiced and paid:** Total amount of USD (and tickets) that have locked Ada in a reservation.
- **Amount paid:** Total amount of USD (and tickets) that has been paid.

The screenshot shows the Sales App interface. On the left is a sidebar with the following menu items:

- attain admin
- test-email+admin@ohk.io
- SALE OVERVIEW** (highlighted with a red arrow)
- GENERATE PARTNER LINK
- VIEW PARTNER LINKS
- ADD ADMINISTRATOR
- ADD CS OFFICER
- BUYERS
- LOGOUT

Below the sidebar is the ATTAIN CORPORATION logo.

The main content area has two tabs:

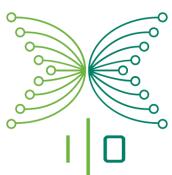
- Settings** (highlighted with a red border)
- Simulator**

Settings Tab Data:

Settings	
Current Tranche	t3.5
Sales Status	▶
Total Amount Available (TAA)	\$8,000,000
Overcapacity Threshold (OCT)	0.25
Amount Invoiced and Paid (AIP)	\$0 (Tickets : 0)
Amount Paid (AP)	\$0 (Tickets : 0)

Simulator Tab Data:

Simulator	
Total Amount Available (TAA)	1000



User Support

Find User as Customer Service

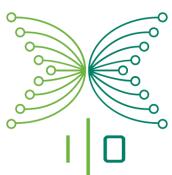
Customer Service officers can find any buyer or distributor in the application by searching for their email. Customer Service search works in two different ways:

Searching by the Exact Email Address

The CS will enter the exact email address, (for example john+doe@emailservice.com) and the application will return that single user.

The screenshot shows the Attain Dashboard with a search bar containing "john+doe@emailservice.com". A red arrow points to the search bar. Below it, a table displays user information for John Doe, with a red border around the row. The table columns are: Progress, Surname, Name, Email, Country of Residence, Account Type, Compliance Level (Old/New), Indicators, and Current Reviewer. The "Review" button is also visible.

Progress	Surname	Name	Email	Country of Residence	Account Type	Compliance Level (Old/New)	Indicators	Current Reviewer
■	Doe	John	john+doe@emailservice.com	Japan	Individual, Buyer	- / A		<button>Review</button>



“Fuzzy” Email Search

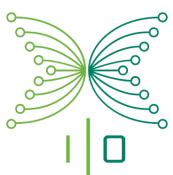
If the email has got an alias (that is for example `name+alias@emailservice.com`) then the CS can input only “`name@emailservice.com`” and it will search for all the emails like `name+alias@emailservice.com` but with any alias in the middle.

Attain Dashboard LOGOUT

Customer Service

→ 🔍

Progress	Surname	Name	Email	Country of Residence	Account Type	Compliance Level (Old/New)	Indicators	Current Reviewer
█	Doe	John	john+doe@emailservice.com	Japan	Individual, Buyer	- / A	Commission Wallet	Review
█	Smith	John	john+smith@emailservice.com	Japan	Individual, Distributor			Review



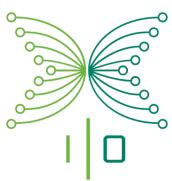
Send to Head Compliance Officer

When a CS reviews a user (by clicking on the Review button in the screenshot above), they will access the user summary page. If the user under review has not yet been sent to the Head Compliance Officer, a green button saying “Send to HCO” will be displayed. If that button is pressed, that user will be sent to the HCO for further reviewing or action.

The screenshot shows the Attain Dashboard with a user profile. The profile includes the following details:

ID	CtLyd2qfhSJ4pyseu
Account Type	Individual, Distributor
Name	Doe John
Email	john+doe@anotheremailservice.com
Status	Pending
Is the user under investigation?	No
Birthdate	01/26/2017
Distributor level	Tier 2
Branch Partner	悟志 中本 <test-email+Tad.Cobb108@iohk.io>
Referred By	John Smith <john+smith@emailservice.com>
Enrolled At	1月 16日 2017
Enrollment IP	127.0.0.1
Phone	1234
Language	en
Residence Country	Japan

A red box highlights the green “Send to HCO” button at the top left of the profile area. An arrow points to this button. To the right of the profile, there is a section titled “No document selected” with a “Choose a file” button.

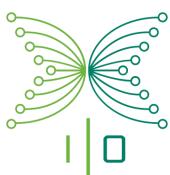


Find User as Investigator

When you're logged in as an Investigator, you will see the investigator queue. Here, you will find several sections which will be explained below.

First of all, you'll find two tabs. On the first tab, you'll see listed all users who have been sent to "*Under Investigation*" by the HCO (This will be explained in the next section of the document).

Progress	Surname	Name	Email	Country of Residence	Account Type	Indicators
Red	Smith	John	john+smith@emailservice.com	Japan	Individual, Distributor	Commission Wallet
Red	Doe	John	john+doe@anotheremailservice.com	Japan	Individual, Distributor	Commission Wallet



On the second tab, you'll see some filters which you can use to find possible fraudulent distributors. You can search by their enrollment refcodes, name, surname, company names, registration date, phone, birthdate or email address. Take into account that Investigators also use the Fuzzy Email search functionality (see Customer Service search section for further information).

Investigators also count with a partial search functionality, which means that, if you use many filters, it will return all the users that match with at least one filter, and it will order them by the amount of matches, creating a ranking of the users who best matches the filters applied. A matching by refcode will overrun any other matching, as this a very specific search, it's weight as a 100 match.

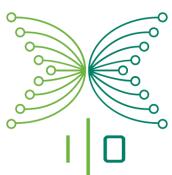
To perform the search, you'll have to press Search button after filtering by any data in the filters (the application does not allow to perform a search having all filter fields empty).

The screenshot shows the Attain Dashboard with the title "Attain Dashboard" and a "LOGOUT" link. The main area is titled "Investigator". At the top, there are two buttons: "Under Investigation (0)" and "All (3)". Below these are several input fields: "Refcode" (empty), "Name" (John Doe), "Surname" (Doe), "Company Name" (empty), "Registration Date" (mm/dd/yyyy), "Phone" (empty), "Birthdate" (mm/dd/yyyy), and "Email Address" (empty). A "Search" button is located at the bottom right of this section. Below this, a message "Results found: 3" is displayed above a table. The table has columns: Matches, Progress, Surname, Name, Email, Country of Residence, Account Type, and Indicators. The first row has 2 matches, the second 1 match, and the third 1 match. The "Matches" column is highlighted with a green border. The "Indicators" column contains buttons labeled "Commission Wallet" and "Review".

Matches	Progress	Surname	Name	Email	Country of Residence	Account Type	Indicators
2	■	Doe	John	john+doe@anotheremailservice.com	Japan	Individual, Distributor	Commission Wallet Review
1	■	Doe	James	james+doe@emailservice.com	Japan	Individual, Distributor	Commission Wallet Review
1	■	Smith	John	john+smith@emailservice.com	Japan	Individual, Distributor	Commission Wallet Review

After filtering data, you'll see below the Results count, which has the total amount of results of the query, and below you'll see the list of users with matching data.

You'll notice the *Matches* column, which corresponds to the matching functionality described before in the Review User section.



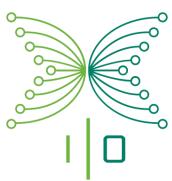
Put a User Under Investigation

When the Head Compliance Officer thinks that the user is suspicious and requires further investigation, he can send that user to the Investigator. To do so, HCO must review that user, and edit the field named “*Is user under investigation?*”.

ID	CtLyd2qthSJ4pyseu
Account Type	Individual, Distributor
Name	Doe John
Email	john+doe@anotheremailservice.com
Status	Pending
Is the user under investigation?	No
Is the user under investigation?	
Yes	
Verify Account	
Confirm Password	
Update Investigation Status	
Birthdate	01/26/2017
Distributor level	Tier 2
Branch Partner	悟志 中本 <test-

While a user is under that state, any new enrollment reflinks that he creates (or any previous one that is still valid), cannot be accessed. They also cannot change their wallet address.

To remove that user from that state, the inverse procedure must be executed.



System Operation

Monitor Sale

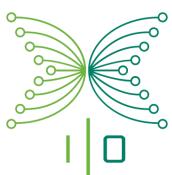
Just like the Sales Administrator, Sysop has access to the same Sales Overview page as well as summary of the current sales status.

- **Current tranche:** Indicates which tranche is currently active.
- **Sales status:** It shows if the sale is running or stopped.
- **Total amount available(TAA):** Total amount of USD that the current tranche has.
- **Overcapacity threshold:** The percentage that the TAA can be exceeded.
- **Amount invoiced and paid:** Total amount of USD (and tickets) that has been requested.
- **Amount paid:** Total amount of USD (and tickets) that has been paid.

The screenshot shows the Sales Overview page for a user named 'sysop'. The left sidebar includes links for 'SALE OVERVIEW' (which is highlighted with a red arrow), 'JOB MONITOR', and 'LOGOUT'. The main content area is divided into two sections: 'Settings' and 'Simulator'. The 'Settings' section contains the following data:

Settings	
Current Tranche	t3.5
Sales Status	▶
Total Amount Available (TAA)	\$8,000,000
Overcapacity Threshold (OCT)	0.25
Amount Invoiced and Paid (AIP)	\$1,540 (Tickets : 13)
Amount Paid (AP)	\$539 (Tickets : 11)

The 'Simulator' section at the bottom has a 'Total Amount Available (TAA)' input field containing '1000'.



Monitor Workers

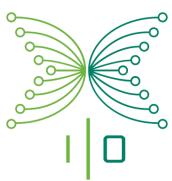
Sysop can watch the worker's status in real time by accessing the job-monitor page through the right dashboard.

The sysop will see a list of all the workers that are running and their current status. They will appear in one of these status:

- Waiting
- Paused
- Cancelled
- Failed
- Running
- Ready
- Complete
- Stalled

The screenshot shows a dashboard titled "JOBS MONITOR" with a sidebar on the left containing user information ("sysop", email, "SALE OVERVIEW", "LOGOUT") and the Attain Corporation logo. The main area is titled "Jobs" and displays a table of worker tasks. The columns are "Type", "Status", "Created", and "Updated". The "Status" column uses color-coded boxes to indicate the current state of each task.

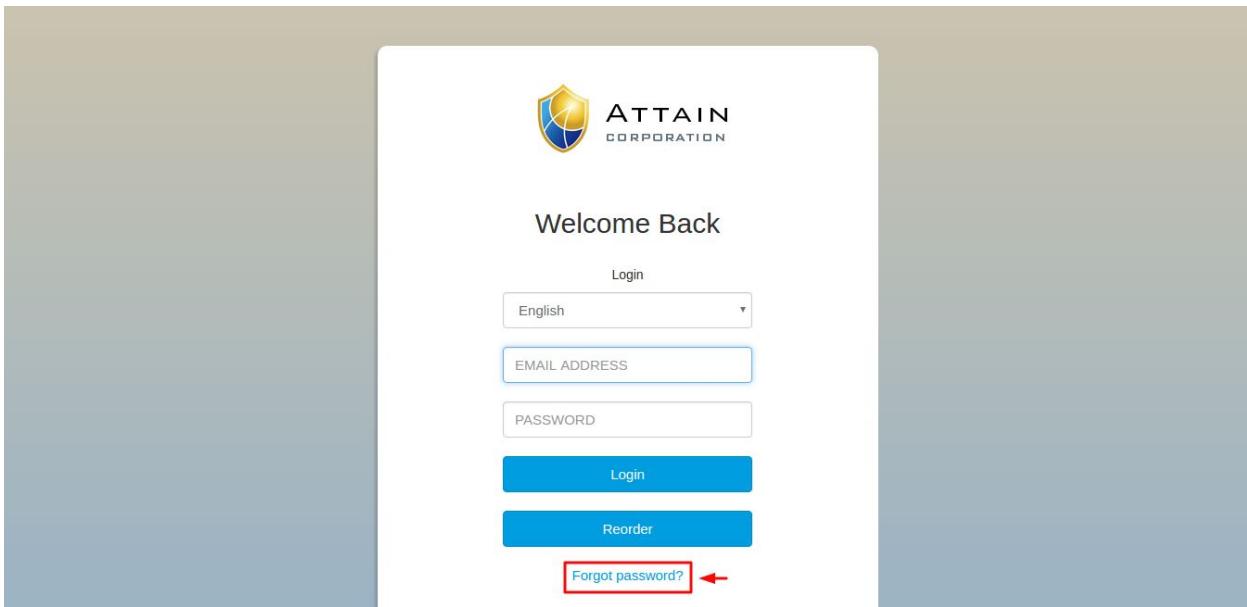
Type	Status	Created	Updated
approveComplianceBankWorker	waiting	1月 17日 2017 15:45	1月 17日 2017 15:45
approveComplianceBankWorker	completed	1月 17日 2017 15:43	1月 17日 2017 15:45
approveComplianceBtcWorker	waiting	1月 17日 2017 15:46	1月 17日 2017 15:46
approveComplianceBtcWorker	completed	1月 17日 2017 15:44	1月 17日 2017 15:46
assignAdaPasscodeWorker	waiting	1月 17日 2017 15:45	1月 17日 2017 15:45
assignAdaPasscodeWorker	completed	1月 17日 2017 15:43	1月 17日 2017 15:45
assignBtcAddressWorker	ready	1月 11日 2016 20:17	1月 11日 2016 20:18
assignBtcAddressWorker	completed	1月 11日 2016 20:17	1月 11日 2016 20:17
assignHoldingWalletAddressWorker	waiting	1月 17日 2017 15:46	1月 17日 2017 15:46
assignHoldingWalletAddressWorker	completed	1月 17日 2017 15:44	1月 17日 2017 15:46
assignInvoiceAddressWorker	waiting	1月 17日 2017 15:47	1月 17日 2017 15:47
assignInvoiceAddressWorker	completed	1月 17日 2017 15:45	1月 17日 2017 15:47
backfillHoldingWalletsWorker	ready	8月 3日 2016 13:04	8月 3日 2016 13:05
backfillHoldingWalletsWorker	completed	8月 3日 2016 13:03	8月 3日 2016 13:04
backfillInvoiceWalletsWorker	waiting	1月 17日 2017 15:45	1月 17日 2017 15:45
backfillInvoiceWalletsWorker	completed	1月 17日 2017 15:43	1月 17日 2017 15:45
btcPaymentWorker	waiting	1月 17日 2017 15:46	1月 17日 2017 15:46
btcPaymentWorker	completed	1月 17日 2017 15:44	1月 17日 2017 15:46
checkComplianceWorker	ready	12月 16日 2015 07:14	12月 16日 2015 07:16
checkComplianceWorker	completed	12月 16日 2015 07:14	12月 16日 2015 07:14

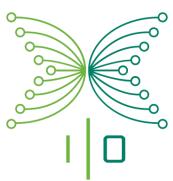


Self Administration by External Actors

Password Reset

In case you forgot your password, you can reset it by clicking on the “*Forgot Password*” button in the login page. After doing that, you need to insert your email address, and then after pressing the “*Send Reset Email*” button, you’ll receive an email with a reflash.





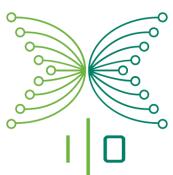
When you follow that link, you'll see a form where you have to insert the new password twice to perform the password reset.

Reset Password

Please set a password to log in:

Re-enter Password:

Reset Password

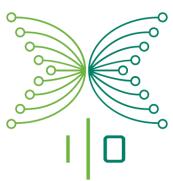


Change Email Address

As a buyer, you can change your own email address. This process requires you to have access to both your old and your new email address.

To start the process, you have to navigate to your “*My account*” panel by logging in. After that you click on “*Change Email Address*”

The screenshot shows the Sales App interface. On the left, a sidebar has a dark blue background with white text: "卜增 桃寿" (Name), "test-email+Quynn.Murphy3863@iohk.io" (Email), "MY ACCOUNT" (link), "REORDERS", "PURCHASES", and "LOGOUT". Below the sidebar is the Attain Corporation logo. The main content area has a light gray background. At the top, a message box says: "Your account is now pending Administrator approval. An approval email will be sent to your registered email address within 6 business days." Below this is a "My Account" section with tabs "INFORMATION" and "AGREEMENTS" (the former is selected). It displays: "Name" (卜增 桃寿), "Email" (test-email+Quynn.Murphy3863@iohk.io), a "Change Email Address" button (which is highlighted with a red box), and "Status" (PENDING).



The user will have to fill in their new email address and click “*Confirm*”. An email will be sent to their new email-address. Email contains a link which they'll have to click to confirm the change.

Change Email Address

A confirmation email will be sent to your currently registered address as well as new email addresses. To complete the process you must have access to both email accounts

New Email Address

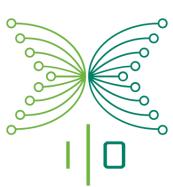
Email

test-email+Quynn.Murphy3863@iohk.io

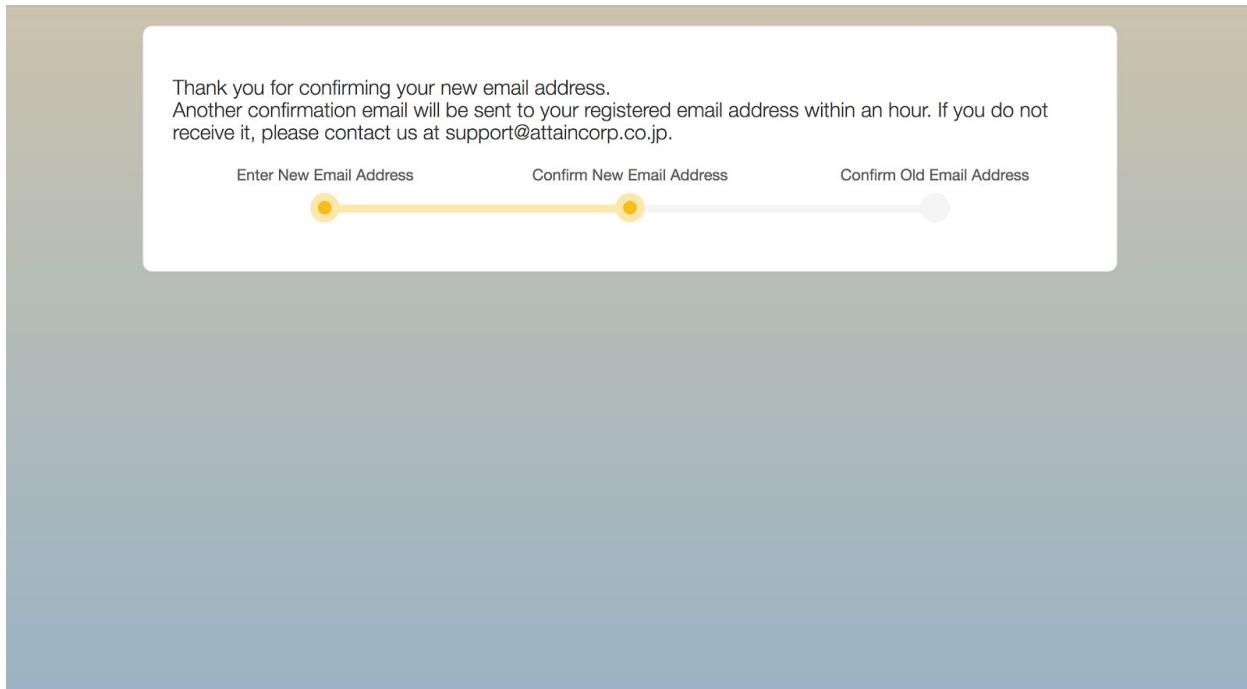
Change Email Address

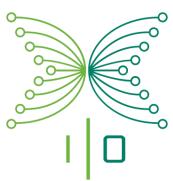
Status

PENDING

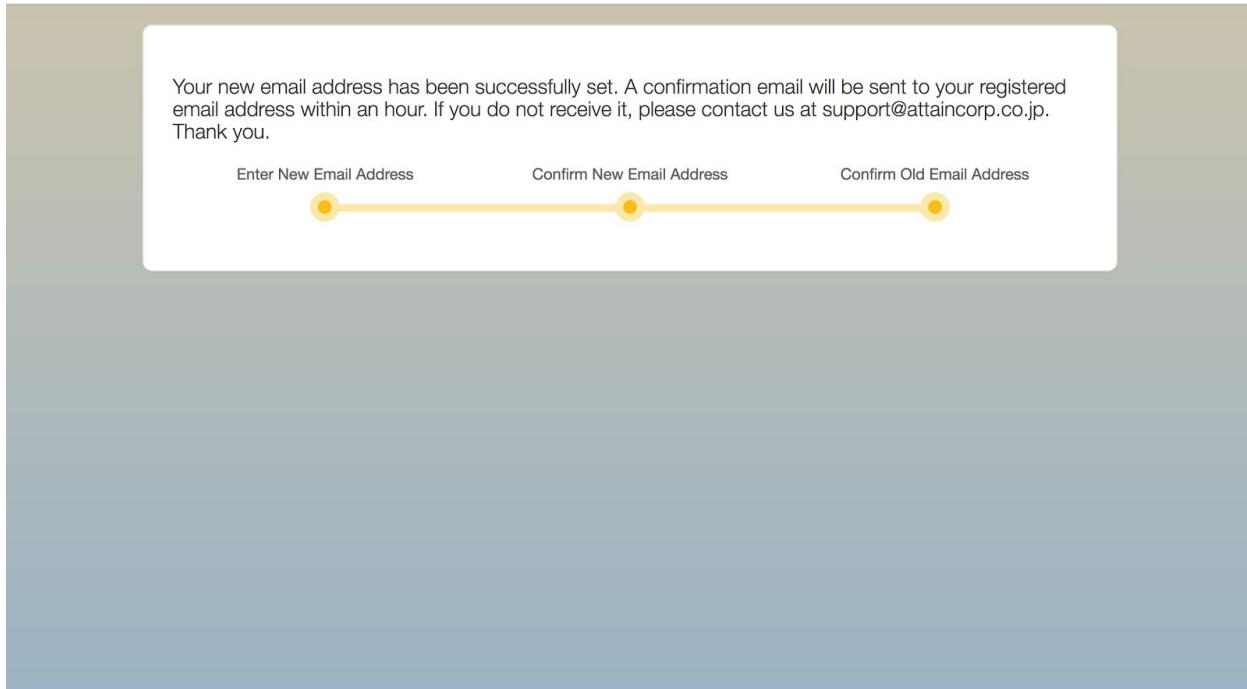


When opening the link in that email, another email will be send to their old email address.

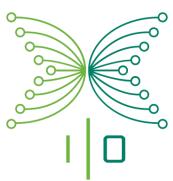




A link will be in that email as well, and after following it, your email address will be finally updated.



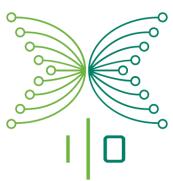
Once email address change is confirmed, all buyer's orders will regenerate their *Add Passcode*, and a new email will be delivered to the new address with the new invoice Passcodes.



Change Btc Address

As a Partner or Distributor, you have the option to change your *Commission Wallet Address*. This is done via the “*My Account*” page by clicking on the “*Update*” button next to your current address.

The screenshot shows the 'My Account' page of the Sales App. On the left, there's a sidebar with user information (Smith John, email), account links (MY ACCOUNT, GENERATE DISTRIBUTOR LINK, GENERATE BUYER LINK, BUYERS, COMMISSIONS, VIEW MY LINKS), and a LOGOUT link. At the bottom of the sidebar is the Attain Corporation logo. The main content area has tabs for INFORMATION and AGREEMENTS. Under INFORMATION, it shows Name (Smith John), Email (john+smith@emailservice.com), and Status (APPROVED). The Commission Wallet Address field is highlighted with a red box and contains the value: mk6fCVgydypzyXedHYjoefj4gwfFvHf1. To the right of this field is a blue 'Update' button with a white arrow pointing to it.



Fill in your new wallet address

Smith John
john.smith@emailservice.com

MY ACCOUNT

GENERATE DISTRIBUTOR LINK

GENERATE BUYER LINK

BUYERS

COMMISSIONS

VIEW MY LINKS

LOGOUT

ATTAİN CORPORATION

Update Wallet Address

Bitcoin Wallet Address

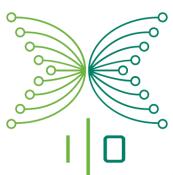
mk6fCVgydypzyXedHYjoefj4gwFvHf1

Status
APPROVED

Commission Wallet Address
mk6fCVgydypzyXedHYjoefj4gwFvHf1

Cancel Update

Then confirm the change by clicking the link send to the email address associated with the account. New orders commissions will be delivered to the new bitcoin address from now on, but ongoing payment transactions to old address won't be canceled.



Resend Ada Passcodes

When a buyer login into the application, you'll see the purchases section in the left menu. From here, you can see all the invoices you've created. For the ones that have already generated the Ada Passcode, you have the option to regenerate it.

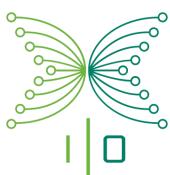
You can choose if you want to regenerate it for a single invoice or for all the invoices that have generated Ada Passcodes.

In both instances, you will receive a confirmation email to your email address with a link that you must follow in order to finish the operation.

INVOICE #	PAYMENT METHOD	CREATED AT	ORDERED AMOUNT	YEN RECEIVED	SATOSHIS EXPECTED	SATOSHIS RECEIVED	ADA PASSCODE GENERATIONS
1004398	Bank	1月 13日 2017	\$1,234	¥0	฿0	฿0	0
200000	Btc	3月 29日 2016	\$3,700	¥0	฿8.82570426	฿8.82570426	0

Regenerate ADA Code

Regenerate ADA codes



Misc

Log In

To be able to log in to the application, a person needs an account which has a password set. The language can be selected for the log-in form itself.

Logging in as Internal Actor

The account needs to be created by an admin or by the person managing the database. After that, the account is ready for the log-in process.

Logging in as External Actor

In the case of enrollment, Buyers will have to set their password first if they want to login. This is done by *forgot password* functionality.

Distributors and Partners case is different, they will be sent an email with a link after enrolling where they can set their personal password.

To log in, the email address has to be filled in with the password, after which you press the “Login” button.

Welcome Back

Login

English

test-email+Unity.Paul2461@johk.io

.....

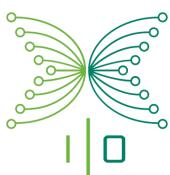
Login

Reorder

Forgot password?

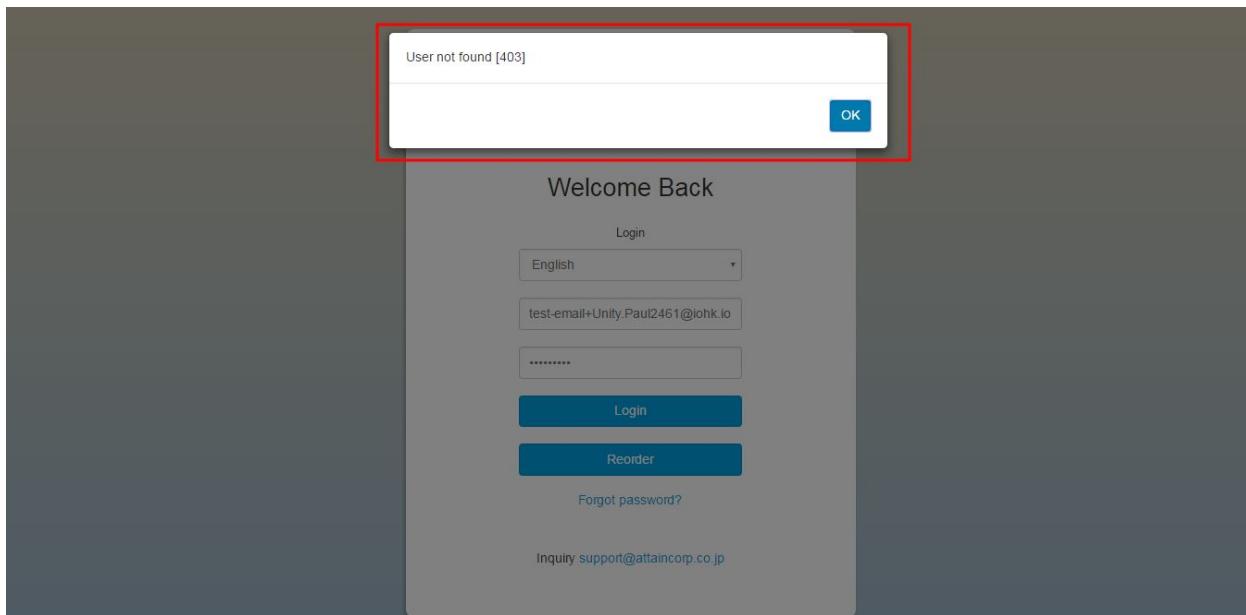
Inquiry support@attaincorp.co.jp

If there are any errors during this process the interface will show a message on the screen informing the user.

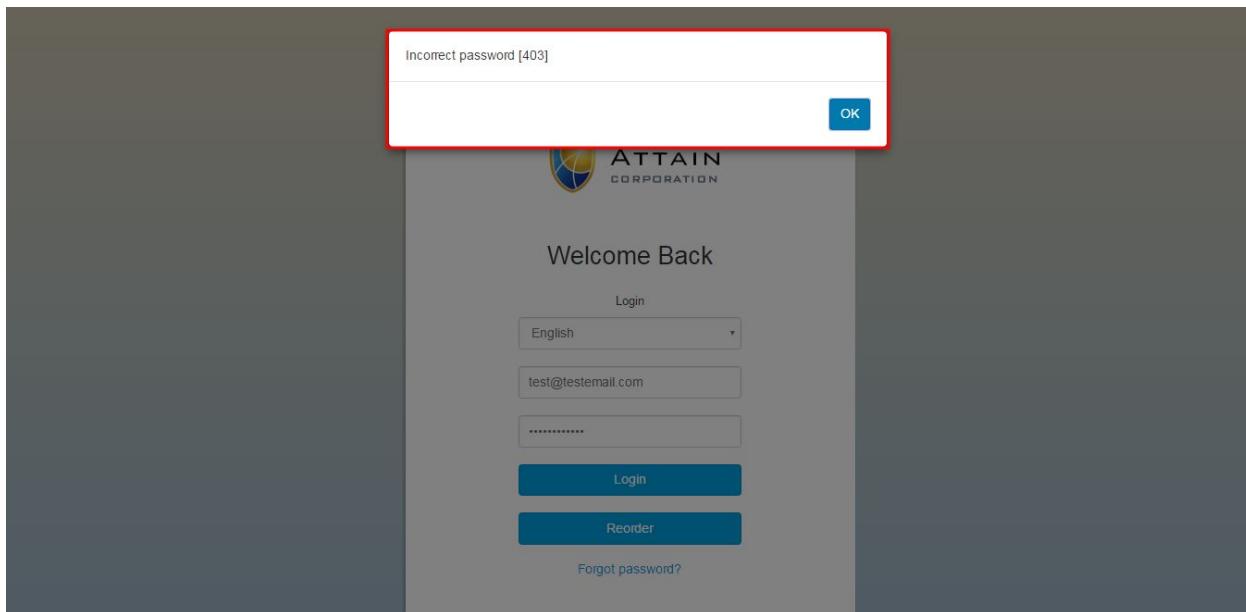


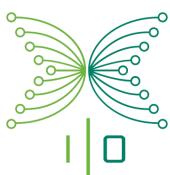
The inquiry email would help the user if he has any doubt, it would variated accordingly with the language selected.

As shown by the following examples, if the User is not found (wrong email):



Or if the password is incorrect.





If the user receives the message “Password not set” the user will have to go through the “forgot password” functionality and generate one.

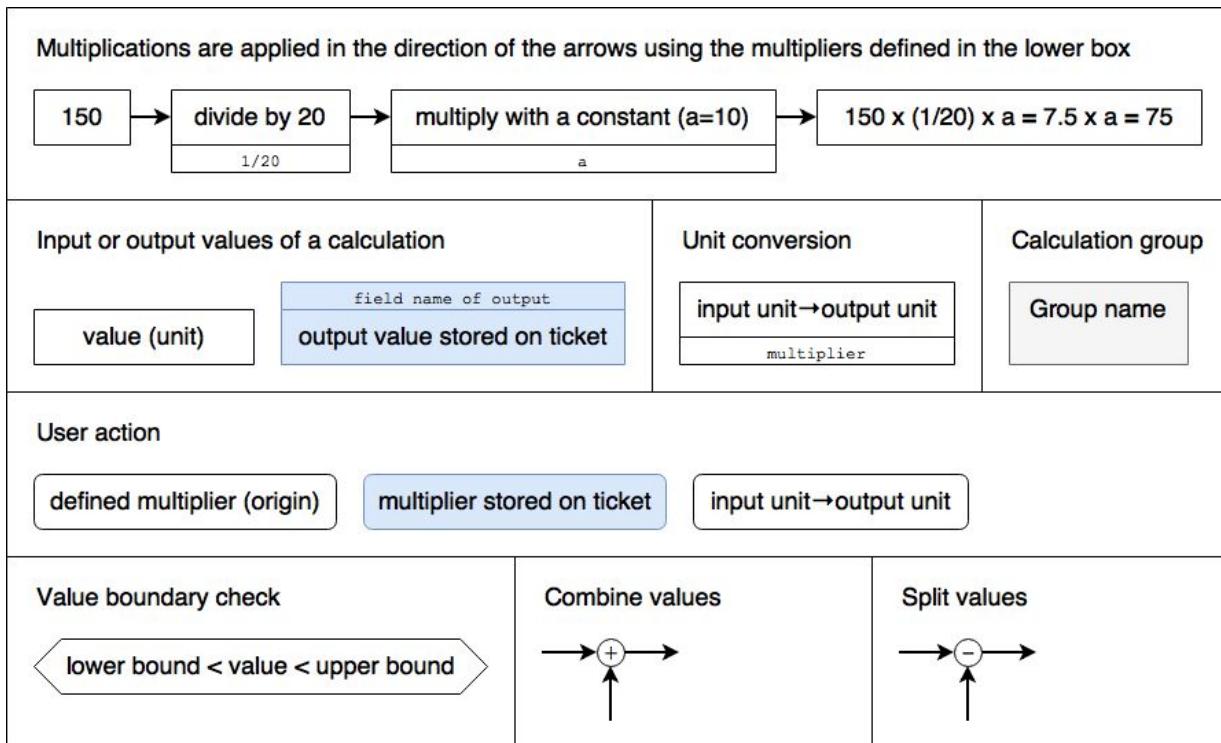
Bitcoin Pricing

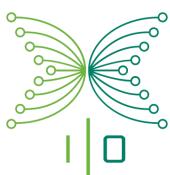
To ensure that the user gets an accurate exchange rate between USD and BTC, it fetches the bitcoin exchange rate from a list of online providers.

This way, the application provides the latest conversion between these two currencies in real time.

This feature is used to calculate the exchange rate at any stage where the invoice should receive Bitcoins (These are the stages when the Btc Invoice ticket must be paid by the buyer, or the stage where the Bank ticket must receive the Btc from the Exchange office).

Value Calculations Graphs Reference





Emails

Emails are highly tight to the application business as they are used on almost every step whether it is for enrollment, confirm user actions, inform an order state and other features. As user identity is linked to their email, any change that involves a security feature will demand a two factor verification by email.

Email templates

Email templates are in plain text, although HTML is possible. The email templates need to be made for each language and are stored in separate files. Inside the templates there are certain variables/values that can be used to construct the email itself.

Example of an email template:

```
Hello {{firstName}} {{lastName}},
```

```
Thank you for purchasing ADA!
```

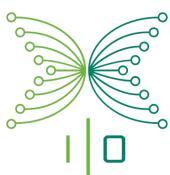
```
Please make sure that the half of the Bitcoin wallet address below corresponds  
to the address for your payment:
```

```
{{halfBtcAddress}}
```

```
Please go to the URL below and confirm the address is correct. You can complete  
your transaction from the URL only when it corresponds.
```

```
{{url}}
```

```
If it does not correspond, please contact {{supportEmail}}.
```



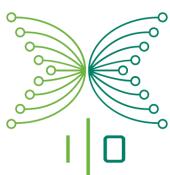
```
Invoice ID: {{invoiceNumber}}
Invoice Date: {{date}}
USD Requested: {{usdRequested}}

Regards,
{{> footer_LANG_ . }}
```

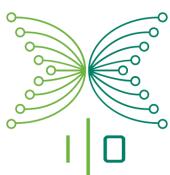
Values between double curly brackets “{{value}}” are values that will be filled in by the application when the email gets send.

The values available, change with the type of email being send.

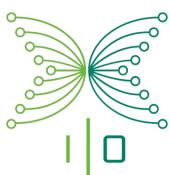
Email template	Available variables
distributor-approval.html	<ul style="list-style-type: none">• firstName• lastName• Url
distributor-signup.html	<ul style="list-style-type: none">• firstName• lastName• Url
email-changed.html	<ul style="list-style-type: none">• firstName• lastName
et101-ticket-enqueued-email.html	<ul style="list-style-type: none">• firstName• lastName• invoiceNumber• usdRequested• date
et201-ada-reserved-but-pending.html	<ul style="list-style-type: none">• firstName• lastName• invoiceNumber• usdRequested• date
et204-invoice-expired-email.html	<ul style="list-style-type: none">• firstName• lastName• invoiceNumber• usdRequested• date



et205-confirm-funds-received.html	<ul style="list-style-type: none">• firstName• lastName• invoiceNumber• usdRequested• yenReceived• date
et206-invoice-canceled-email.html	<ul style="list-style-type: none">• firstName• lastName• invoiceNumber• usdRequested• date
et301-receipt.html	<ul style="list-style-type: none">• firstName• lastName• invoiceNumber• usdRequested• yenReceived• date• btcUsd• btcReceived• paymentOptionBank• adaPurchased• adaPasscode
eu102-waiting-for-ada-while-pending-compliance.html	<ul style="list-style-type: none">• firstName• lastName
eu201-buyer-approval.html	<ul style="list-style-type: none">• firstName• lastName
eu302-wallet-address-changed-confirmation.html	<ul style="list-style-type: none">• firstName• lastName• walletAddress• url
eu303-wallet-address-changed.html	<ul style="list-style-type: none">• firstName• lastName• walletAddress
eu304-email-change-new-email-confirmation.html	<ul style="list-style-type: none">• firstName• lastName• url
eu305-email-change-current-email-confirmation.html	<ul style="list-style-type: none">• firstName• lastName



	<ul style="list-style-type: none">• newEmailAddress• url
eu306-email-changed.html	<ul style="list-style-type: none">• firstName• lastName
eu501-re-generate-all-ada-passcodes.html	<ul style="list-style-type: none">• firstName• lastName• url
eu502-re-generate-ada-passcode.html	<ul style="list-style-type: none">• firstName• lastName• url• invoiceNumber• usdRequested• date
footer.html	<ul style="list-style-type: none">• supportEmail
invoice-bank.html	<ul style="list-style-type: none">• firstName• lastName• invoiceNumber• usdRequested• processingFee• total• halfBtcAddress• url• date
invoice-btc.html	<ul style="list-style-type: none">• firstName• lastName• invoiceNumber• usdRequested• processingFee• total• halfBtcAddress• url• date
password-reset.html	<ul style="list-style-type: none">• firstName• lastName• url
verificationCode.html	<ul style="list-style-type: none">• code
waiting-for-sale-to-start.html	<ul style="list-style-type: none">• firstName

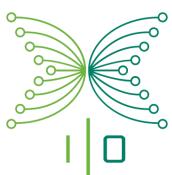


	<ul style="list-style-type: none">• lastName• invoiceNumber• usdRequested• date
wallet-address-change-confirmation.html	<ul style="list-style-type: none">• firstName• lastName• walletAddress• url

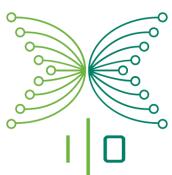
Email Naming and Numbering Conventions

Emails are organized by a two letter and number code to easily track and follow up, both for users (buyers and distributors) as well as officers.

Number	Type
EU---	User actions
EU1--	Non-logged in users actions
EU101	Password reset
EU102	Waiting for Ada while pending compliance
EU103	Email verification on enrollment
EU2--	Compliance officer actions
EU201	Buyer compliance approved
EU202	Distributor approval
EU3--	Distributor actions
EU301	Distributor signup
EU302	Wallet address change confirmation
EU303	Wallet address changed
EU304	Email change new email confirmation
EU305	Email change current email confirmation
EU306	Email changed



EU4--	Head compliance officer actions
EU401	Email changed
EU5--	ADA code regeneration
EU501	Re-generate all ADA pass-codes
EU502	Re-generate ADA pass-code for invoice
ET---	Invoice ticket states
ET1--	Non-reserved invoice ticket states
ET101	Enqueued
ET102	Waiting for sale to start
ET2--	Reserved invoice ticket states
ET201	Ada reserved but pending
ET202	Invoice Bank
ET203	Invoice Btc
ET204	Invoice expired
ET205	Confirm funds received
ET206	Invoice canceled
ET3--	Paid invoice ticket states
ET301	Receipt



EU - User Actions

Emails sent as a direct consequence of a user's action.

EU1 - Non-Logged in User's Actions

Emails sent as a direct consequence of a non-logged in user performing an action.

- **EU101 - Password reset**

Template: password-reset.html

Description: When a user requests to reset their password, an email with a reset link is sent to them.

- **EU102 - Waiting for Ada while pending compliance**

Template: eu102-waiting-for-ada-while-pending-compliance.html

Description: During the “All Ada Reserved period”, as a newly enrolled user or as an approved buyer needing compliance, users should receive a compliance email EU102 letting them know that they are in compliance.

- **EU103 - Email verification on enrollment**

Template: verificationCode.html

Description: This email contains a refcode that verifies a user's email address when enrolling as distributor or buyer. It's sent after a user inputs their email address and pushes the “receive code” button.

EU2 - Compliance Officer Actions

Emails sent as a direct consequence of a Compliance Officer performing an action.

- **EU201 - Buyer compliance approved**

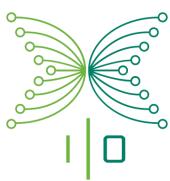
Template: eu201-buyer-approval.html

Description: Is sent to a pending buyer who has been approved.

- **EU202 - Distributor approval (Distributor compliance approved)**

Template: distributor-approval.html

Description: Is sent to a pending distributor who has been approved.



EU3 - Distributor Actions

Emails sent as a direct consequence of a distributor performing an action.

- **EU301 - Distributor signup**

Template: distributor-signup.html

Description: When a distributor finishes enrolling this email is sent to them containing a link to set a password.

- **EU302 - Wallet address change confirmation**

Template: eu303-wallet-address-changed-confirmation.html

Description: When a distributor changes his wallet address he should receive a confirmation email containing a confirmation link.

- **EU303 - Wallet address changed**

Template: eu303-wallet-address-changed.html

Description: When a distributor changes his wallet address he should receive an email notifying him about the change.

- **EU304 - Email change new email confirmation**

Template: eu304-email-change-new-email-confirmation.html

Description: Email sent to the user's old email address, with a link to change the email to the new email address.

- **EU305 - Email change current email confirmation**

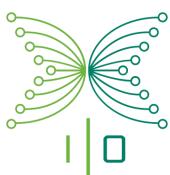
Template: eu305-email-change-current-email-confirmation.html

Description: The user receives this email at their newly registered email, after they click the link in the email sent to their old email address. This email has a final confirmation link.

- **EU306 - Email changed**

Template: eu306-email-changed.html

Description: The final email the user receives after changing their email, informing them that the process has been completed successfully.



EU5 - ADA Pass-Code Regeneration

Emails sent when ADA pass-codes are regenerated for various reasons.

- **EU501 - Regenerate all ADA pass-codes**

Template: eu501-re-generate-all-ada-passcodes.html

Description: When an user requests to have all of the ADA pass-codes be regenerated, they receive this email with a link.

- **EU502 - Regenerate ADA pass-codes for invoice**

Template: eu502-re-generate-ada-passcode.html

Description: When a user requests to regenerate ADA pass-codes for a specific invoice, they will get this email with a link to regenerate.

ET - Invoice Ticket States

Emails sent as a direct consequence of a worker moving invoice tickets in the invoice ticket state machine.

ET1 - Non-Reserved Invoice Ticket States

Emails sent when a ticket makes a transition to a Non-reserved invoice ticket state.

- **ET101 - Enqueued**

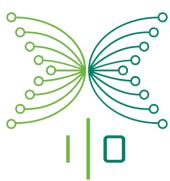
Template: et101-ticket-enqueued-email.html

Description: During All Ada Reserved period, as an approved buyer reordering or an approved buyer needing compliance reordering or a pending user enrolling, they should receive an queued email ET101 informing them that all Ada is currently reserved but Ada will be allocated to them once Ada is freed.

- **ET102 - Waiting for sale to start**

Template: waiting-for-sale-to-start.html

Description: Should be sent when the setting salesStarted is set to false, or when the sales goal has been reached for the current tranche.



ET2 - Reserved Invoice Ticket States

Emails sent when a ticket makes a transition to a Reserved invoice ticket states.

- **ET201 - Ada reserved but pending**

Template: et201-ada-reserved-but-pending.html

Description: During Ada Available, as an approved buyer needing compliance or and newly enrolled user, I should received ET201 an email letting me know that I am pending compliance but that Ada has been allocated to me.

- **ET202 - Invoice Bank**

Template: invoice-bank.html

Description: During Ada Available, as an approved buyer not needing compliance I should get an invoice for my order

- **ET203 - Invoice Btc**

Template: invoice-btc.html

Description: During Ada Available, as an approved buyer not needing compliance I should get an invoice for my order

- **ET204 - Invoice expired**

Template: et204-invoice-expired-email.html

Description: Is sent when an invoice expires. The email contains a reorder link.

- **ET205 - Confirm funds received**

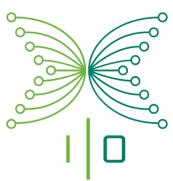
Template: et205-confirm-funds-received.html

Description: When a buyer has made a bank transaction which has been confirmed by Attain, the buyer should receive an email notifying the buyer about the transaction been successful.

- **ET206 - Invoice canceled**

Template: et206-invoice-canceled-email.html

Description: When a buyer requests to cancel their invoice and HCO cancels it, the buyer will receive this email.



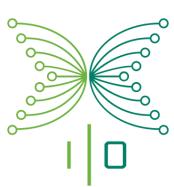
ET3 - Paid Invoice Ticket States

Emails sent when a ticket makes a transition to a Paid invoice ticket states.

- **ET301 - Receipt**

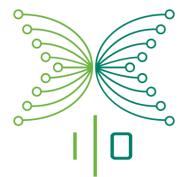
Template: et301-receipt.html

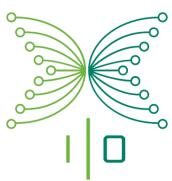
Description: Contains the receipt info and the Ada passcode, and is send to the user when the purchase has been completed.



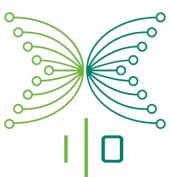
Sales App | Frontend - Business logic

Frontend - System Setup

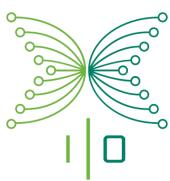




System Requirements	4
Operating System	4
Windows	4
Linux	4
MacOS	4
Node	4
NPM	5
Meteor	5
MongoDB	5
Backend	6
System Configuration	7
Environment Variables	7
Root URL	7
MongoDB URL	7
MongoDB Oplog URL	7
Mailgun Domain Name	8
Mailgun API Key	8
Backend Server IP Address	8
BCC Email Address	8
Backend Client Id	8
Backend Access Token	9
Backend Signature	9
Settings File	9
Sales and Tranches Settings	9
Regional Settings and Legal Documents	11
Legal Documents	13
Price Service Provider	16
Sales App Api Token	18
Backend Settings	19
Loggly	20
AWS	20
Deployment	21
Compose	21
SSL	21
Failover	22
Create a Database	23



Oplog	24
Galaxy	25
Certificate	26
AWS S3	28
Loggly	37
Mailgun	38
Recommendations	39
Example Setup	39
Hosting Services Configuration	39
Galaxy	39
Compose	39
Slack	39
Settings	39
Scripts	45
Known Quick Fixes	45
Scaling Down and Up Container	45
Scale Down	46
Scale Up	47
Restart the Application	48
Stop the containers	48
Start the containers	50



System Requirements

Operating System

Windows

For this project, it is **not** recommended to use a Windows development environment. There are too many issues with tests not running and certain modules not working in Node.js.

Linux

For developing any Linux distribution with a GUI is enough to manage this app. There might be some small issues with rights for the Meteor application, but they can be easily resolved by changing ownership of the .meteor folder. The developers that used Linux are using Ubuntu 16.04.

MacOS

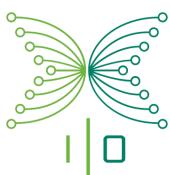
Most of the developers working on this project used Macs although there were some slight problems caused by running wrong versions of Node.js. This can be easily resolved by using Node Version Manager.

Node

The Node current version we're using in this project is 4.6.0. In order to download this version, you can either download it from their web or use Node Version Manager (nvm), following the tutorial.

References:

- Node download: <https://nodejs.org/download/release/v4.6.0/>
- NVM tutorial:
<http://stackoverflow.com/questions/7718313/how-to-change-to-an-older-version-of-node-js>



NPM

Node Package Manager is the best tool for importing and updating external packages. The current version in this project is 3.10.8, and it's automatically installed when installing Node, but in case you need to install it manually, you can download it (see the references for the link).

Please make sure to run `meteor npm install --save` once it's installed, so it downloads all the dependencies defined in `package.json`.

References:

- NPM download: <https://registry.npmjs.org/npm/-/npm-3.10.8.tgz>

Meteor

The application runs in Meteor 1.4.2.3, to install a specific version of Meteor you can do this in two ways:

- You can add the release version as a parameter to the download endpoint
`curl "https://install.meteor.com/?release=1.4.2.3"`
- You can use the `meteor update` command
`meteor update --release 1.4.2.3`

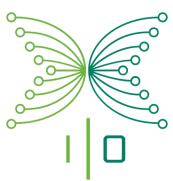
MongoDB

MongoDB is the default database that comes along with Meteor. It's a NoSQL engine which uses JSON standard to store its documents.

MongoDB is installed automatically when Meteor is installed in the project. The current version that is running in this project is 3.2.6.

However, instead of using the latest engine called WiredTiger (which is the default engine in this mongo version), we are using the MMAPv1 engine.

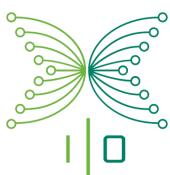
In order to connect to the tests databases, please ensure that the connection to the database uses SCRAM-SHA1 authentication mechanism, and Self-Signed Certificate for the SSL connection protocol.



Backend

Sales process is managed by two independent applications, Sales application that allow users to register, generate orders, see balances, etc; and Backend that handles all Bitcoin related interactions.

This architecture has many security advantages and allows decoupling the order process from the payments itself. Although each application can run independently, both are needed to complete the purchase process. We'll discuss in details about Backend and Sales application configuration in Backend Configuration Settings sections.



System Configuration

Environment Variables

The application is making use of several environment variables, each listed below.

Root URL

Name ROOT_URL

Example value `https://*****`

Description Root URL for the application.

MongoDB URL

Name MONGO_URL

Example value `mongodb://*****@*****:*****?ssl=true&authMechanism=SCRAM-SHA-1`

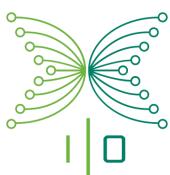
Description URL for MongoDB connection string. Should contain username, password, url and port number for primary and fallback databases, database name, ssl setting, authentication mechanism. Can be found at Compose deployment overview page.

MongoDB Oplog URL

Name MONGO_OPLOG_URL

Example value `mongodb://oploguser:*****:*****@***:*****/local?authSource=admin&ssl=true`

Description URL for MongoDB oplog. Should contain username, password, url and port number for the oplog, database name associated with the user, SSL setting.
Can be found at Compose deployment overview page.



Mailgun Domain Name

Name MAILGUN_DOMAIN
Example value *****
Description Domain name for mailgun. Can be found at Mailgun domain list.

Mailgun API Key

Name MAILGUN_API_KEY
Example value key-*****
Description API key for mailgun. Can be found at Mailgun domain list.

Backend Server IP Address

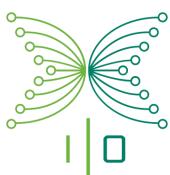
Name IOHK_BACKEND_SERVER_IP
Example value https://*****/api/v1/
Description IP address for IOHK backend server.

BCC Email Address

Name ATTAIN_BCC_EMAIL
Example value test-email+test-attain@iohk.io
Description This is used as a BCC email address when sending out emails.

Backend Client Id

Name IOHK_BACKEND_CLIENT_ID
Example value SalesAppTesting
Description Client ID for IOHK backend server.



Backend Access Token

Name IOHK_BACKEND_TOKEN

Example value ****_****_****_****_*****

Description Token for IOHK backend server.

Backend Signature

Name IOHK_BACKEND_SIGNATURE

Example value ****_****_****_****_*****

Description Signature for IOHK backend server.

Settings File

Sales application settings are divided in two sections. **Settings.json** contains all general and public settings, none of those settings could compromise application security. On the other hand, all sensitive keys and settings are placed in a **Secret.json** for each environment.

Naturally, *production* secret file is not pushed to git repository but it's managed separately and added on deploy by authorised personnel only.

Sales and Tranches Settings

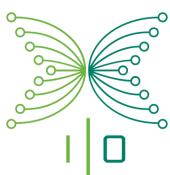
The sales has two main configurations:

- salesStarted: It indicates if the sales are on/off.
- salesLimits: It contains configurations for the tranches.

The “*expireTickets*” configuration is used to manage for how long a ticket is considered valid. It’s separated by “bank” and “btc”.

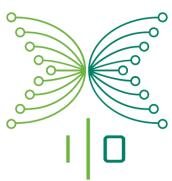
Each period (tranche) has its configuration inside “*salesLimits*”. Here you will have two configurations:

- currentTranche: Indicates which tranche is currently active.
- tranches: Here you can define as many tranches as you want, each one will have two options “*totalAmountAvailable*”⁽¹⁾ and “*overCapacityTolerance*”⁽²⁾



```
// file path: "/config/settings.json"
{
  "public": {
    "salesStarted": true,
    "expireTickets": {
      "bank": {
        "afterInvoiceSent": {
          "days": 3,
          "businessDays": true
        }
      },
      "btc": {
        "afterInvoiceSent": {
          "days": 3,
          "businessDays": false
        }
      }
    }
  },
  "salesLimits": {
    "currentTranche": "t3",
    "tranches": {
      "t1": {},
      "t2": {},
      "t3": {
        "totalAmountAvailable": 14000000,
        "overCapacityTolerance": 0.2
      },
      "t4": {}
    }
  },
  ...
}
```

- (1) totalAmountAvailable: Predefined total of USD equivalent of ADA's.
- (2) overCapacityTolerance: Percentage that the totalAmountAvailable can be exceed.



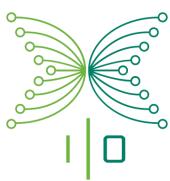
Regional Settings and Legal Documents

Before running the application, it is necessary to define from which residence countries can buyers and distributors enroll from.

You also need to define which countries are allowed for the sale. Buyers from countries where the sale has not started can order ADA, but the ticket is not processed until the sales start.

Example: Available countries in the settings file. ('/config/settings.json')

```
{  
  ...  
  "availableCountries": ["JP", "KH", "CN", "KR", "PH", "TH", "VN"],  
  "residenceCountriesAllowedSale": ["KR"],  
  ...  
}
```

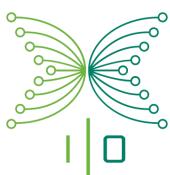


Each available country belongs to a region which must be defined in settings file. Each region has its own configuration related to enrollment and legal documents. For the enrollment section, you need to define the payment option (btc and/or bank) and the “paymentMinimum”⁽¹⁾.

Example: This shows mapping between countries and regions in the settings file.
('/config/settings.json')

```
{  
  ...  
  "regionalMappings": {  
    "JP": "regionNippon",  
    "KH": "regionCambodia",  
    "CN": "regionChina",  
    "HK": "regionHongKong",  
    "KR": "regionSouthKorea",  
    "MY": "regionMalaysia",  
    "PH": "regionPhilippines",  
    "SG": "regionSingapore",  
    "TW": "regionTaiwan",  
    "TH": "regionThailand",  
    "VN": "regionVietnam"  
  },  
  ...  
}
```

(1) paymentMinimum: This is the minimum amount of USD that a buyer can purchase.



Legal Documents

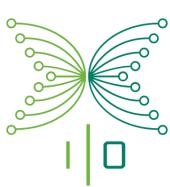
Existing types of policies are: Term of conduct (TOC), PRIVACY and RISK. Each type of user has its own set of documents. Distributor's policies differ depending on their tier.

To add or update policy documents you have to place the documents in *attainenroll/public* folder. Then in the settings.json for each region, you need to write the filename for each policy and describe in what language it is written in (you could define a default option too).

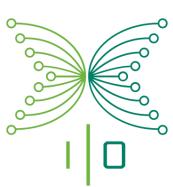
TOC and PRIVACY policies can expire, so the settings needs to have the date of the last update of the files (see "latestPolicyUpdate" in settings file).

Example: Regional settings by each region (settings file '/config/settings.json')

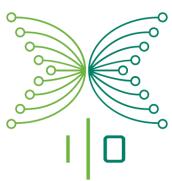
```
{  
  ...  
  "regionalSettings": {  
    "regionNippon": {  
      "enrollment": {  
        "paymentOptions": ["bank", "btc"],  
        "paymentMinimum": 1000  
      },  
      "latestPolicyUpdate": {  
        "buyer": {  
          "TOC": "2016-11-26"  
        },  
        "distributor": {  
          "0": "2016-02-04",  
          "1": "2016-02-04",  
          "2": "2016-02-04",  
          "3": "2016-02-04"  
        },  
        "PRIVACY": "2015-10-01"  
      },  
      "policies": {  
        "TOC": {  
          "distributor": {  
            "0": [{  
              "lang": "ja",  
              "file": "TOC_ja.html"  
            }]  
          }  
        }  
      }  
    }  
  }  
}
```



```
        "file": "appoiu.pdf"
    }],
    "1": [
        "lang": "ja",
        "file": "t1k4j3kkjh4l3k4.pdf"
    ],
    "2": [
        "lang": "ja",
        "file": "t2j3j9dadflk34.pdf"
    ],
    "3": [
        "lang": "ja",
        "file": "t3y4y2y5uhkjh.pdf"
    ]
},
"buyer": [
    "lang": "ja",
    "file": "toCforbuyerJP_v5a_jp_userpoli.pdf"
],
"default": [
    "lang": "ja",
    "file": "toCforbuyerJP_v5a_jp_userpoli.pdf"
],
},
"RISK": {
    "default": [
        "lang": "ja",
        "file": "riskPolicyJP_v2a_jp_rskpoli.pdf"
    ]
},
"PRIVACY": {
    "default": [
        "lang": "ja",
        "file": "privacy.pdf"
    ]
}
}
```



```
},
"regionCambodia": {
    ...
},
...
}
```

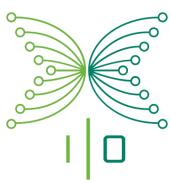


Price Service Provider

In order to get the bitcoin rate price in a controlled manner, Sales application uses a protected endpoint⁽¹⁾ provided by the Backend that gets the rate prices according to the currency code (for example: USD, JPY) from some predefined providers (bitstamp, bitpay, etc). Currently only USD prices are used.

Using Backend as price provider instead of accessing each provider directly, has several advantages:

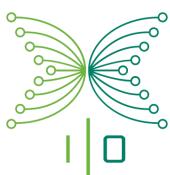
- Unified format for different providers
- Simplifies provider switching if needed.
- Controlled accessibility, doesn't rely on third party availability.
- Keeps request per minute steady. Some providers had limits on the rpm an IP can query their endpoints. If we query the prices "on demand", there is always a risk of request limit overflow and get the IP blocked.

**Example:** Backend response

```
[  
 {  
   "providerId": "bitpay",  
   "currencyCode": "JPY",  
   "price": 112632,  
   "lastUpdate": "2016-12-28T12:43:05.705Z"  
 },  
 {  
   "providerId": "kraken",  
   "currencyCode": "JPY",  
   "price": 114000,  
   "lastUpdate": "2016-12-28T12:43:06.319Z"  
 },  
 {  
   "providerId": "blockchain",  
   "currencyCode": "JPY",  
   "price": 112759,  
   "lastUpdate": "2016-12-28T12:43:06.465Z"  
 }  
 ]
```

With this answer, the application chooses the first service price that matches these two conditions:

1. It exists in the *Providers ranking*⁽²⁾ (defined in the settings file).
2. Difference between "*lastUpdate*" time and current time does not exceed max outdated tolerance (*outdatedMaxToleranceInMinutes*⁽³⁾).



Settings file ('/config/settings.json'):

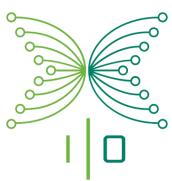
```
{  
  ...  
  "btcPriceCriteria": {  
    "providersRanking": [  
      "bitstamp",  
      "blockchain",  
      "bitpay",  
      "bitgo",  
      "bitfinex",  
      "kraken"  
    ],  
    "outdatedMaxToleranceInMinutes": 2  
  }  
}
```

For more information, the algorithm is defined in file: '/server/workers/btc-payment-worker.js'.

- (1) Backend Endpoint *GET: "IOHK_BACKEND_SERVER_IP/price/rates/\${currencyCode}"*
- (2) Providers ranking: Some providers are more reliable than others, so the application has its own ranking defined.
- (3) outdatedMaxToleranceInMinutes: Tolerance criteria defined in the settings file.

Sales App Api Token

Some Sales application API endpoints manipulate sensitive data, and although used by external systems, they are not public. These endpoints uses an static TOKEN based authentication which is predefined in the setting key SALES_APP_AUTH_API_TOKEN.

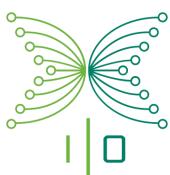


Backend Settings

There are four basic setting configurations needed for the sales-app to communicate with the Backend (Payment processor):

- **IOHK_BACKEND_SERVER_IP:** This is the complete URL to Backend endpoint services root (for example: *https://services.backend.com/api/v1/*)
- **IOHK_BACKEND_CLIENT_ID:** (*) This is used to identify Sales application instance on the backend. On a signed response, this (as a header) will allow the Backend to know who is signing in and easily log the verification process.
- **IOHK_BACKEND_TOKEN:** (*) Every new protected request to the backend needs this Bearer TOKEN header to authenticate. This should be a Read-Write TOKEN.
- **IOHK_BACKEND_SIGNATURE:** (*) This key will be used to sign (encrypt) backend payload responses to yield any middleman interference.

(*) All this three Keys are generated and provided by the backend itself and linked together, cannot change one without the others. They need to be manually generated and added to both systems to create the bound.



Loggly

The application uses the miktam:loggly (version 2.0.0) package for Meteor to communicate with the Loggly service. For this you have to add some settings to the secret.json file.

```
"loggly": {  
  "enabled" : true,  
  "token": "*****_*****_*****_*****_*****",  
  "subdomain": "*****.loggly.com",  
  "tags" : ["salesapp-test"],  
  "auth": {  
    "username": "myUsername",  
    "password": "myPassword"  
  }  
}
```

The “enabled” field is not used in the package, but the application uses it as a flag to turn on/off the logs to Loggly.

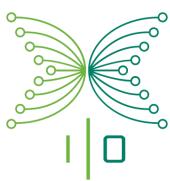
References:

- Miktam:loggly <https://atmospherejs.com/miktam/loggly>

AWS

Amazon S3 is cloud storage for the Internet. Regarding this project, we are currently using AWS to store pictures and documents. To properly configure AWS, you need to set the correct keys under settings file.

```
AWSAccessKeyId: <AWSAccessKeyId>  
AWSSecretAccessKey: <AWSecretAccessKey>  
SLINGSHOT_S3_REGION: <SLINGSHOT_S3_REGION>  
SLINGSHOT_S3_BUCKET: <SLINGSHOT_S3_BUCKET>
```



Deployment

Deployment is the procedure of moving the application from a local machine to a web server. The steps provided here are for deploying to Galaxy given a Linux machine with Meteor.js installed. In addition a MongoDB instance is created on Compose, and a file container (for user uploaded files) is created on Amazon S3. However the Application can be deployed through any means available for deploying a Meteor application, and the MongoDB instance can be hosted through other providers or being run on the local machine.

These are the requirements for deploying

- An account on Compose (compose.com)
- An account on AWS (aws.amazon.com)
- An account on Galaxy (galaxy.meteor.com)
- The source code
- Linux with Bash installed

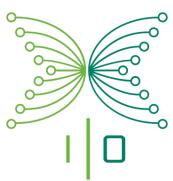
Compose

The application require an instance of MongoDB 3.2, which can be hosted anywhere as long as it's accessible via a URL. The steps in this setup assumes the usage of Compose, but nothing prevents the usage of any other hosting solution. The application needs read and write permission on the database and preferably Oplog access.

SSL

When separating the location of the application and the database, like letting Compose host the database and Galaxy the application, it's important to secure the data link to prevent interception. SSL is a technique to secure this link and should be enabled for the MongoDB deployment. See Environment variables section for more information how to set it up.

SSL has to be enabled when creating a MongoDB 3.2 deployment, and the URL has to be configured in accordance with the Environment variables section.

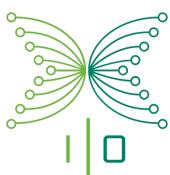


Failover

If the database connection goes down the application will normally stop working, so to improve robustness a fallback URL can be used, this technique is called failover.

Failover address is offered by default when setting up an MongoDB 3.2 database, and can be enabled in accordance with the Environment variable section.

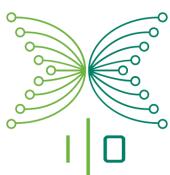
The screenshot shows the Heroku Compose interface for managing databases. On the left, there's a sidebar with icons for Overview, Resources, Browser, Backups, Jobs, Settings, Metrics, Logfiles, Access, Security, and Add-ons. The main area is titled "Deployment Overview" and shows details for a MongoDB 3.2.10 instance named "tangible-mongodb-57". The instance is located in "us-east-1 on aws", has a "Healthy" status, and is TCP Available. It uses 2GB of 2GB Disk and costs \$53.50/month. The storage engine is mmapv1. The access level is Admin. A note field says "Click to add some notes." Below this, there's a "Connection info" section with a "Connection string" field containing the URL: "mongodb://<username>:<password>@<redacted>/admin?ssl=true". The "redacted" part is highlighted with a red rectangle.



Create a Database

The screenshot shows the Heroku Compose interface. At the top right, there is a 'Create Deployment' button with a red circle around it. Below the button, there are three existing deployment entries, each with a status bar and a location indicator (aws:us-east-1). A message at the bottom says, "Don't see the deployment you're looking for? You might need to ask an administrator to grant you access. [Learn more](#) about access and roles at Compose."

The screenshot shows the 'New Deployment' page for MongoDB. On the left, there is a sidebar with categories: Production Deployments (MongoDB, RethinkDB, Elasticsearch, Redis, PostgreSQL, RabbitMQ, MongoDB (classic)), Beta Deployments (etcd, MySQL, ScyllaDB), and Alpha Deployments (Disque). The main area is titled 'New MongoDB Deployment'. It includes fields for 'Deployment Name' (with a placeholder), 'Location' (set to 'US East 1 (Northern Virginia)'), 'Enable SSL access' (checked), 'WiredTiger Storage Engine' (unchecked), 'Database Version' (set to '3.2.11'), and 'Allocate initial deployment resources' (a slider from 1GB to 125GB, currently at 1GB). A note at the bottom says, 'Start with 1GB Storage / 102MB RAM at \$31.00/month.' At the bottom right is a 'Create Deployment' button with a red circle around it.



Oplog

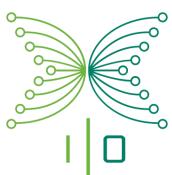
Meteor is built on push notification to update clients with new data changes. When several instances are running in parallel (containers in Galaxy) each connected to a single shared database, the propagation of data between instances is key to keep all instances and clients up to date. Meteor is subscribing to the Oplog to achieve this, and if the Oplog is not configured or goes offline then Meteor will fallback on polling with a frequency of 6 request per minute.

Oplog is not enabled by default in Compose when setting up a MongoDB 3.2 database, instead it has to be manually installed as an Add-on.

The screenshot shows the 'Deployment Overview' page for a MongoDB 3.2.10 database. The left sidebar includes links for Overview, Resources, Browser, Backups, Jobs, Settings, Metrics, Logfiles, Access, Security, and Add-ons. The 'Add-ons' link is circled in red. The main panel displays database details: Location (us-east-1 on aws), Status (Healthy), Connection (TCP Available), Usage (2GB of 2GB Disk), Billing (\$53.50/month), Storage Engine (mmapv1), Access (Admin), and Notes (Click to add some notes).

The screenshot shows the 'Add-ons' page. The left sidebar includes links for Overview, Resources, Browser, Backups, Jobs, Settings, Metrics, Logfiles, Access, Security, and Add-ons. The 'Add-ons' section lists supported add-ons: Syslog-NG, Oplog Access, and Telegraf. The 'Oplog Access' section is highlighted with a red box and contains the text: 'Access mongo's oplog via a haproxy portal to the shard's primary.' A 'Configure' button is visible next to this section. To the right, there is a 'Add-on Capsules' section with descriptive text about add-on capsules.

After installation, the Oplog url has to be added in accordance with the Environment variables section.



Galaxy

Regarding scaling, we have experienced some issues when running the app on more than two containers, so use it with caution.

A script, `./bin/deploy/deploy`, has been made available to ease up deployment, for information on how to use it there is a help page available by running `./bin/deploy/deploy --help`.

In Galaxy an independent parallel instance is called container.

The screenshot shows the Galaxy application management interface. On the left, under 'APPS', there are three app instances listed:

- The first app instance has a green circle icon and is labeled 'Deployed February 1st at 9:15pm to 3 containers by [REDACTED]'. A red arrow points to the green circle with the text 'Green flag means that the app is up and running'.
- The second app instance has a green circle icon and is labeled 'Deployed January 12th at 5:45pm to 2 containers by [REDACTED]'.
- The third app instance has a red circle icon and is labeled 'Deployed July 7th at 6:53pm to 1 container by [REDACTED]'. A red arrow points to the red circle with the text 'Red flag means the app is not working'.

On the right side of the interface, there is an 'ACTIVITY' log:

- Galaxy successfully built v329 (February 1st at 9:16pm)
- [REDACTED] deployed v329 (February 1st at 9:15pm)
- Galaxy successfully built v328 (February 1st at 7:42pm)
- [REDACTED] deployed v328 (February 1st at 7:41pm)
- Galaxy successfully built v327 (February 1st at 7:21pm)
- [REDACTED] deployed v327 (February 1st at 7:20pm)
- Galaxy successfully built v326 (January 31st at 8:05pm)
- [REDACTED] deployed v326 (January 31st at 8:04pm)

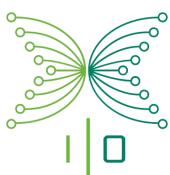
The screenshot shows the Galaxy application management interface with the 'OVERVIEW' tab selected. It displays the following resource metrics:

- Connections: 1
- CPU (ECU): 0.33
- Memory: 18%
- Containers: 3 (Standard size)

A red box highlights the 'Containers' section, and a red arrow points to it with the text 'These buttons allows to scale up and down containers. use with caution.' Below the containers section is a performance chart for 'Connections' over time (5m, 1h, 1d, 30d).

On the right side of the interface, there is an 'ACTIVITY' log:

- Galaxy successfully built v329 (February 1st at 9:16pm)
- [REDACTED] deployed v329 (February 1st at 9:15pm)
- Galaxy successfully built v328 (February 1st at 7:42pm)
- [REDACTED] deployed v328 (February 1st at 7:41pm)
- Galaxy successfully built v327 (February 1st at 7:21pm)
- [REDACTED] deployed v327 (February 1st at 7:20pm)
- Galaxy successfully built v326 (January 31st at 8:05pm)
- [REDACTED] deployed v326 (January 31st at 8:04pm)



Certificate

To secure the webpage (login credentials etc), SSL should be used. This is enabled by uploading a certificate via the settings tab. In order to force use of SSL the checkbox “Always use HTTPS” can be checked. Galaxy also has a feature to auto generate a SSL certificate if needed.

To set up a custom certificate follow the instructions below.

You are currently using 3 Standard containers. Scale your app by changing the size of containers now.

DOMAINS & ENCRYPTION

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS `us-east-1.galaxy-ingress.meteor.com`. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains	Encryption	Primary Domain
[REDACTED] ⓘ	Enabled	

+ Add new domain

DANGER ZONE

Stop app
Stopping an app shuts down all of its containers. When stopped, the app's settings are preserved and organization members may continue to view app logs and performance metrics.

DELETE

You are currently using 3 Standard containers. Scale your app by changing the size of containers now.

DOMAINS & ENCRYPTION

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS `us-east-1.galaxy-ingress.meteor.com`. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains	Encryption	Primary Domain
test.johk.io	Enabled	

Automatic Certificate

Common name: [REDACTED]
Expiry: Auto-renewing

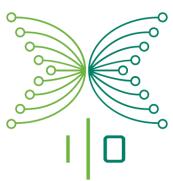
Click here to replace the automatically generated certificate for a custom SSL certificate.
Replace with custom certificate Remove

Force HTTPS

Always use HTTPS on [REDACTED]. When enabled, new incoming clients will transparently redirect to HTTPS. Existing connected clients will not be affected.

CLOSE

+ Add new domain



Sales App | Frontend - System setup

You are currently using 3 Standard containers. Scale your app by changing the size of containers now.

DOMAINS & ENCRYPTION

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS us-east-1.galaxy-ingress.meteor.com. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains

Encryption

Primary Domain

UPLOAD CERTIFICATE

SSL Key & Certificate

Key: [Choose File](#)
Certificate: [Choose File](#)

Upload here your SSL Key and Certificate

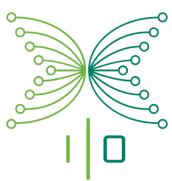
CANCEL UPLOAD CERTIFICATE

+ Add new domain

DANGER ZONE

Stop app

STOP

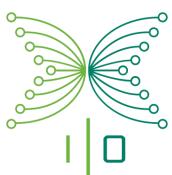


AWS S3

The application requires file storage system to store user's documents. We use AWS S3 as an host.

1. To use AWS S3 you will first need to sign up for AWS. Visit <https://aws.amazon.com/s3/> and press "Sign Up"





2. Choose your account details and press the “Sign in using our secure server”



Sign In or Create an AWS Account

What is your email (phone for mobile accounts)?

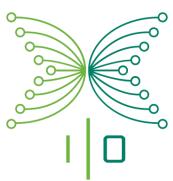
E-mail or mobile number:

I am a new user.

**I am a returning user
and my password is:**

Sign in using our secure server 

[Forgot your password?](#)



3. Fill in your details and create the account.



Login Credentials

Use the form below to create login credentials that can be used for AWS as well as Amazon.com.

My name is:

My e-mail address is:

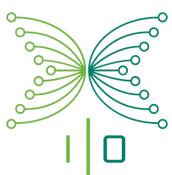
Type it again:

note: this is the e-mail address that we
will use to contact you about your
account

Enter a new password:

Type it again:

Create account



4. Fill in your contact information and create the account

Contact Information

Company Account Personal Account

* Required Fields

Full Name*

Company Name*

Country*

United States ▾

Address*

Street, P.O. Box, Company Name, c/o

Apartment, suite, unit, building, floor, etc.

City*

State / Province or Region*

Postal Code*

Phone Number*

Security Check



[Refresh Image](#)

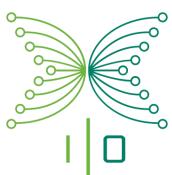
Please type the characters as shown above

AWS Customer Agreement



Check here to indicate that you have read and agree to the terms of the [AWS Customer Agreement](#)

[Create Account and Continue](#)



5. Enter your payment information



Payment Information

Please enter your payment information below. You will be able to try a broad set of AWS products for free via the Free Tier. We will only bill your credit or debit card for usage that is not covered by our Free Tier.

› [Frequently Asked Questions](#)

Credit/Debit Card Number	Expiration Date
<input type="text"/>	<input type="text"/> 03 ▾ <input type="text"/> 2017 ▾

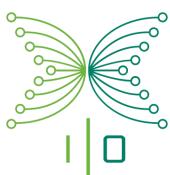
Cardholder's Name

Use my contact address

Use a new address

[Continue](#)

6. The next step requires phone verification, please follow the steps presented on screen. You will be called by Amazon to verify.



7. Please choose the appropriate support plan for your needs.

Support Plan

AWS Support offers a selection of plans to meet your needs. All plans provide 24x7 access to customer service, AWS documentation, whitepapers, and support forums. For access to technical support and additional resources to help you plan, deploy, and optimize your AWS environment, we recommend selecting a support plan that best aligns with your AWS usage.

Please Select One

Basic

Description: Customer Service for account and billing questions and access to the AWS Community Forums.

Price: Included

Developer

Use case: Experimenting with AWS

Description: One primary contact may ask technical questions through Support Center and get a response within 12–24 hours during local business hours.

Price: Starts at \$29/month (scales based on usage)

Business

Use case: Production use of AWS

Description: 24x7 support by phone and chat, 1-hour response to urgent support cases, and help with common third-party software. Full access to AWS Trusted Advisor for optimizing your AWS infrastructure, and access to the AWS Support API for automating your support cases and retrieving Trusted Advisor results.

Price: Starts at \$100/month (scales based on usage)

Enterprise

Use case: Mission-critical use of AWS

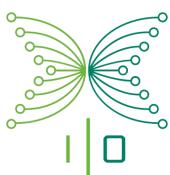
Description: All the features of the Business support plan, plus an assigned Technical Account Manager (TAM) who provides proactive guidance and best practices to help plan, develop, and run your AWS solutions, a Support Concierge who provides billing and account analysis and assistance, access to Infrastructure Event Management to support product launches, seasonal promotions/events, and migrations, and 15-minute response to critical support cases with prioritized case handling.

Price: Starts at \$15,000/month (scales based on usage)

If you select this option, customer support will contact you within 48 hours to discuss your needs and finalize the signup. Support resources will be available when signup is finalized, and no charges will be incurred until that time.

To explore all features and benefits of AWS Support, including plan comparisons and pricing samples, [click here](#).

Continue



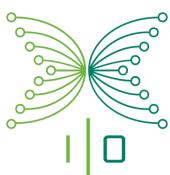
Sales App | Frontend - System setup

8. To enable S3, click “Services” and select S3.

The screenshot shows the AWS Services dashboard. On the left, there's a sidebar with 'History' and 'Console Home'. Under 'S3', there are several services listed: Compute (EC2, EC2 Container Service, Lightsail, Elastic Beanstalk, Lambda, Batch), Storage (S3, EFS, Glacier, Storage Gateway), Database (RDS, DynamoDB, ElastiCache, Redshift), and Networking & Content Delivery (VPC). The 'S3' service is highlighted with a red box. The main pane displays various AWS services categorized into groups: Developer Tools (CodeCommit, CodeBuild, CodeDeploy, CodePipeline, X-Ray), Analytics (Athena, EMR, CloudSearch, Elasticsearch Service, Kinesis, Data Pipeline, QuickSight), Application Services (Step Functions, SWF, API Gateway, Elastic Transcoder), Messaging (Simple Queue Service, Simple Notification Service, SES), Artificial Intelligence (Lex, Polly, Rekognition, Machine Learning), Internet Of Things (AWS IoT), Contact Center (Amazon Connect), and Business Productivity (WorkDocs, WorkMail, Amazon Chime). A 'Group' button and an 'A-Z' link are at the top right of the main pane.

9. Create a bucket by clicking “Create bucket”

The screenshot shows the Amazon S3 console. At the top, there's a banner that says "Want to manage your data based on what it is instead of where it's stored? Try S3 Object Tagging". Below the banner, the 'Amazon S3' logo is on the left, followed by three buttons: '+ Create bucket', 'Delete bucket', and 'Empty bucket'. To the right of these buttons are links to 'Switch to the old console', 'Discover the new console', and 'Quick tips', along with dropdown menus for '- Buckets' and '- Regions'. In the center, a message says "You do not have any buckets. Here is how to get started with Amazon S3." Below this message are three sections: "Create a new bucket" (with an icon of a bucket and cloud), "Upload your data" (with an icon of a bucket and an upload arrow), and "Set up your permissions" (with an icon of two user profiles and a plus sign). Each section has a brief description and a "Learn more" link.



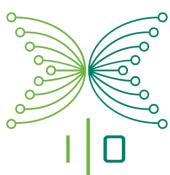
10. Input bucket name and click “Create”

The screenshot shows the 'Create bucket' dialog box over a blurred background of the AWS S3 console. The dialog has four tabs: 1. Name and region (selected), 2. Set properties, 3. Set permissions, and 4. Review. The 'Name and region' section contains fields for 'Bucket name' (containing 'salesapp') and 'Region' (set to 'Asia Pacific (Singapore)'). Below this is a section for 'Copy settings from an existing bucket' which says 'You have no buckets'. At the bottom are 'Create', 'Cancel', and 'Next' buttons.

In order to use S3 buckets for the application, AWS credentials are needed.

11. On console menu, click your name on the top right and click “My Security Credentials”

The screenshot shows the 'Your Security Credentials' page in the AWS IAM console. The left sidebar lists navigation options like Dashboard, Groups, Users, Roles, Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main content area is titled 'Your Security Credentials' and describes how to manage credentials. It lists several items with plus signs: Password, Multi-Factor Authentication (MFA), Access Keys (Access Key ID and Secret Access Key), CloudFront Key Pairs, X.509 Certificates, and Account Identifiers. A vertical navigation bar on the right includes links for My Account, My Organization, My Billing Dashboard, My Security Credentials (which is underlined to indicate it's selected), and Sign Out.



12. Under Access Keys, click “Create New Access Key” to create an access key.

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the [IAM Console](#).

To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#) in AWS General Reference.

+	Password
+	Multi-Factor Authentication (MFA)
-	Access Keys (Access Key ID and Secret Access Key)

You use access keys to sign programmatic requests to AWS services. To learn how to sign requests using your access keys, see the [signing documentation](#). For your protection, store your access keys securely and do not share them. In addition, AWS recommends that you rotate your access keys every 90 days.

Note: You can have a maximum of two access keys (active or inactive) at a time.

Created	Deleted	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
Create New Access Key							

⚠ Important Change - Managing Your AWS Secret Access Keys

As described in a [previous announcement](#), you cannot retrieve the existing secret access keys for your AWS root account, though you can still create a new root access key at any time. As a [best practice](#), we recommend creating an IAM user that has access keys rather than relying on root access keys.

+	CloudFront Key Pairs
+	X.509 Certificates
+	Account Identifiers

13. Please copy or download the keys, you will have to place these in the application's setting files.

Create Access Key

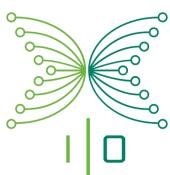
Your access key (access key ID and secret access key) has been created successfully.

Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

► Show Access Key

[Download Key File](#) [Close](#)



S3 configuration in file: “/config/develop.json”

```
"SLINGSHOT_S3_BUCKET": "*****",
"SLINGSHOT_MAX_FILE_SIZE_MB": 10,
"SLINGSHOT_S3_REGION": "ap-northeast-1",

"AWSAccessKeyId": "*****",
"AWSSecretAccessKey": "*****",
```

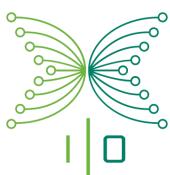
Loggly

For enabling logs on the server actions, it's required to add in Loggly configurations in secrets file, located at “/config/\${mode}/secret.json”. “Enabled” field should be set to true and you should add in all required authentication credentials.

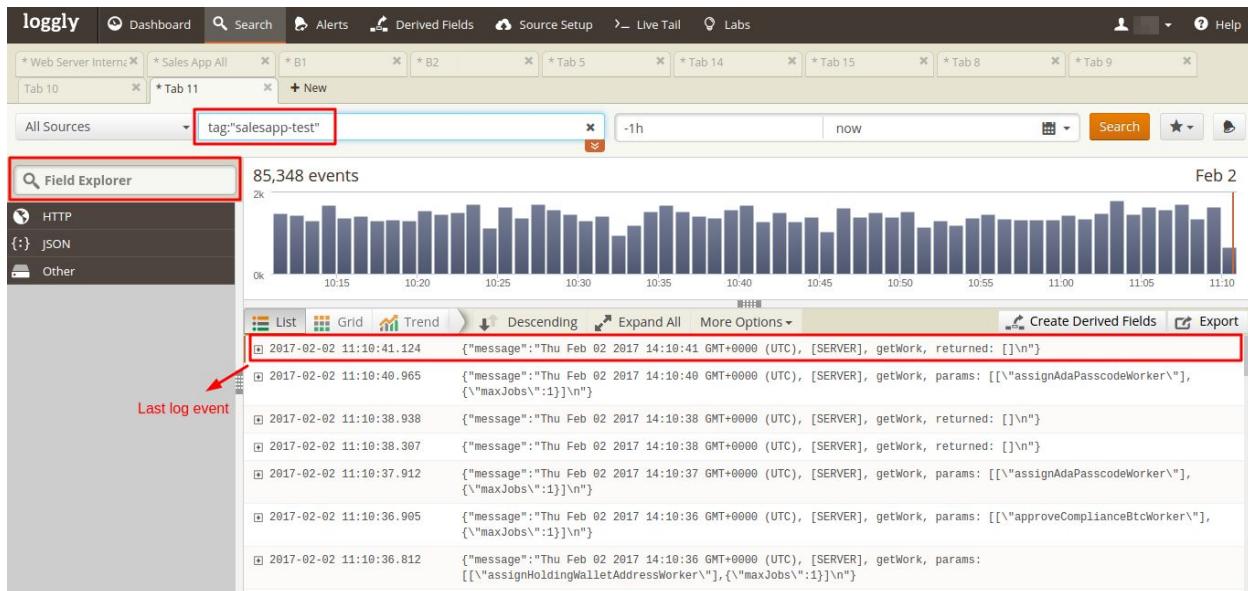
Loggly configuration in file: “/config/test/secret.json”

```
{
  "loggly": {
    "enabled" : true,
    "token": "*****_****_****_****_*****",
    "subdomain": "*****.loggly.com",
    "tags" : ["salesapp-test"],
    "auth": {
      "username": "*****",
      "password": "*****"
    }
  }
}
```

After the deploy is finished, if Loggly configuration was successful, you should be able to access the web page (in this example the url would be https://*****.loggly.com) and check that the logs are working.



Loggly search section:

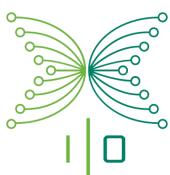


Mailgun

In order to allow Sales App to send emails using Mailgun service, it's needed to set the `MAILGUN_DOMAIN`, and `MAILGUN_API_KEY` as environment variables for Galaxy. They can be set in `"/bin/${mode}/secret.json"` file (the "mode" can be "production" or "test" depending on which environment will it be deployed).

Mailgun configuration in file: `"/config/test/secret.json"`

```
{  
  "galaxy.meteor.com": {  
    "env": {  
      "MAILGUN_DOMAIN": "*****",  
      "MAILGUN_API_KEY": "key-*****"  
    }  
  }  
}
```



Recommendations

These are recommendations and experiences gathered by running the application through the presale of ADA.

Example Setup

This section will give an examples of setup, the setup given is the setup we are using in the development environment. The example setup will contain information of version of configuration settings that have been mentioned in this document. Example settings files and deployment scripts will also be provided.

Hosting Services Configuration

These are the settings we use for different hosting services.

Galaxy

Scale up to 3 containers, enforce https, and use a custom certificate.

Compose

Enable ssl and oplog, and use MMAPv1 as storage engine.

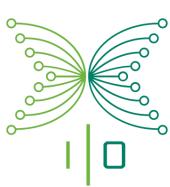
Slack

Create a custom slack channel webhook for receiving deployment notifications.

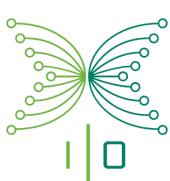
Settings

settings.json

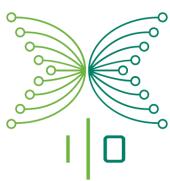
```
{  
  "public": {  
    "salesOver": false,  
    "development": false,  
    "salesStarted": true,  
    "expireTickets": {  
      "bank": {  
        "afterInvoiceSent": {  
          "days": 3,  
          "businessDays": true  
        }  
      },  
      "btc": {}  
    }  
  }  
}
```



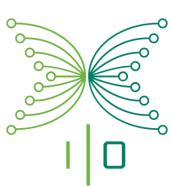
```
"afterInvoiceSent": {
    "days": 3,
    "businessDays": false
},
},
"acceptInviteBtcTimer": 1800,
"minUsdOrderValue": 1000,
"persistent_session": {
    "default_method": "persistent"
},
"salesLimits": {
    "currentTranche": "t4",
    "tranches": {
        "t1": {},
        "t2": {},
        "t3": {},
        "t3.5": { "totalAmountAvailable": 8000000, "overCapacityTolerance": 0.25 },
        "t4": { "totalAmountAvailable": 30000000, "overCapacityTolerance": 0.20 }
    }
},
"availableCountries": ["JP", "KH", "CN", "KR", "PH", "TH", "VN", "MY", "TW"],
"availableLanguages": ["en", "ja", "ko", "zh"],
"residenceCountriesAllowedSale": ["JP", "KH", "CN", "KR", "PH", "TH", "VN", "MY", "TW"],
"support": {
    "ja": {
        "email": "support@*****"
    },
    "ko": {
        "email": "korean@*****"
    },
    "zh": {
        "email": "chinese@*****"
    },
    "default": {
        "email": "support@*****"
    }
},
"regionalMappings": {
    "JP": "regionNippon",
    "KR": "regionSouthKorea",
},
"regionalSettings": {
    "regionNippon": {
        "enrollment": {
            "paymentOptions": ["bank", "btc"],
            "paymentMinimum": 1000
        },
    }
},
```



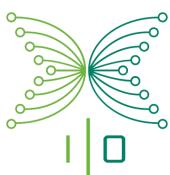
```
"latestPolicyUpdate": {
    "buyer": {
        "TOC": "2016-11-26"
    },
    "distributor": {
        "0": "2016-02-04",
        "1": "2016-02-04",
        "2": "2016-02-04",
        "3": "2016-02-04"
    },
    "PRIVACY": "2015-10-01"
},
"policies": {
    "TOC": {
        "distributor": {
            "0": [{ "lang": "ja", "file": "appoiu.pdf" }],
            "1": [{ "lang": "ja", "file": "t1k4j3kkjh4l3k4.pdf" }],
            "2": [{ "lang": "ja", "file": "t2j3j9dadflk34.pdf" }],
            "3": [{ "lang": "ja", "file": "t3y4y2y5uhkjh.pdf" }]
        },
        "buyer": [
            {
                "lang": "ja",
                "file": "toCforbuyerJP_v5a_jp_userpoli.pdf"
            }
        ],
        "default": [
            {
                "lang": "ja",
                "file": "toCforbuyerJP_v5a_jp_userpoli.pdf"
            }
        ]
    },
    "RISK": {
        "default": [
            {
                "lang": "ja",
                "file": "riskPolicyJP_v2a_jp_rskpoli.pdf"
            }
        ],
        "PRIVACY": {
            "default": [
                {
                    "lang": "ja",
                    "file": "privacy.pdf"
                }
            ]
        }
    }
},
"regionSouthKorea": {
    "latestPolicyUpdate": {
        "buyer": {
            "TOC": "2016-11-26"
        },
        "PRIVACY": "2015-10-01"
    }
}
```



```
"distributor": {
    "0": "2016-09-20",
    "1": "2016-09-20",
    "2": "2016-09-20",
    "3": "2016-09-20"
},
"PRIVACY": "2016-09-20"
},
"enrollment": {
    "paymentOptions": ["btc"],
    "paymentMinimum": 1000
},
"policies": {
    "TOC": {
        "distributor": {
            "0": [
                {
                    "lang": "en",
                    "file": "kr-toc-dist-t0.pdf"
                },
                {
                    "lang": "en",
                    "file": "kr-toc-dist-t1.pdf"
                },
                {
                    "lang": "en",
                    "file": "kr-toc-dist-t2.pdf"
                },
                {
                    "lang": "en",
                    "file": "kr-toc-dist-t3.pdf"
                }
            ],
            "1": [
                {
                    "lang": "en",
                    "file": "toCforbuyerEN_v5a_en_userpoli.pdf"
                }
            ],
            "2": [
                {
                    "lang": "en",
                    "file": "toCforbuyerEN_v5a_en_userpoli.pdf"
                }
            ],
            "3": [
                {
                    "lang": "en",
                    "file": "riskpolicyEN_v2a_en_rskpoli.pdf"
                }
            ]
        },
        "buyer": [
            {
                "lang": "en",
                "file": "toCforbuyerEN_v5a_en_userpoli.pdf"
            }
        ],
        "default": [
            {
                "lang": "en",
                "file": "toCforbuyerEN_v5a_en_userpoli.pdf"
            }
        ]
    },
    "RISK": {
        "default": [
            {
                "lang": "en",
                "file": "riskpolicyEN_v2a_en_rskpoli.pdf"
            }
        ]
    },
    "PRIVACY": {
```



```
"default": [{  
    "lang": "ko",  
    "file": "privacy-kr.pdf"  
}],  
}  
},  
"regionOther": {  
    "enrollment": { "paymentOptions": ["btc"], "paymentMinimum": 1000 }  
}  
}  
},  
"dollarsPerAda": 0.0026,  
"disableJobCollectionLogger" : false,  
  
"btcPriceCriteria": {  
    "providersRanking": [  
        "bitstamp",  
        "blockchain",  
        "bitpay",  
        "bitgo",  
        "bitfinex",  
        "kraken"  
    ],  
    "outdatedMaxToleranceInMinutes": 2  
}  
}
```

*secret.json*

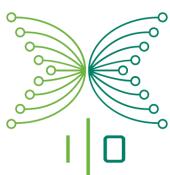
```
{
  "SLINGSHOT_S3_BUCKET": "*****",
  "SLINGSHOT_MAX_FILE_SIZE_MB": 10,
  "SLINGSHOT_S3_REGION": "ap-northeast-1",

  "AWSAccessKeyId": "*****",
  "AWSSecretAccessKey": "*****",
  "SALES_APP_AUTH_API_TOKEN": "*****_*****_*****_*****",

  "loggly": {
    "enabled" : true,
    "token": "*****_****_****_****_*****",
    "subdomain": "*****.loggly.com",
    "tags" : ["salesapp-test"],
    "auth": {
      "username": "*****",
      "password": "*****"
    }
  },

  "kadira": {
    "appId": "*****",
    "appSecret": "*****"
  },

  "galaxy.meteor.com": {
    "env": {
      "ROOT_URL": "http://*****",
      "MONGO_URL": "mongodb://*****@*****:*****?ssl=true&authMechanism=SCRAM-SHA-1",
      "MONGO_OPLOG_URL": "mongodb://oploguser:*****@*****:*****/local?authSource=admin&ssl=true",
      "INVOICE_FOLDER": "./invoices/",
      "MAILGUN_DOMAIN": "*****",
      "MAILGUN_API_KEY": "key-*****",
      "IOHK_BACKEND_SERVER_IP": "https://*****/api/v1/",
      "ATTAIN_BCC_EMAIL": "test-email+test-attain@iohk.io",
      "IOHK_BACKEND_CLIENT_ID": "SalesAppTesting",
      "IOHK_BACKEND_TOKEN": "*****",
      "IOHK_BACKEND_SIGNATURE": "*****"
    }
  }
}
```



Scripts

To ease up deployment to testing and development environment we used following scripts. They can be modified to suite any environment similar to our. The scripts are located at `./bin/deploy` folder, and they are wrapping the more general deploy script.

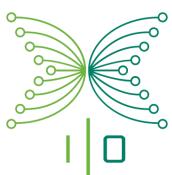
<code>to-production-config</code>	Creates a configuration file for production. The file will be save in the temp folder and the folder will be printed in the terminal.
<code>to-production-deploy</code>	Deploy to production using configuration file, which needs to be passed in as a parameter. You can use <code>to-production-config</code> to generate this config file.
<code>to-production-no-config</code>	Deploy to production without updating the configuration file.
<code>to-test-attain-iohk-io</code>	Deploys to test.attain.iohk.io server with the default settings file. Since this script is setting test mode, the minimum purchase is overwritten to become 1 USD.
<code>to-test-iohk-io</code>	Deploys to test.iohk.io server with the default settings file. Since this script is setting test mode, the minimum purchase is overwritten to become 1 USD.

Known Quick Fixes

The application is not always running smoothly when scaled to run multiple instances in parallel against the same database. If it's still desireable to do so, then these quick fixes have proven to solve the situation regardless of what the initial problem was for a number of times. However, there is no guarantee they will work, and in some cases they might cause more issues than they solve.

Scaling Down and Up Container

This fix assumes use of Galaxy as cloud host. By simply scaling down the container size by one container, and then scaling up again some issues seems to disappear magically. This is really useful when a worker has stalled (see System operation in Frontend - Business logic).



Scale Down

Press the down arrow button to scale down the amount of containers by one.

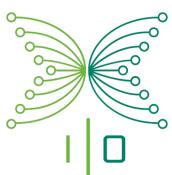
The screenshot shows the Galaxy application interface. At the top, there's a navigation bar with tabs: OVERVIEW, CONTAINERS (which is selected), LOGS, and SETTINGS. Below the navigation bar, the main area is divided into sections: CONTAINERS, PERFORMANCE, and ACTIVITY.

CONTAINERS: This section displays the current state of the application. It shows 2 Running containers and 0 Unavailable containers. A red arrow points to the number '2' with the text "Scaling from 3 to 2 containers". Below this, a progress bar is labeled "Scaling containers down" and "PROGRESS BAR" with "0/1 container".

PERFORMANCE: This section contains performance metrics and a timeline chart. Metrics include CPU (ECU) usage (~1.0), Memory (MB) usage (~1024), and Network connections (~1). The timeline chart shows CPU usage over time from 2:54 PM to 2:56 PM.

ACTIVITY: This section lists deployment and build history. Recent entries include:

- Galaxy successfully built v329 (February 1st at 9:16pm)
- Galaxy successfully built v328 (February 1st at 7:42pm)
- Galaxy successfully built v327 (February 1st at 7:21pm)
- [redacted] deployed v327 (February 1st at 7:41pm)
- [redacted] deployed v326 (February 1st at 9:15pm)
- [redacted] scaled [redacted] down to 2 containers a few seconds ago



Scale Up

Press the up arrow button to scale up the amount of containers by one.

The screenshot shows the Galaxy application interface. At the top, there's a header with 'GALAXY' and 'Docs'. Below the header, a message says 'Version 329 Deployed February 1st at 9:15pm by [redacted]' and '3 containers'. The main area has tabs for 'OVERVIEW', 'CONTAINERS', 'LOGS', and 'SETTINGS'. Under 'CONTAINERS', it shows 'Containers Standard' with 3 total, 2 running, and 0 unavailable. A red box highlights a progress bar labeled 'Scaling containers up' with '0/1 container'. To the right, there's an 'ACTIVITY' log with several entries, including deployment and scaling events. On the left, there's a 'PERFORMANCE' section with metrics like CPU (ECU) and Memory (MB). A vertical green bar on the far right contains the text 'Frontend System Setup'.

Scaling from 2 to 3 containers

PROGRESS BAR

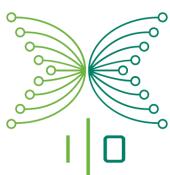
0/1 container

ACTIVITY

Scaling containers up

PERFORMANCE

Frontend System Setup



Restart the Application

This quickfix has so far fixed any issue not solved by scaling up and down containers (see section above), but with the drawback that the application will go offline for a couple of minutes. Simply shut down all running instances and start them up again one by one. The steps below is for doing it in Galaxy.

Stop the containers

Navigate to the bottom of the settings page.

You are currently using 3 Standard containers. Scale your app by changing the size or containers now.

DOMAINS & ENCRYPTION

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS `us-east-1.galaxy-ingress.meteor.com`. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains	Encryption	
[REDACTED] ⓘ	Enabled	Primary Domain
+ Add new domain		

DANGER ZONE

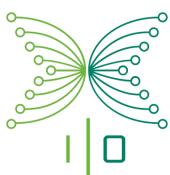
Stop app
Stopping an app shuts down all of its containers. When stopped, the app's settings are preserved and organization members may continue to view app logs and performance metrics.

Delete app
Deleting an app shuts down all containers and removes the app from Galaxy.

Click here 

STOP

DELETE



Sales App | Frontend - System setup

Click “stop” followed by “confirm stopping”.

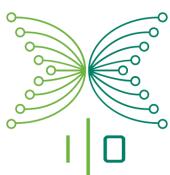
Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS `us-east-1.galaxy-ingress.meteor.com`. We recommend that you also enable encryption for your domains to secure sensitive data.

The screenshot shows the Galaxy app management interface. At the top, there are sections for 'Domains' (with a placeholder domain and an 'Add new domain' button), 'Encryption' (set to 'Enabled'), and 'Primary Domain'. Below these, a 'DANGER ZONE' modal is open, prompting the user to 'Stop app'. The modal contains the text: 'Stopping an app shuts down all of its containers. When stopped, the app's settings are preserved and organization members may continue to view app logs and performance metrics.' It includes a question 'Are you sure you'd like to stop test.iohk.io?' with 'CANCEL' and 'STOP APP' buttons, where 'STOP APP' is highlighted with a red arrow. Below the modal, another section for 'Delete app' is visible, with a 'DELETE' button.

Navigate to the containers tab

The screenshot shows the Galaxy app management interface with the 'CONTAINERS' tab selected. A prominent message 'The containers are stopping!' with an exclamation mark is displayed. Below this, the 'CONTAINERS' section shows three containers: 'Containers Standard' (3), 'Running' (0), and 'Unavailable' (0). A progress bar indicates 'Scaling containers down'. The 'PERFORMANCE' section displays metrics over time (5m, 1h, 1d, 30d) for CPU (ECU) and Memory (MB). The CPU usage is shown as a blue line graph with values around 1.4 and 0.7, and the memory usage is shown as a green line graph with values around 1024 and 512. On the right, the 'ACTIVITY' log shows deployment and scaling events for version v329 and v328. A red arrow points to the 'Stopping status' message.

Wait for the containers to stop



GALAXY

Containers are fully stopped!

Version 329 Deployed February 1st at 9:15pm by [REDACTED] 3 containers

OVERVIEW CONTAINERS LOGS SETTINGS

CONTAINERS

3 Standard Containers

0 Running 0 Unavailable

Empty status

PERFORMANCE

Showing all containers

5m 1h 1d 30d

ACTIVITY

- [REDACTED] stopped [REDACTED] 2 minutes ago
- [REDACTED] scaled [REDACTED] up to 3 containers 27 minutes ago
- [REDACTED] scaled [REDACTED] down to 2 containers 28 minutes ago
- ✓ Galaxy successfully built v329 February 1st at 9:16pm
- ✓ [REDACTED] deployed [REDACTED] v329 February 1st at 9:15pm
- ✓ Galaxy successfully built v328 February 1st at 7:42pm
- ✓ [REDACTED] deployed [REDACTED] v328

Start the containers

Navigate to the bottom of the settings page and click “start”.

Container Size: Standard

You are currently using 3 Standard containers. Scale your app by changing the size of containers now.

CHANGE

DOMAINS & ENCRYPTION

Apps deployed to Galaxy get a unique Galaxy Domain. Add custom domains to make your app accessible via other domain names then configure your DNS provider to point at Galaxy's DNS us-east-1.galaxy-ingress.meteor.com. We recommend that you also enable encryption for your domains to secure sensitive data.

Domains

Encryption

Enabled

Primary Domain

+ Add new domain

DANGER ZONE

Start app

Start [REDACTED] with 3 containers

Click here

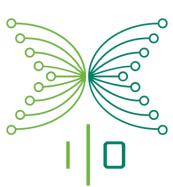
START

Delete app

Deleting an app shuts down all containers and removes the app from [REDACTED] Galaxy.

DELETE

Navigate to the containers tab and wait for the containers to start up again.



Sales App | Frontend - System setup

The screenshot shows the Galaxy application interface for a Sales App Frontend. At the top, there's a header with a gear icon, the word "GALAXY", and navigation links for "Docs" and "CR". Below the header, a message says "The containers are starting up!" with a red arrow pointing to a green circular progress bar.

Below the message, it says "Version 329 Deployed February 1st at 9:15pm by [redacted] 3 containers". The main interface has tabs for "OVERVIEW", "CONTAINERS", "LOGS", and "SETTINGS".

The "CONTAINERS" section shows the following status:

- Containers: 3 (Standard)
- Running: 0
- Unavailable: 0

A progress bar indicates "Scaling containers up" from 0/3 to 0/3 containers. A red arrow points to this bar with the text "Loading status".

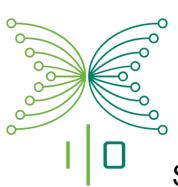
The "PERFORMANCE" section is highlighted with a red box and shows metrics for container f3zb in us-east-1a:

Connections	CPU (ECU)	Memory (MB)
~1	~1.4	~1024
~1	~0.7	~512

Time markers at the bottom of the performance chart are 3:22 PM, 3:24 PM, 3:22 PM, 3:24 PM, 3:22 PM, and 3:24 PM.

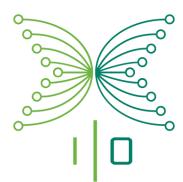
The right side of the interface shows an "ACTIVITY" log with the following entries:

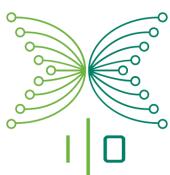
- Started [redacted] a minute ago
- Stopped [redacted] 3 minutes ago
- Scaled [redacted] up to 3 containers 26 minutes ago
- Scaled [redacted] down to 2 containers 29 minutes ago
- Galaxy successfully built v229 February 1st at 9:16pm
- Deployed [redacted] v329
- Galaxy successfully built v328



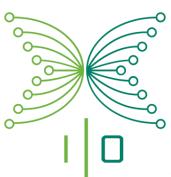
Sales App | Frontend - System setup

Frontend - Technical Documentation

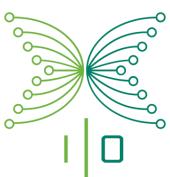




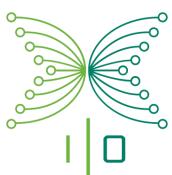
Development System Requirements	6
Software	6
Hardware	6
Code Structure	7
Naming Convention	7
Folder System	8
Important Files	9
Client Router	9
Invoice State Machine	9
API Endpoints	10
System Boot	10
Code Base Metrics	11
LOC	11
Files	13
Commits	14
Code Contributors	15
Contributions per person (excluding merge commits)	15
Test Cases	16
Development Process	17
Workflow	17
Github Projects, Trello	17
What is Trello	17
What is Github Projects	17
Trello	17
Github Projects	18
Why We Use Trello and Github Projects	18
Task, Story Cards	18
Story	18
Story Format	19
Task	19
Format	20
Lanes	21
MoSCoW	22
Scrum	22
What is Scrum	23



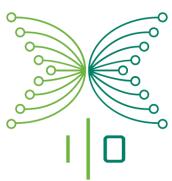
Main Features of Scrum	23
Scrum in Our Project	23
Kanban	23
What is Kanban	24
Kanban in Our Project	24
Weekly Development Meeting	25
Git	25
Gitflow	26
Branches	26
Commenting	27
Git Comments	27
Pull Requests	29
Testing	31
Setting Up Your System	32
Unit Test	32
Integration	32
End-to-End	33
Running the Tests	33
Running	33
Running Tests Locally	35
Running Tests on CircleCI	35
API	35
Backend	36
JSON Generator	41
Set Invoice Ticket PII Hash	41
Headers	41
Body	41
Response Codes	43
Advanced Features	45
Jobs and Workers	45
Ada Reservation	47
Sales Predictor	48
Invoice Ticket State Machine	50
Refcodes	55
Router Validation	55
Languages, Standards and Frameworks	61



Standards	61
JSON	61
HTTP	61
i18n	61
DDP	65
Languages	67
ECMAScript 6	67
HTML	67
LESS	68
Cucumber	70
Shell Script	71
Perl	71
PowerShell	72
Frameworks	72
Meteor	72
Blaze	73
Node	75
Mocha.js	77
Chai.js	78
Chimp.js	79
Databases	79
MongoDB	80
Improvement Suggestions	81
Test Suite Improvement	81
Middleman Attack Yielding	81
Toggle Sorting Button for Invoice Managers	82
Block Password Reset Emails for X Amount of Time	82
Turn Pay Page into an Invoice Ticket Status Page	82
Show The Buyer's Residence Country on the Invoice Ticket Card in the Invoice Queue	82
Add Backend Status Info to Sysop Dashboard	83
Extend the Region Filter for Compliance Officers	83
Sort the Invoice List in Review User Page	84
Hover in Birth Date Comparison Table Doesn't Line Up	85
Better Ticket Information in User Summary View	86
Special Holiday Settings	87
Add Expiration for Users Pending Compliance	88
Known Bugs	89



Zip Api Check not Always Saving During Enrolment	89
When the Pricing Providers are Offline, then the Application Will Set satoshisExpected, centsAskPrice and btcUsd to 0	89
When Reviewing a User, and CO Rotates the Image, the Next Image Viewed Will Also Be Rotated	89
When You Login as an Investigator and Press “Back” After Reviewing, Misleading Message Appears	89
When Commissions are Made, the Origin’s Email are Stored in the Document and Remains Unchanged Even If User Change Their Email Address	90
During Logout or When Switching Views, Console Errors Can Appear	90
The Payment Information on the Email and Payment Link for Bank Tickets are Inconsistent	
90	
'RegenerateAdaPasscode' Button is Still Visible Even Though the Ticket is in the State Where Passcode Cannot Be Regenerated	91
Counter On the Tab and Search Result Counter Show Different Number in Compliance View	
91	
User Can Enter String on Phone Field on Enrollment	91



Development System Requirements

Software

For the development of the application we used 2 particular Operating Systems namely; MacOS and Linux (Ubuntu).

Windows is not recommended as running a local server and the test suites will cause trouble. A lot of these issues can't be solved.

The following software is required for the development of the application.

- NodeJS
- Meteor
- An editor capable of recognizing JavaScript

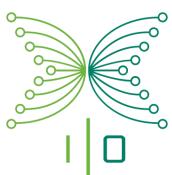
Hardware

For running a local server not much is required, the most basic hardware is able to do this although it might be slightly taxing when using big database queries.

To speed up the process of rebuilding the server a faster system would be preferred.

The following requirements are just a recommendation for a smooth development process:

- Intel i5 or similar
- 8-16GB of RAM
- Fast HDD or SSD with at least 3GB of space left
- A screen with a resolution that is 1920x1080 or more



Code Structure

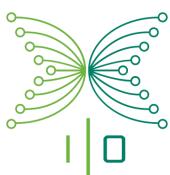
This is the way the team discussed and decided to program and structure the software. For a team, a good standard helps maintainability and keeps everything clear when others read it.

Naming Convention

Naming conventions are the set of rules when naming folder, files, functions, etc. These rules were set so that other developer can understand the code just by looking at the name instead of reading the whole code and try to understand it.

- We do not use abbreviations in names, ex use “language” instead of “lang”.
- Make names as self-describing as possible (comments should never be needed for names).

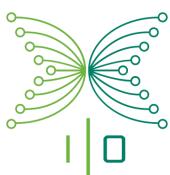
	Style	Example	Remarks
Folder	Dasherized	important-folder	-
File	Dasherized	good-file-name.js	File type or content doesn't matter.
Routes	Dasherized	my-path-name	-
Templates	Lower Camel case	remarkableTemplate	-
Variable	Lower Camel case	someVariable	Boolean variables should start with “is” or “has”.
Function	Lower Camel case	niceFunction	Don't let function start with “get”, “has”, etc edit any states.
Function Constructor	Upper Camel case	MyAwesomeClass	-
CSS classes/ids	Dasherized	my-awesome-button	-



Folder System

In this section you will find the most relevant folders in the entire application.

- ❖ **attainenroll**: Sales application main folder, it contains all the MeteorJs code.
 - **client**
 - **lib**
 - **components**: Generic components that are used multiple times.
 - **stylesheets**: Code written in [LESS](#) about the application styling.
 - **views**: [Blaze templates](#) related to the application main views.
 - **imports**
 - **client**
 - **lib**
 - **fixtures**: Model builders and factories.
 - **server**: Modules like email actions and invoice state machine.
 - **lib**
 - **collections**: Collections that are used in client and server side.
 - **locale**: Translations of the application.
 - **packages** Modified packages
 - **private**
 - **emails**: html templates for the emails.
 - en for English emails
 - ja for Japanese emails
 - ko for Korean emails
 - zh for Chinese emails
 - **public**: Application public resources like agreements pdf files.
 - **server**: Meteor calls, publications, etc.
 - **collections**: Collections that are only available in server side, like jobs collection.
 - **emails**: Server side rendering for complied email templates
 - **lib**: Server related codes.
 - **workers**: Codes for all the server side [workers](#)
 - **test**
 - **end-to-end**: Tests written with the Gherkin language and cucumber. Most tests are in subfolders separated by features.
 - **_support**: Tests helpers and common resources
 - **unit**: Gherkin and cucumber unit styles tests.
 - **integration**: Gherkin and cucumber integration styles tests.
 - **integration**: Integration tests (written with mocha).
 - **unit**: Unit tests (written with mocha).
 - ❖ **bin**: Scripts used for execute, run tests and deploy the application.



- **deploy:** Scripts for deploying the application to test and production server.
- **run:** Scripts for running the application.
 - “dev” are used to run the sales application on local machine.
- **test:** Scripts for running test of:
 - End-to-end
 - Integration
 - Unit
- **env:** Contains environment settings for running application and tests.
- **db:** Scripts for importing the db from test environment to local storage.
- ❖ **config:** Settings files for the application. It contains sensitive information, such as server secret key, access tokens, so be careful not to put this folder in public place.
- ❖ **dump:** Test environments database dumps.
- ❖ **playhooks:** YAML configuration files, and encrypted credentials for mailgun, backend, and other services.
- ❖ **scripts:** Script to run on database. Scripts include:
 - **backfill**
 - **lookup**
 - **update**
 - **validation**
 - **exports**
 - **Development**
- ❖ **UX-Slides:** Application initial mockups, could be outdated.

Important Files

Client Router

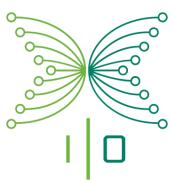
File path: "/client/router.js"

This file contains all the routes that a user can access in the Sales application. Router defines the user permissions that is needed to access to each route and the Templates that are rendered.

Invoice State Machine

File path: "/imports/server/invoice-ticket-state-machine.js"

The expected ticket states and changes are defined by the Invoice state machine. All the information about the flows that a ticket could take are in this file.



API Endpoints

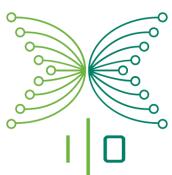
File path: "/server/router-server.js"

The API endpoints for the Sales application are defined in this file. Mostly they are used by the BackendApp or the AVVM in order to communicate special events.

System Boot

File path: "/server/init.js"

The System setup is in this file. Mainly the workers are triggered and the Logger is configured here.

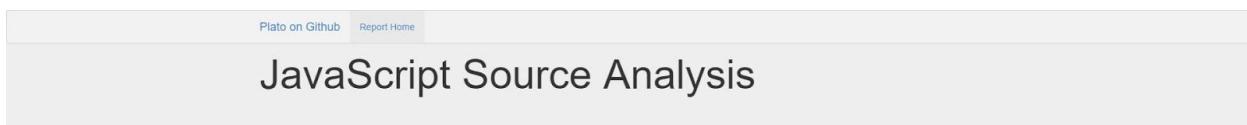


Code Base Metrics

LOC

LOC = Lines of Code

NCLOC = Non Commented Lines of Code



Summary

Total/Average Lines ⓘ

28247 / 80

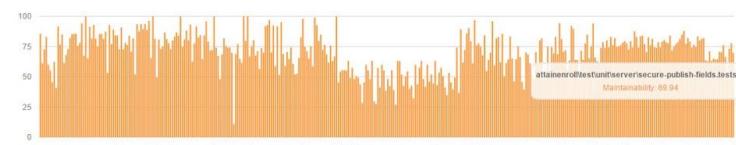


Average Maintainability ⓘ

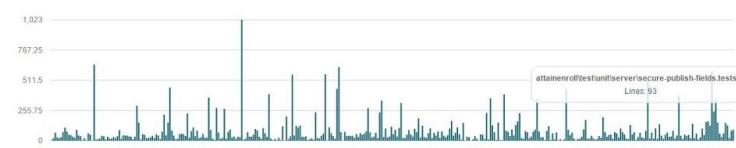
70.43



Maintainability ⓘ



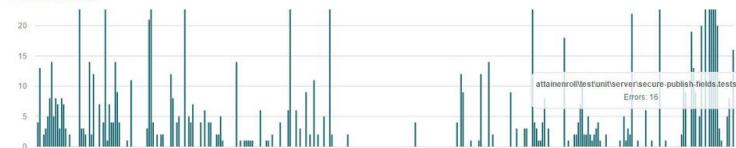
Lines of code ⓘ



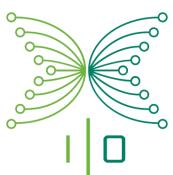
Estimated errors in implementation ⓘ



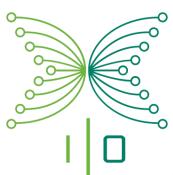
Lint errors



Files

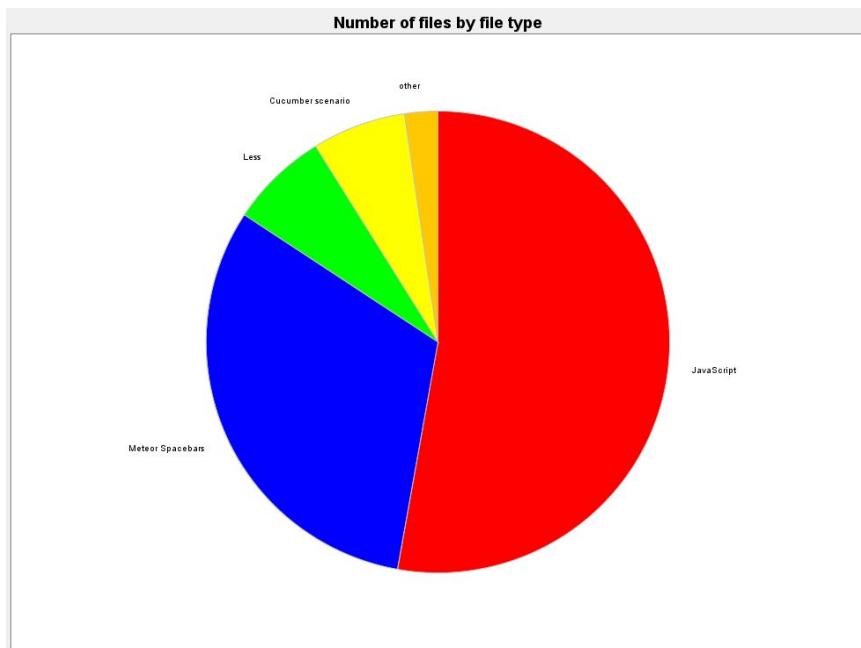


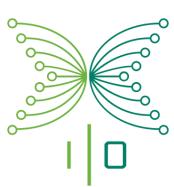
File type	LOC	NCLOC
Cascading style sheet	52	50.0
Cucumber scenario	1660	1643
JavaScript	25365	24639
JSON	16	16
JSON Schema file	63	63
Less	3119	2643
Markdown language file	4	4
Meteor Spacebars	4186	4186
XML	151	146
Total	34628	33402
Average	4328.50	4175.25



Files

File type	Amount
Cascading style sheet	3.0
Cucumber scenario	43.0
JavaScript	344.0
JSON	2.0
JSON Schema file	1.0
Less	45.0
Markdown language file	1.0
Meteor Spacebars	199.0
XML	8.0
Total	646.0
Average	80.75

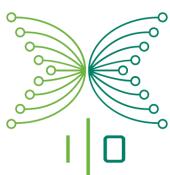




Commits

As of Release Z4 we have 6,360 commits (including merge commits).





Code Contributors

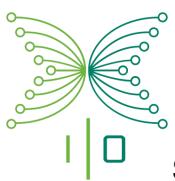
For more details visit contributors section in github

References:

- Code contributors: <https://github.com/input-output-hk/sales-frontend/graphs/contributors>

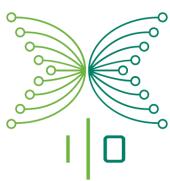
Contributions per person (excluding merge commits)

Contributor	Commits		Additions		Deletions	
*****	899	21.65%	59,928	21.07%	56,767	28.08%
*****	360	8.67%	51,699	18.17%	26,220	12.97%
*****	348	8.38%	20,517	7.21%	12,727	6.30%
*****	318	7.66%	13,951	4.90%	11,601	5.74%
*****	250	6.02%	6,446	2.27%	3,970	1.96%
*****	226	5.44%	16,693	5.87%	6,007	2.97%
*****	211	5.08%	9,606	3.38%	5,582	2.76%
*****	209	5.03%	15,867	5.58%	5,672	2.81%
*****	188	4.53%	11,347	3.99%	8,122	4.02%
*****	184	4.43%	6,018	2.12%	9,042	4.47%
*****	172	4.14%	9,784	3.44%	3,759	1.86%
*****	168	4.05%	8,938	3.14%	5,881	2.91%
*****	164	3.95%	43,642	15.34%	42,414	20.98%
*****	102	2.46%	5,428	1.91%	1,270	0.63%
*****	100	2.41%	348	0.12%	316	0.16%
*****	66	1.59%	1,016	0.36%	430	0.21%
*****	52	1.25%	698	0.25%	652	0.32%
*****	44	1.06%	866	0.30%	325	0.16%
*****	39	0.94%	553	0.19%	557	0.28%
*****	38	0.92%	730	0.26%	708	0.35%
*****	11	0.26%	401	0.14%	138	0.07%
*****	3	0.07%	0	0.00%	0	0.00%
*****	1	0.02%	7	0.00%	1	0.00%
Total	4153		284,483		202,161	



Test Cases

Test type	Amount
Unit	82
Integration	69
End-to-End	330



Development Process

Workflow

Github Projects, Trello

What is Trello

Trello is a collaboration tool that organizes your projects into boards. In one glance, Trello tells you what's being worked on, who's working on what, and where something is in a process. Here are some features that were used for our task management.

- Tags
 - Tags are used to indicate what the card is related to. If it's bug related, we add bug tag.
- Checklist
 - List of things that need to be done. This is used to check the progression of the work.
- Markdown
 - We used markdown to write down code snippets and file names.

References:

- Trello: <https://trello.com/>

What is Github Projects

Github Projects is a similar collaboration tool. But with Projects, you can manage work directly from your GitHub repositories. Create cards from Pull Requests, Issues, or Notes and organize them.

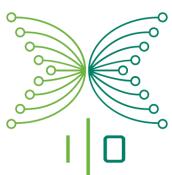
Although Trello and Github Projects have similarities, there are differences.

References:

- Github projects: <https://help.github.com/articles/about-projects/>

Trello

- Trello reactively shows if someone's editing the card. This means that team can check latest state of the card without making someone wait.
- Checklist are shown as a progress bar in the overview so it is easy for team to check the progression of the tasks.
- Trello has a mobile application so you can see the board using mobile phones, tablets.



Github Projects

- Syntax highlighting of code snippets.
- One system is a goal in itself.
- Can reference cards via commit.
- Only one person can edit a specific card at a time.
- No mobile application supported.

Why We Use Trello and Github Projects

We decided to use Trello as a board to use for planning meetings and GitHub Projects for task management of our development.

Github Projects is **not reactive** and only one person can create and edit task cards while the meeting is held. This means that most people had to wait until each card was finished, which was a waste of time.

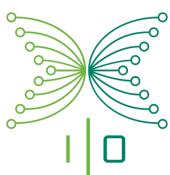
Trello on the other hand is **reactive** and multiple people can work on same card at the same time. Using Trello has made our planning smoother since we did not have to wait for someone to finish editing the cards.

Task, Story Cards

In collaboration tools we have two types of cards, **story** and **task**.

Story

A “story” is a request from the client, development team or product owner which they want to discuss and implement to the application.



Story Format

Follow the story format so that the team and product owner can understand what the issue is.

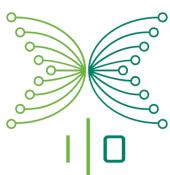
- **Tag**
 - Add a tag of who has requested it. ex. client, product owner, development team.
 - Add what kind of issue it is such as feature, bug, chore, and analysis.
- **Title**
 - Explain what the issue is in a few words.
- **Background**
 - Describe what the issues are and why is this needed to be implemented.
 - If it is user interface related, paste in a picture as well.
- **Description**
 - Describe how it can be solved. Try to add a number of possible solutions.
 - Adding pros/cons would be helpful for the team to decide which solution they should implement.
- **Comments**
 - If there was any progression to the story, write them in the comment. Team members should not edit the story itself since we want to keep the timeline trackable.

Example:

The screenshot shows a Jira ticket interface for a story titled "Show link to blockchain-info on Distributor dashboard". The ticket is labeled "Feature" and "Release W". It includes a "Description" field containing the URL "input-output-hk/sales-frontend: Issue #1479" and a "Background" section describing the need for customer service officers to see distributor Bitcoin addresses. The "Description" field also contains a note about adding a quick link to the distributor dashboard.

Task

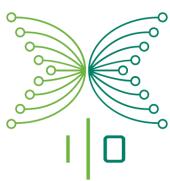
Task cards are used in development process. It is used so that the team understands what will be implemented and what they should be working on. Tasks are usually made from stories which are what the client, product owner, or development team requested.



Task cards contain more code related instructions than story cards. This is so that person taking the task would not waste their time figuring out what they actually have to do or how to implement it.

Format

- **Title**
 - Short description of what you want to accomplish.
- **Number of reviewers**
 - Number of persons to review this task. the number is decided on planning meeting.
- **Story**
 - Link URL to the Trello/Github story card.
- **Background**
 - Describe the issue and why is this task necessary.
- **Description**
 - Describe how the issue can be solved. If it is code related, write concrete instructions.
- **Files**
 - Link URL to the Github with develop branch or link may expire and we cannot see the files.
- **Refs**
 - Any reference that would help the team complete the tasks.
 - If any similar codes were used in the application, add a link to Github files.
- **Sub tasks**
 - Progression checklist of things that needs to be done to complete the task.
 - Each list is a mini task to complete the whole task. This will also be used to keep track of progression.
- **Acceptance**
 - Describe what kind of behavior is expected to see on the application after the task is complete.
 - It will be written in gherkin style so that development team can write and review them by end-to-end-test.
- **Comments**
 - If the person taking the task have any question of something he or she want to discuss with the team.
 - When you want someone else to take over the task, write what you've done and ask someone in the team to take over.



Example: This is an example of task card that implements new feature.

Show link to blockchain-info on Distributor dashboard
in list Release W - Backlog

Labels: **Feature** + Last Updated: a few seconds ago

Description [Edit](#)

Story

[Show link to blockchain-info on Distributor dashboard](#)

Description

Number of reviewers: 1
Add a quick link in the distributor dashboard where they will be able to go directly to their bitcoin wallet on [blockchain.info](#).

Files

[attainenroll/client/views/account-overview/account.html](#)
[attainenroll/client/views/account-overview/account.js](#)

Refs

None

Tasks

- [] Add link to the [blockchain.info](#) on the distributor dashboard
- [] Add end-to-end test to check if link exists

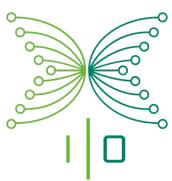
End-to-end

Scenario Outline: I should see a link to |
Given I am the approvedDistributor
When I fill login form with my account
Then I should see a link to blockchain.info

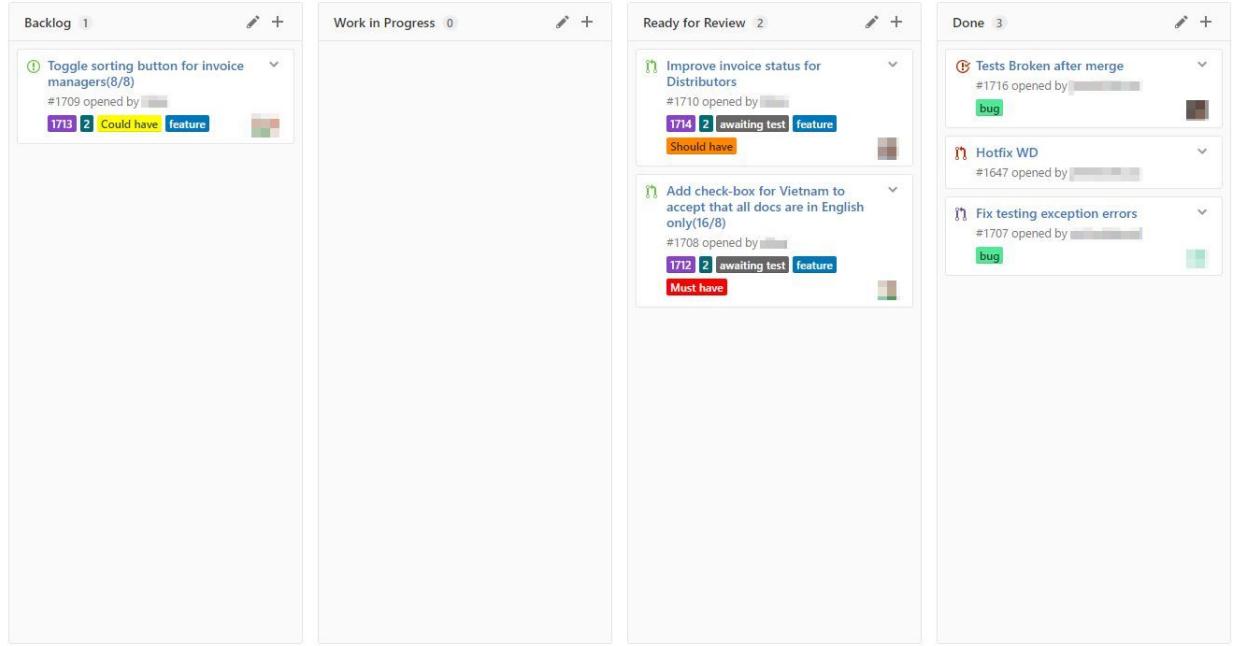
Lanes

To keep track of development tasks, we have 4 lanes. Each lane represents a stage in the development process:

- **Backlog**
 - Not started, roughly prioritized, indeterminate completion date. Development team will select a task from this lane and move them to "work in progress".
- **Work in Progress**
 - Once you've made a new branch or begun working on a task, you should move the card to the "Work in Progress" list.
- **Ready for review:**
 - When development is finished, move the card to "Ready for review" and paste a link to the pull request in the Trello card description.
- **Done**
 - Whenever you merge a pull request to develop, move the appropriate card to this list; you should only merge pull requests that belong in the next release.



Example:



MoSCoW

The MoSCoW method is a prioritization technique used for deciding which tasks are essential for the current release and which should not be prioritized. The labels used are:

Must have

The release can't go out without this task being implemented.

Should have

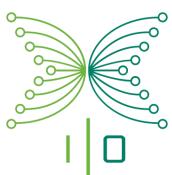
This task is not required to be in release, but implementation is highly recommended.

Could have

If this task is finished, it can go in the release else save it for another one.

Won't have

This task will not go in the release.



Scrum

What is Scrum

Scrum is an iterative and incremental agile software development framework for managing product development.

Main Features of Scrum

- A product owner creates a prioritized wish list called a product backlog.
- During sprint planning, the team pulls a small chunk from the top of that wish list, a sprint backlog, and decides how to implement those pieces.
- The team has a certain amount of time — a sprint (usually two to four weeks) — to complete its work, but it meets each day to assess its progress (daily Scrum).
- Along the way, the ScrumMaster keeps the team focused on its goal.
- At the end of the sprint, the work should be potentially shippable: ready to hand to a customer, put on a store shelf, or show to a stakeholder.
- The sprint ends with a sprint review and retrospective.
- As the next sprint begins, the team chooses another chunk of the product backlog and begins working again.

References:

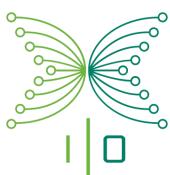
- Scrum Alliance: <https://www.scrumalliance.org/why-scrum>
- Agile Software Development: https://en.wikipedia.org/wiki/Agile_software_development

Scrum in Our Project

Until Release Y we used Scrum approach for our releases. Before each sprint we did a planning meeting in which we created task cards for current user stories, prepared by the product owner, and estimated them. The typical sprint length was 2 weeks.

Cards were prioritized with the [MoSCoW method](#) (*Must have, Should have, Could have, and Won't have*). The releases continued until most of the important cards were in. We didn't do sprint reviews. The sprint was reviewed by the client and product owner. No sprint review between developers was held.

Every day we held two standup meetings. First with Osaka/Austria team and the second one with Austria/Argentina team. During these meetings we pointed out what we did that day, the day before and what blockers we had. We tried to keep the meetings as short as possible having a target of maximum 15 minutes.



Kanban

What is Kanban

Kanban is a popular framework for agile software development with a long history. The main concept is to split task cards in 'To do', 'Doing' and 'Done' lanes. The 'Doing' lane should have a limit of cards it can contain, this forces the team to finish tasks they have already started before grabbing new cards.

Kanban and Scrum share some concepts, but there are many differences. One of them is that Kanban uses a continuous flow instead of fixed length sprints, which means it is completely okay to add new cards at any time and any undone cards are automatically pushed to the next release. The task cards in backlog are sorted based on their priority.

The main benefits of Kanban over Scrum are the ease to do bugfixes, because in Scrum, if the fix is not a hotfix, you need to wait for the next release to get it out, which in some cases can take a long time. Kanban solves this problem by doing more frequent, flexible releases. Another benefit is doing ad-hoc tasks, since you can add/prioritize tasks in the middle of sprint.

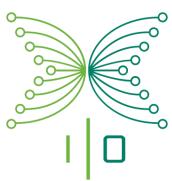
Some drawbacks of Kanban are accumulation of technical debt. The reason for it is that in Scrum, for each release, you do a planning meeting as well as a sprint review meeting and a Sprint Retrospective Meeting. Having all those meetings usually leads to a more organized code structure. Kanban also has a problem with refactoring. Since Kanban doesn't have a concept of code stop, it's hard to find a good time when to do refactoring, without interfering with other development.

Kanban in Our Project

After Release Y we decided to stop implementing new features and focus on maintenance and bugfixes. At that point we started getting more ad-hoc tasks. Since in kanban it's easier to change priorities for tasks and add new tasks in the middle of sprint, we deemed it more suitable to switch to the kanban approach.

We also did some modifications to the kanban method. For example, we continued with the planning meetings and daily standup meetings which are usually part of the Scrum workflow. Instead of going with the kanban method for prioritising cards, we stayed with the [MoSCoW method](#). Also a limit of 5 cards was set for each lane.

We switched to weekly releases with approximately 3 story cards added in each of them and started using the continuous flow.



Weekly Development Meeting

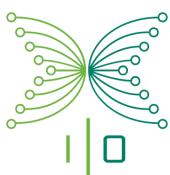
Weekly developments meetings are held every Wednesday on 8PM Osaka time, and they are one of the most important parts in our development process, since they help us keep track of the following items:

- Sprint progression.
- What has been done.
- What needs to be finished.
- Any notices that needs to be notified to the team.
- Discuss topics that needs to be discussed with the team.

Every development team member was participating in the meeting. Our team was using Google Hangouts for the meeting and we recorded it by voice recorder. Recordings were posted in Slack's **dev-meeting-rec**. Basic agenda of the meeting was provided by the project manager.

Example:

- State of development
 - Current state of development. What team has finished, what needs to be done.
- Must Haves
 - Clear out what need to be done as soon as possible.
- Deadlines
 - Deadline of our next release.
- Git
 - Git history needs improvement.
- Misc
 - We have discussed about certain tasks and why they are necessary.

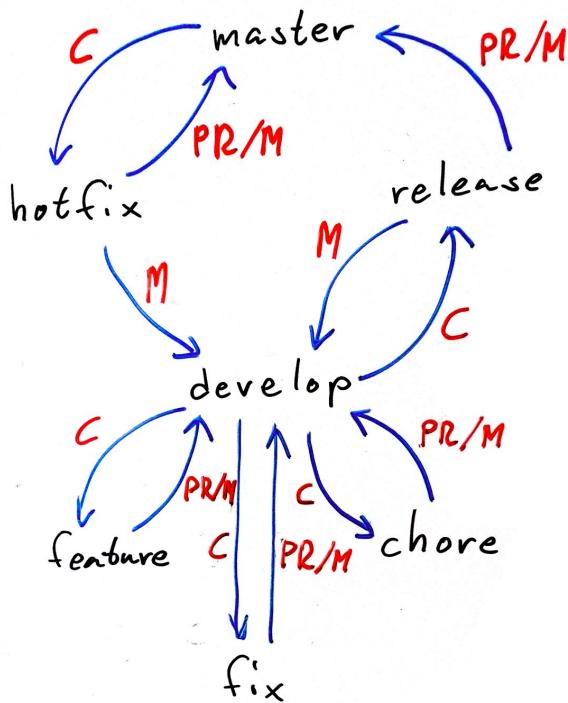


Git

Gitflow

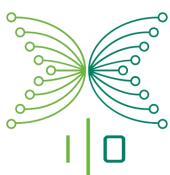
Git is not an improved CVS or SVN, Git is made for and thriving in branching and merging. Resolving merge conflicts is part of the workflow and git offers a multitude of merging strategies and conflict resolving tools for helping out.

So for a developer having a background in CVS or SVN it might be frightened when figuring that we apply something called GitFlow, which probably will feel like branching on steroids. Drop what you've learnt from CVS, SVN or similar tools and take a while to study up on how Git works, and this will be an easy ride.



References:

- GitFlow: <http://nvie.com/posts/a-successful-git-branching-model/>
- Git: <https://git-scm.com/book/en/v2>



Branches

- **master:** This is the branch containing the product, and every merge into this branch is a new version of the product.
- **develop:** The main development branch, no code should get directly committed onto this branch.
- **feature/...:** A feature branch is created for every new feature that has to be developed.
- **fix/...:** A fix branch is created for every non-critical bug that has been detected.
- **chore/...:** A chore branch is created if plain refactoring is needed.
- **release/x.x.x:** The release branch is the link between develop and master.
- **hotfix/...:** This branch type is only made if there is critical bugs that needs to be pushed directly to the product.

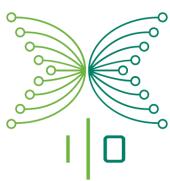
Commenting

Git Comments

Git comments are based upon Tgaberry's article "*A Note About Git Commit Messages*". If something is unclear or ambiguous, then fall back on that article as a second source of information.

The commit message guideline in bullet form:

- **The first line/headline**
 - Start with Trello card/ Github issue ID followed by the headline, separated by a dash.
 - **Max 50** character (including id, dash and space).
 - Written in imperative form(Fix instead of Fixed, Fixes, etc).
- **The second line**
 - *Always* empty.
- **The message body**
 - **Max 72** character wide.
 - Contains three sections each separated by an empty line.
 - Description
 - Title; Nature.
 - At *least* two sentences.
 - Location
 - Title; Location.
 - At least two sentences.
 - Solution
 - Title; Solution.
 - At *least* two sentences.



Example:

```
1234 - Fix bug in counter function
```

Description

The counter `function increaseByOne` increased the given value `by two`, instead of the expected one. This caused the database to only contain even numbered indexes.

Location

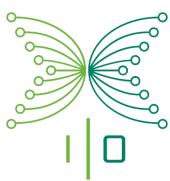
The bug was located `in increaseByOne`, but the effect became apparent `in prepareNextRecord`. Both functions are located `in database_library.js`.

Solution

Increased the argument, called `value`, only `in the end of increaseByOne`. Previously it was increased both at the beginning `and the end of the function`.

References:

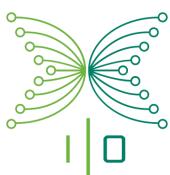
- A note about Git commit messages:
<http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html>



Pull Requests

The Pull Request message guideline in bullet form:

- **The title**
 - Same as Trello card/ Github Issue.
- The message body has up to 5 sections, each separated by an empty line
 - **Description**
 - At least two sentences.
 - **Overview**
 - **Story**
 - Start with “Story”.
 - Link text is the name of the Github story issue.
 - Link URL is the URL to the Github story issue.
 - **Affected Pull Requests**
 - Starts with “Affected Pull Requests”.
 - Comma separated list of all known other Pull Requests that in any way risk colliding with this one.
 - Link text is the name of the Pull Request.
 - Link URL is the URL to the Pull Request.
 - **Affected Critical Components**
 - Title “Affected Critical Components” in **bold**.
 - List with one bullet for each affected Pull Request.
 - Starts with the component name followed by colon.
 - Description
 - What changes has been made to the component.
 - At least two sentences.
 - **Affected Pull Requests**
 - Title “Affected Pull Requests” in **bold**.
 - List with one bullet for each affected Pull Request.
 - Start with a linked title followed by colon
 - Link text is the name of the Pull Request.
 - Link URL is the URL to the Pull Request.
 - Description
 - How the Pull Requests affecting each other.
 - At least two sentences.
 - **Pictures** (If the UI has been changed)



Example:

The document indexes in the database are all evenly numbered.

(Github PR title)

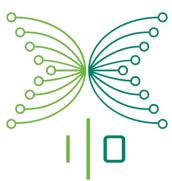
Resolves the issue regarding all documents having even numbered indexes **in** the database. The counter **function is** now increasing the index **by** one instead of **by two**.

- ****Github Issue:**** [The document indexes **in** the database are all evenly numbered](<https://github.com/input-output-hk/sales-frontend/issues/1234>)
- ****Critical Components:**** None
- ****Affected Pull Requests:**** [Add edit counter to all documents **in** the database](<https://github.com/input-output-hk/sales-frontend/pull/1357>)

Release Notes

The Pull Request message guideline in bullet form:

- **Title**
 - Same as the release name.
- **Feature section** (if any features has been implemented)
 - Title; Features.
 - List with one bullet for each feature Pull Request **as a Link**.
 - Link text is the description of the Pull Request.
 - Link URL is the URL to the Pull Request.
- Bug fix section (if any bug fixes has been implemented)
 - Title; Bug fixes.
 - List with one bullet for each bug fix Pull Request **as a link**.
 - Link text is the description of the Pull Request.
 - Link URL is the URL to the Pull Request.

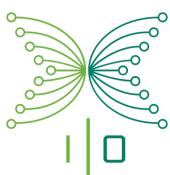


Example:

```
## Release - X

Features
* [A super cool feature to always finish the sprints on time. The solution let the user chose when the feature was finished if a flux capacitor is connected to the application.](https://github.com/input-output-hk/sales-frontend/pull/2015)

Bug fixes
* [Resolves the issue regarding all documents having even numbered indexes in the database. The counter function is now increasing the index by one instead of by two.](https://github.com/input-output-hk/sales-frontend/pull/1357)
```



Testing

Testing allows you to ensure your application works the way you think it does, especially as your codebase changes over time. If you have good tests, you can refactor and rewrite code with confidence. Tests are also the most concrete form of documentation of expected behavior, since other developers can figure out how to use your code by reading the tests.

Automated testing is critical because it allows you to run a far greater set of tests than you could do manually, allowing you to catch regression errors immediately.

References:

- Meteor testing: <https://guide.meteor.com/testing.html>

Setting Up Your System

You have to install these two npm packages globally to be able to run our tests:

- npm i -g chimp
- npm i -g spacejam

All tests of the sales application live in the `attainenroll/test` directory are split up according to their type.

Entire books have been written on the subject of testing, so we will simply touch on some basics of testing here. The most important thing to consider when writing a test is what part of the application you are trying to test, and how you are verifying the behavior works.

Unit Test

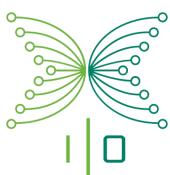
If you are testing one small module of your application, you are writing a unit test. You'll need to stub and mock other modules that your module usually leverages in order to isolate each test. You'll typically also need to spy on actions that the module takes to verify that they occur. Unit tests are run by Mocha.js

Unit test mode:

- Doesn't eagerly load any of our application code as Meteor normally would.
- Does eagerly load any file in our application (including in imports/folders) that look like `.test[s]` or `.spec[s]`

References:

- Mocha.js: <http://mochajs.org/>



Integration

If you are testing that multiple modules behave properly in concert, you are writing an integration test. Such tests are much more complex and may require running code both on the client and on the server to verify that communication across that divide is working as expected.

Typically an integration test will still isolate a part of the entire application and directly verify result in code. Meteor offers a “full application”, or integration test mode.

Integration test mode:

- It loads test file matching `.app-test[s]` and `.app-spec[s]`...
- It does eagerly load our application code as Meteor normally would.
- Sets the `Meteor.isAppTest` flag to be true (instead of the `Meteor.isTest` flag).

This means that the entirety of your application (including for instance the web server and client side router) is loaded and will run as normal. This enables you to write much more complex integration tests and also additional files for acceptance tests.

End-to-End

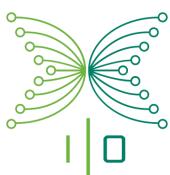
If you want to write a test that can be run against any running version of your application and verifies at browser level that the right things happen when you push the right buttons, then you are writing an acceptance test.

Such tests typically try to hook into application as little as possible, beyond perhaps setting up the right data to run a test against.

We use [Chimp.js](#) to write our acceptance tests in [Cucumber](#) scenarios with great Meteor integration. Chimp.js integrates everything you need and is based on [Webdriver.io](#) for browser automation.

Running the Tests

To run the tests, you have to run the `./bin/run/dev` or use a command to start up the Mongo database `mongod --port 3100` before you run the acceptance tests. This is because the test application used by Chimp.js actually hooks up a new `chimp_db` with the local Meteor server from the dev application. You can inspect that in [Robomongo](#) with the same connection like for the dev application.



Running

First you have to tag the features and/or scenarios you want to run with a @watch tag like this:

```
@watch # If you put it here, all scenarios in the feature will be run
Feature: Enroll

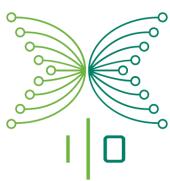
@watch # if you put it here then only this scenario is run (you can tag multiple
scenarios like this)
  Scenario Outline: A new user enrolls correctly
    Given There is a <ROLE> enroll link created
      And I go to enroll screen
    When I finish completing residence selection
      And I finish completing contact information for <ROLE>
      And I finish selecting agreements for <ROLE>
      And I finish selecting document type
    And I skip upload documents
    Then I should see a thank you message for enrolling

  Examples:
  | ROLE          |
  | buyer         |
  | distributor   |
```

Then execute ./bin/test/end-to-end-watch and it will only pick up the ones tagged with @watch.

References:

- Robomongo: <https://robomongo.org/>



Running Tests Locally

To manually execute the tests on your local machine, you should execute any of the following commands from the terminal:

- bin/test/all-run Runs end-to-end, integration and unit tests at once.
- bin/test/end-to-end-run - Runs every Chimp test,
- bin/test/end-to-end-watch - Runs every Chimp test tagged with @dev or @watch.
- bin/test/integration-run - Runs every Mocha test inside '/test/integration' folder.
- bin/test/integration-watch - Runs every Mocha test inside '/test/integration' on the browser
- bin/test/unit-run - Runs every Mocha test inside inside '/test/unit folder
- bin/test/unit-watch - Runs every Mocha test inside inside '/test/unit folder on the browser
- bin/test/packages-run - Tests calculateOrderFulfillmentAmount method with latests packages.

Running Tests on CircleCI

CircleCI is a Continuous Integration platform, which will automatically execute 'unit-run', 'integration-run' and 'end-to-end-run' scripts every time there's a change (i.e. a commit) in the remote repository (located in GitHub).

However, if you intend to re-run these tests, you are able to select one execution from the list and click 'Rebuild'.

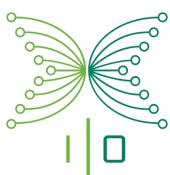
Sometimes, there are some tests that fail due to timeout issues. In that case, it's strongly recommended to rebuild the test execution choosing the option 'Rebuild without cache'.

It is also possible to integrate CircleCI with Github, in order to block PRs from merging if tests are failing in Circle. (This should only be set if we're 100% sure that tests are not failing randomly due to timeout issues or false positives).

Regarding to configuration, CircleCI uses node version 4.5.0, and updates to the latest chrome driver version. The entire configuration for CircleCI can be found in the file 'circle.yml'.

References:

- CircleCI: <https://circleci.com/>



API

As stated before, the Sales application interacts with other external systems. For this purpose there is an API (Application Programming Interface) that allows them to communicate by simple REST style HTTP requests.

There are two basic API services; to interact with the backend, that handles all bitcoin network events and the JSON Generator, which exposes necessary Vending Machine methods.

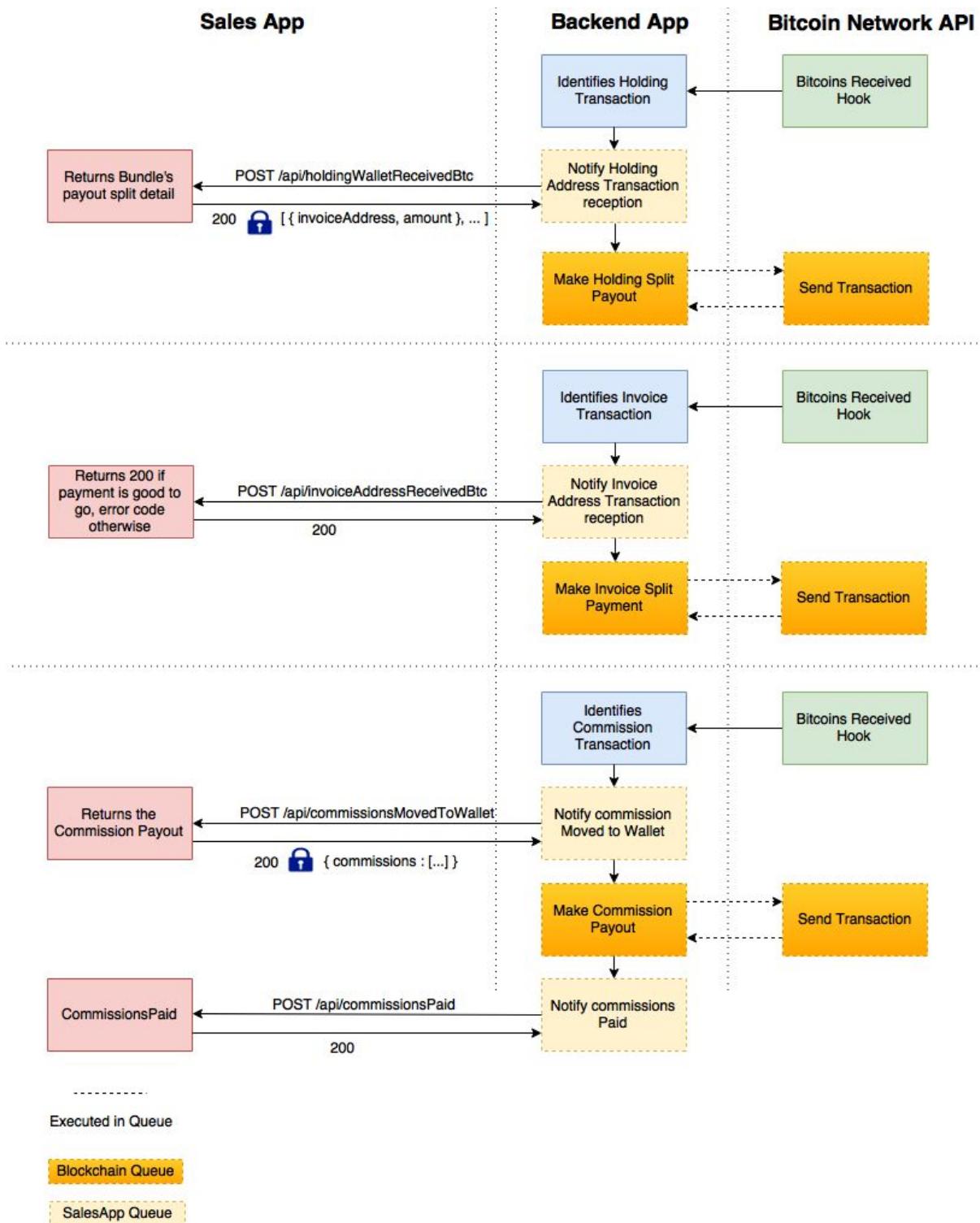
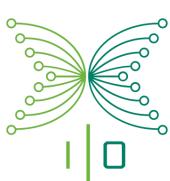
Backend

Backend application is the Sales application interface with the Bitcoin network. It's a wrapper that simplifies all payments and notifications made over that network in a way that abstract the sales process of the actual payments and transactions. It also serves as a security layer as it gives a degree of independence to the business logic from the Wallets and private keys associated to the process.

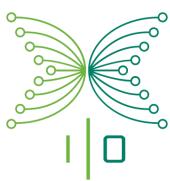
Backend application is documented in detail in other documents. This section focuses on the API services that the Sales application uses to receive Backend related events. Notice that even though these endpoints were designed for Backend application interaction, they are actually public and demand no authentication so that they could be manually triggered easily if necessary.

There is a prearranged protocol for endpoints' HTTP Status codes response, on top of the basis standard (200 OK, 4XX/5XX error) we distinguish between two main sets to indicate whether the Backend needs to retry again later with this notification or if it should stop. Any error code bigger than 555, will bring an end to this notification, otherwise the Backend will retry if there is an error. Besides that, any expected business error will have a descriptive text message as well.

Every sensitive data return by these API Endpoints, will be encrypted with a predefined key arranged with the Backend; this guarantees both authorization and data integrity.



Backend notifies Sales application on Bitcoin Network events.



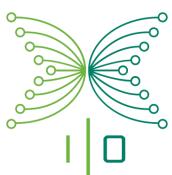
Endpoints Details:

- Holding Wallets Received Bitcoins Notification

Holding Wallet is where Bank Bundled payments are received. One single address in that Wallet is linked to a particular Bundle and funds are also sent by one single transaction. This endpoint is where the Sales application gets notified that one of those transactions has been confirmed. Sales application will search for the corresponding bundle by the received address, verify the amount and return the list of individual orders (invoice address) that belong to that bundle with each corresponding amount. This list is returned as a json encrypted response payload.

```
POST /api/holdingWalletReceivedBtc body
{
    "holdingWalletAddress": "address",
    "satoshisReceived": "#ofsatoshisreceived",
    "transactionId": "transactionId"
}
→ 200 signed(*) body
  { "payout" : [ { invoiceAddress": "invoiceAddress", "amount": "amount"}, ... ] }
→ < 555 => (any error < 555 triggers retry)
→ > 555 => Stop Call
```

(*) Note: The payload body must be signed with the backend given key.

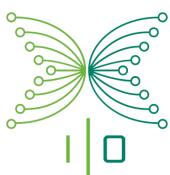


- Invoice Address Received Bitcoin

Each Invoice Ticket is associated with an individual address, this address belongs to HD Multisig Wallets linked as well as with the responsible distributor of this sale. Whenever a payment is received and confirmed, Sales application will receive a notification on this endpoint.

The notification will specify the amount of confirmations received, and the sales application will respond to basically 2 states, zero confirmation (aka lightning signal) and any confirmation. First one will just record the event with the transaction Id and the amount, but when confirmed, the payment is verified and emails are sent. On a successful response to this last event, Backend will split the funds to the exodus address and the Commission address.

```
POST /api/invoiceAddressReceivedBtc body
{
    "invoiceAddress": [String],
    "satoshisReceived": [Integer],
    "transactionId": [String],
    "date" : [DateTime],
    "confirmations" : [Integer],
} → { 200 | 4xx | 5xx}
```



- Commission are ready to be paid

Once an Invoice has been paid and the transaction is confirmed and approved by the invoiceAddressReceivedBtc endpoint, commissions must be paid as well to each corresponding distributor. This endpoint notifies the Sales application that the Backend is ready to make those payments, and expects the detailed information of which address and quantities should be done. For this, the sales application will respond - if successful - with an encrypted payload containing the distributor's personal address and the commission share amount (in satoshis) corresponding to each one. The sum of all, should add up to the commission address totals.

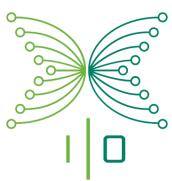
```
POST /api/commissionsMovedToWallet body
  {"invoiceAddress": "invoice-address",
   "commissionWalletAddress": "commission-address",
   "satoshisAmount": "satoshis-amount",
   "transactionId": "transactionId",
   "date": "tx-date"}
→ 200 signed(*) body
  {"commissions": [{"commissionAddress": "address", "amount": "#ofsatoshis"}]}
→ < 555 => (any error < 555 triggers retry)
→ > 555 => Stop Call
```

(*) Note: The payload body must be signed with the backend given key.

- Commissions Paid

Once the commission payout transaction is confirmed, this endpoint should be called to finish the payment process.

```
POST /api/commissionsPaid body
  {"invoiceAddress": "invoice-address", "transactionId": "transactionId"}
→ 200 => Success/Confirmation
→ < 555 => (any error < 555 triggers retry)
→ > 555 => Stop Call
```



JSON Generator

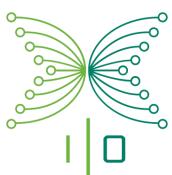
The JSON Generator is the intermediate data export step between the Sales application and the AVVM. The actual data is read directly from the same database as the Sales application but updates have to be done through API calls.

Set Invoice Ticket PII Hash

Saves a PII hash to the database together with an invitation email. The request should be sent as a `POST` package to `/api/setInvoicePiiHash`.

Headers

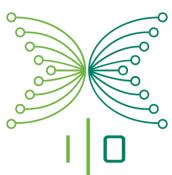
Type	Description
authorization	API authentication token. Example value: <code>Bu4vH1vW-i1PVmmXg-peRFLqVH-hZU5xZtO</code>



Body

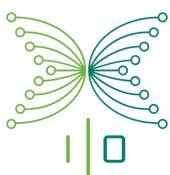
The body should be formatted as a JSON string with the following fields.

Field name	Description
ticketId	<p>The id of the ticket for which the PII hash should be updated. The id format depends on the id incrementer used when inserting documents in the database.</p> <p>Example id: 2EBPJaDa8mGfGcxfr</p>
piiHash	<p>The PII hash to be set. The hash is a SHA256 hash of a concatenation of specific database values.</p> <p>Example hash: 5uOHZGlIH4XBACULMXsdz120N89/9sYmlqAgjrSMNFM=</p>
email	<p>The invitation email used by the JSON Generator. The email format has to comply with the W3C recommendations.</p> <p>Example email: email@domain.com</p>
comment (optional)	<p>An optional comment to store with the PII hash.</p> <p>Example comment: A comment about the PII hash generation.</p>



Response Codes

Status code	Content	Description
200		Successfully updated the piiHash.
400	Malformed request body	The ticketId, piiHash, email keys or the body itself is missing.
400	Invalid piiHash format	The piiHash key was found but the value wasn't a string.
400	Invalid email format	The email key was found but the value didn't match the recommended email format by W3C.
400	Invalid comment format	The comment key was found but the value wasn't a string.
400	Invoice Ticket not found for ticketId: <ticketId>	The ticketId key was found but the value didn't match any existing ticket id.
400	Invoice Ticket <ticketId> is not ready to set piiHash yet	The ticketId key was found but there is no Ada passcode for the ticket.
400	Invalid body content, couldn't save the data	The content couldn't be saved to the database.
401	Token Authorization error	No authentication token was provided or an incorrect token was used.



Example: Request and response packages

```
POST /api/setInvoiceTicketPiiHash HTTP/1.1
host: test.domain.com
authorization: Bu4vH1vW-i1PVmmXg-peRFLqVH-hZU5xZt0

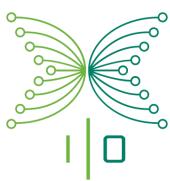
{
  "ticketId": "2EBPJaDa8mGfGcxfr",
  "piiHash": "5uOHZG1IH4XBACULMXsdz120N89/9sYmlqAgjrSMNFM=",
  "email": "email.domain.com",
  "comment": "This email is probably faulty."
}
```

Package with faulty email address (no @ sign)

```
HTTP/1.1 400 Bad Request
content-length: 20
content-type: text/plain; charset=utf-8
date: Thu, 22 Dec 2016 05:57:22 GMT
status: 400
```

Invalid email format

Response with notification about the faulty email format



Advanced Features

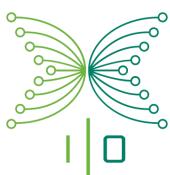
Jobs and Workers

Many state transitions and background processes are run by Jobs and Workers. These workers are integrated into the Sales application using *vsivsi:job-collection* library. This library is a powerful and easy to use job manager for Meteor applications, it allows:

- Scheduled jobs to run (and repeat) in the future, persisting across server restarts.
- Create repeating jobs with complex schedules using [[Later.js](#)].
- Move work out of Meteor's single threaded event-loop.
- Enable work on computationally expensive jobs to run anywhere, on any number of machines.
- Track jobs and their progress, and automatically retry failed jobs.
- Easily build an admin UI to manage all of the above using Meteor's reactivity and UI goodness.

In Sales-app we use endless workers that run along the application's life, each worker will be identify by it's `jobName` (works as key). On startup, `init.js` will start each of the registered jobs.

```
const job = new Job(Jobs, jobName, {})
  .retry({ retries: Jobs.forever, wait: callInterval })
  .repeat({ repeats: Jobs.forever, wait: callInterval })
  .save();
```



Every worker will need to

- Declare an unique name
- Mongodb cursor to pick the data it will act on
- Task (function) it wants to execute over each item returned by that cursor.

For example, the worker that sends funds received confirmation looks like:

```
Jobs.registerWorker({  
  jobName: 'sendConfirmFundsReceivedEmailWorker',  
  cursor: function getFundsReceivedConfirmedTickets() {  
    return InvoiceTickets.find({ state: 'fundsReceivedConfirmed' });  
  },  
  task(ticket) {  
    const options = { ticketId: ticket._id };  
    sendET205ConfirmFundsReceived(options);  
    updateInvoiceTicket('sendConfirmFundsReceived', ticket);  
  }  
});
```

Jobs are recorded to the database on the “jobs.jobs” collection, but a simplified version can be access by the sysop role in the application, the dashboard displays current job status:

The screenshot shows a dashboard with a sidebar on the left and a main content area on the right.

Sidebar (Left):

- Sysop Sysop
- test-email+sysop@lohr.io
- SALE OVERVIEW
- JOB MONITOR (highlighted)
- LOGOUT

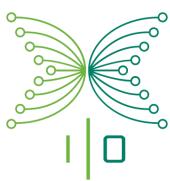
Main Content Area (Right):

Jobs

Type	Status	Created	Updated
approveComplianceBankWorker	waiting	12月 21日 2016 14:44	12月 21日 2016 14:44
approveComplianceBankWorker	completed	12月 21日 2016 14:41	12月 21日 2016 14:44
approveComplianceBtcWorker	waiting	12月 21日 2016 14:44	12月 21日 2016 14:44
approveComplianceBtcWorker	completed	12月 21日 2016 14:41	12月 21日 2016 14:44

References:

- Later.js: <https://bunkat.github.io/later/>
- Vsivsi:job-collection: <https://atmospherejs.com/vsivsi/job-collection>



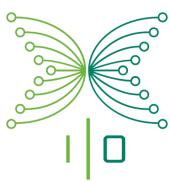
Ada Reservation

ADA reservation process is composed by a set of controlled sales, it can be generally turned on or off by the flag "salesStarted" and allowed in particular for each region by "residenceCountriesAllowedSale".

Each period (tranche) has a predefined total of USD equivalent of ADA ("totalAmountAvaiable") available to be exchanged, this amount can be exceeded by a certain percentage ("overCapacityTolerance").

All of this is configured in the application settings file and needs to be set before each deploy:

```
// file path: "/config/settings.json"
{
  "public": {
    "salesStarted": true,
    "residenceCountriesAllowedSale": ["KR", "JP"],
    "salesLimits": {
      "currentTranche": "t3",
      "tranches": {
        "t1": {},
        "t2": {},
        "t3": {
          "totalAmountAvailable": 14000000,
          "overCapacityTolerance": 0.2
        },
        "t4": {}
      }
    },
    ...
  }
}
```



Sales Predictor

The module in charge of checking if the sales goal has been reached and if a ticket can be sold, is called "SalesPredictor".

To do so, it's instanced with the following parameters:

```
// file path: '/imports/lib/sales-predictor.js'

export default class SalesPredictor {
  /*
  numbers:
  totalAmountAvailable:    Total amount available in this tranche
  totalAmountAvailableCap: Total amount available + over cap
  overCapacityTolerance:   Overcapacity tolerance

  functions:
  amountInvoicedOrPaid:   Amount of invoiced or received payment in this tranche
  amountPaid:              Amount of received payment in this tranche
  */
  constructor(options) {
    this.totalAmountAvailable = parseInt(options.totalAmountAvailable, 10) || 0;

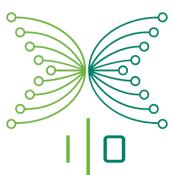
    this.overCapacityTolerance = parseFloat(options.overCapacityTolerance) || 0;

    this.amountInvoicedOrPaid =
      options.amountInvoicedOrPaid || function() { return 0; };

    this.amountPaid = options.amountPaid || function() { return 0; };

    this.overCapacityToleranceValue =
      this.totalAmountAvailable * this.overCapacityTolerance;

    this.totalAmountAvailableCap =
      this.totalAmountAvailable + this.overCapacityToleranceValue;
  }
}
```



```
...  
}
```

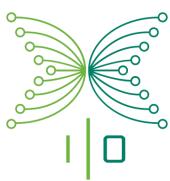
"amountPaid" is calculated as the sum of all ticket's amounts in paid states (*) for current tranche. And "amountInvoicedOrPaid", will add all tickets in invoiced states (**) on top of that.

(*) Paid states:

- satoshisReceived
- adaPasscodeAssigned
- receiptSent

(**) Invoiced states:

- saleStartedBtc
- saleStartedBank
- complianceApprovedBtc
- complianceApprovedBank
- fundsReceivedConfirmed
- fundsReceivedConfirmedSent
- ticketPrepared
- ticketBundled
- ticketFinalized
- invoiceSentBank
- fundsReceived
- invoiceSentBtc
- invalidTicketApproved
- invalidFundsReceived.



A new ticket it can be sold only if the salesPredictor allows it. The validation method is called “allowSale()” and follows the next flow chart:

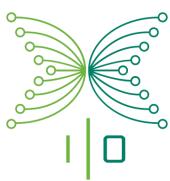


Invoice Ticket State Machine

In order to control the invoice tickets flow, we use a meteor plugin that exposes a *state-machine* to manage all the different states and the transitions between them.

Depending on the case, the transitions on the machine could be triggered by a worker or an user action.

These transitions are defined according to the business logic, and the invoice state machine will only allow a ticket to change its state if the state exists and the transition is valid.



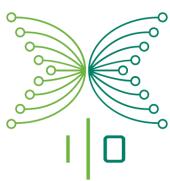
Example: of usage in a worker (from server side)

```
'use strict';

import { Jobs } from '/lib/collections/jobs.js';
import { sendInvoiceEmail } from '../../imports/server/email-actions.js';
import { updateInvoiceTicket } from '/imports/server/invoice-ticket-state-machine';

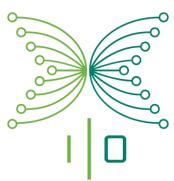
Jobs.registerWorker({
  jobName: 'sendInvoiceWorker',
  cursor: function getTickets() {
    return InvoiceTickets.find({ state: {$in:
      ['complianceApprovedBank', 'complianceApprovedBtc']
    }});
  },
  task: function sendInvoice(ticket) {
    const options = { ticketId: ticket._id };
    const changeToState = 'sendInvoice' + ticket.paymentOption;
    sendInvoiceEmail(options);
    updateInvoiceTicket(changeToState, ticket);
  },
});
```

The stateMachine is defined and created in file “invoice-ticket-state-machine”. In that file you will find the exposed method “updateInvoiceTicket”, that manages the state transitions, validations and data updates for the ticket.

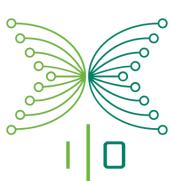


Example: of a state transition triggered by an user action (from client side)

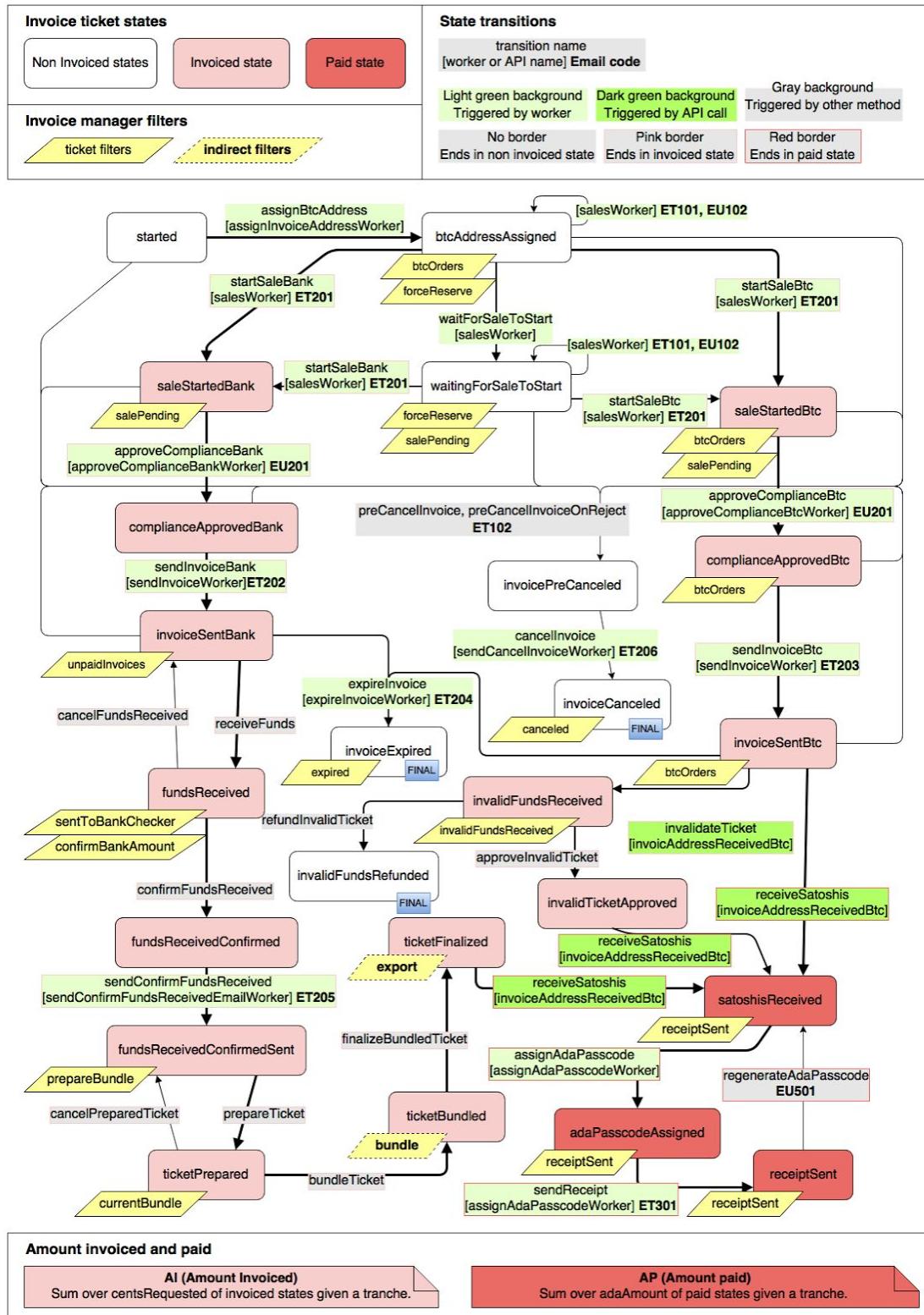
```
// in file: '/client/views/user-overview/partials/invoice-ticket-action-cancel.js'  
// user action:  
  
events: {  
  'click button.cancel-ticket'() {  
    confirmModal(  
      i18n('modals.cancelTicket'),  
      ()=> {  
        this.data.invoice.changeState(  
          'preCancelInvoice',  
          undefined,  
          (err) => {i18nAlert(err, I18N_ERROR_NAMESPACE);}  
        );  
      }  
    );  
  }  
}  
...  
  
// in file: '/lib/collections/invoice-tickets.js'  
// invoice method:  
  
changeState: function(event, modifier, handlerError, handlerSuccess) {  
  Meteor.call('changeState', event, this, modifier, function(err) {  
    if (err && _.isFunction(handlerError)) {  
      handlerError(err);  
    } else if (_.isFunction(handlerSuccess)) {  
      handlerSuccess();  
    }  
  });  
},
```

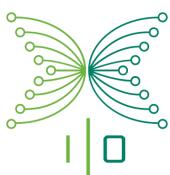


The correct way to use the state-machine from client side is using the method "changeState" of the invoice. This method internally uses the meteor call "changeState" that checks user role and creates a record of the action (in the collection stateTransitionRecords) before it uses the "updateInvoiceTicket" state-machine method.



Full diagram of the Invoice state machine:





References:

- State Machine package: <https://github.com/jakesgordon/javascript-state-machine#usage>

Refcodes

Refcodes are authentication codes that allow external users to interact with their data. This means either adding new data or changing current data. A refcode can be bound to an email-address or userId and has no predefined structure.

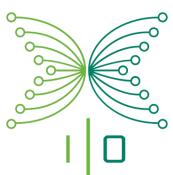
Refcodes are generated by using a md5 hash from a random number. Refcodes expire 24 hours after they are created except for 4 special types.

Router Validation

The refcodes are validated in the router, and then they are marked as isActive = false in the database, so they cannot be used again (Unless the refcode timetype is 'unlimited' which is a possible case for enroll refcodes). Then the application will redirect and call a certain template depending on the ref type, which is defined in the router.

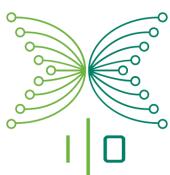
The application has the following refcode types:

Type	Description	Expires
emailVerification	This verifies a user's email	Y
confirmAddress	This confirms the wallet address update	Y
confirmEmailAccount	This confirms an email-account	Y
confirmEmailChange	This confirms the change in email-address	Y
reset	This allows a user to reset the password	Y
re-generate-all	This allows the user to regenerate all of the ADA codes	Y
re-generate-one	This allows the user to regenerate one of the ADA codes	Y
signup	This allows a user to create a password on the first login as a distributor	N
buyer	Refcode for a buyer to enroll	N
distributor	Refcode for a distributor to enroll	N
partner	Refcode for a partner to enroll	N

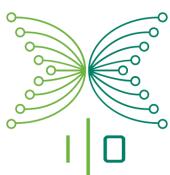


Refcode documents can contain the following keys:

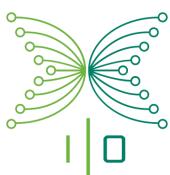
Type	Description
_id	Document id
clicked	Shows how many times a refcode has been used
reftype	The type of refcode
refcode	The code that is used for verification of the ref
mailto	The email linked to the refcode
timetype	This specifies how many times a refcode can be used
originatorId	This is the userId of the owner of the refcode, i.e. the distributor that supplied it
createdAt	The date the refcode was created
isActive	Whether the refcode is active
name	Name of the person bound to the refcode
notes	Any notes left on the refcode
userId	The id of the user bound to the code
emailAddress	The email-address that corresponds to the refcode
invoiceTicketId	The invoice ticket bound to the refcode

**Example:** Refcode documents

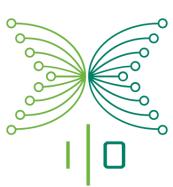
Type	Description
emailVerification	{ "_id" : "LMtzpcYwSYggkvbDY", "clicked" : 0, "reftype" : "emailVerification", "refcode" : "5db3aa940d0b0a386f16", "emailto" : "test-email+xxx@iohk.io", "timetype" : "onetime", "originatorId" : null, "createdAt" : ISODate("2016-12-20T05:56:52.559Z"), "isActive" : true }
confirmAddress	{ "_id" : "B5dpQPa78anW67qJQ", "clicked" : 0, "reftype" : "confirmAddress", "refcode" : "r39cd78dccf7bb7c9a12d", "timetype" : null, "name" : null, "notes" : null, "userId" : "22ThpRGZtdKxRBKFp", "address" : "1FRqs8e8sCY8AQTdCoALJLJDNNqdDxCYVk", "originatorId" : null, "createdAt" : ISODate("2016-12-20T06:41:59.912Z"), "isActive" : true }
confirmEmailAccount	{ "_id" : "PLp4hzarKaJaRF8W3", "clicked" : 0, "reftype" : "confirmEmailAccount", "refcode" : "r4656d0ca8e1266cf0abb", "timetype" : null, "name" : null, "notes" : null, "userId" : "22CZvqFHAqHQhRWuf", "emailAddress" : "test-email+xxx@iohk.io", "originatorId" : null, "createdAt" : ISODate("2016-12-20T06:50:19.718Z"), "isActive" : true }
confirmEmailChange	{ "_id" : "Nw5kpFo5apyvB4gmp", "clicked" : 0, "reftype" : "confirmEmailChange", }



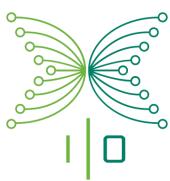
	<pre> "refcode" : "r80707e208adb504734c4", "timetype" : null, "name" : null, "notes" : null, "userId" : "22CZvqFHAghQhRWuf", "emailAddress" : "test-email+xxx@iohk.io", "originatorId" : null, "createdAt" : ISODate("2016-12-20T06:51:34.370Z"), "isActive" : true }</pre>
reset	<pre>{ "_id" : "22ziMACE7jtHKWqxG", "reftype" : "reset", "refcode" : "s0eb666896e9eb9j3r3m7as026292dba7", "mailto" : "test-email+xxx@iohk.io", "originatorId" : null, "createdAt" : ISODate("2016-07-16T06:28:39.005Z"), "isActive" : false, "clicked" : 0 }</pre>
re-generate-all	<pre>{ "_id" : "ghuapwfAbRPB42JZj", "clicked" : 0, "reftype" : "re-generate-all", "refcode" : "ra6927f4f850602a9f23e", "timetype" : null, "name" : null, "notes" : null, "userId" : "NNTPyahg4AqG7hBZG", "mailto" : null, "originatorId" : null, "createdAt" : ISODate("2016-12-20T07:16:08.806Z"), "isActive" : true }</pre>
re-generate-one	<pre>{ "_id" : "KCPTjrwAhvFovSdZD", "clicked" : 0, "reftype" : "re-generate-one", "refcode" : "rd7b8a4c7cb7f8ecf3c25", "timetype" : null, "name" : null, "notes" : null, "userId" : "NNTPyahg4AqG7hBZG", "mailto" : null, "invoiceTicketId" : "2aAyvaMc2FcFPG6Xg", "originatorId" : null, "createdAt" : ISODate("2016-12-20T07:15:20.352Z"), "isActive" : true }</pre>



	<pre>}</pre>
signup	<pre>{ "_id" : "229ZREjWBw6qPJDBK", "reftype" : "signup", "refcode" : "s0eb666896e9eb9j3r3m7as026292dba7", "emailto" : "test-email+xxx@iohk.io", "originatorId" : null, "createdAt" : ISODate("2016-08-24T08:03:31.899Z"), "isActive" : false, "clicked" : 0 }</pre>
buyer	<pre>{ "_id" : "22CEgSYgDirKshtfN", "reftype" : "buyer", "name" : "Trum", "timetype" : "onetime", "isActive" : false, "originatorId" : "EzpBZ6RDDTxraZHNG", "refcode" : "s0eb666896e9eb9j3r3m7as026292dba7", "clicked" : 0, "notes" : "Dope", "createdAt" : ISODate("2016-04-21T04:24:38.829Z") }</pre>
distributor	<pre>{ "_id" : "22DuPnsxybzSDvFh8", "createdAt" : ISODate("2015-09-07T09:32:09.235Z"), "refcode" : "s0eb666896e9eb9j3r3m7as026292dba7", "clicked" : 0, "originatorId" : "dmkd8fGiMjEKwXwYQ", "notes" : "Dope", "name" : "Trum", "timetype" : "unlimited", "distlevel" : 2, "reftype" : "distributor", "isActive" : false }</pre>
partner	<pre>{ "_id" : "2BhmGfz6xwbZkbudT", "distlevel" : 0, "createdAt" : ISODate("2015-09-29T02:30:13.512Z"), "notes" : "Dope", "timetype" : "onetime", "name" : "Trum", "isActive" : false, "refcode" : "s0eb666896e9eb9j3r3m7as026292dba7", }</pre>



	<pre>"clicked" : 0, "reftype" : "partner", "originatorId" : "xDMihrT8EzqfxY6uv" }</pre>
--	---



Languages, Standards and Frameworks

Standards

JSON

JSON (JavaScript Object Notation) is an open-standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs.

In the application all the records are stored into JSON documents (see section [Databases - MongoDB](#)) and also is used as a data format for the communication between the Sales application and the Backend.

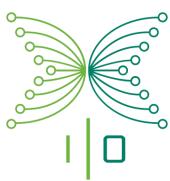
Example: JSON in the application

```
// ref document from the refs collection.  
{  
  "_id" : "22N5msmBzyvByaoti",  
  "clicked" : 0,  
  "reftype" : "signup",  
  "refcode" : "s1f588932417f1384703251cd65059e39",  
  "mailto" : "sample-user@iohk.io",  
  "originatorId" : null,  
  "createdAt" : ISODate("2016-02-20T08:19:24.562Z"),  
  "isActive" : true  
}
```

HTTP

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

Sales application uses HTTP to serve all its frontend content, although it follows modern web browsers standards, we recommend Google Chrome (Version 54+).



i18n

i18n (Internationalization) is a system for allowing the application to support multiple languages. Most important part when using i18n system, is to replace all strings containing text with i18n objects.

We use following meteor packages to implement i18n:

- Anti:i18n
- Gwendall: simple-schema-i18n
- Gwendall: auto-form-i18n

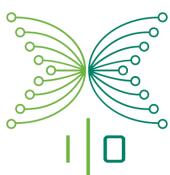
Currently all 18n files are stored at sales-frontend/attainenroll/lib. Format for the files is as follows:

i18n_viewlinks.js

```
i18n.map('en', {
  viewlinks: {
    partner: 'Partner Links',
    distributor: 'Distributor Links',
    buyer: 'Buyer Links',
  }
});

i18n.map('ja', {
  viewlinks: {
    partner: 'パートナー用紹介URL',
    distributor: '代理店用紹介URL',
    buyer: '交換希望者用紹介URL'
  }
});

i18n.map('ko', {
  viewlinks: {
    partner: '파트너용 소개 URL',
    distributor: '대리점용 소개 URL',
    buyer: '교환 희망자용 소개 URL',
  }
});
```



```
    }  
});  
  
i18n.map('zh', {  
  viewlinks: {  
    partner: '合作夥伴介绍连结',  
    distributor: '代理店介绍连结',  
    buyer: '希望兑换者介绍连结',  
  }  
});
```

The application currently supports up to four languages(depending on the part of the application), so the i18n file is split in four i18n maps. In each map there is a property containing a list of translations for that language.

Property name is usually the same as the file name. To insert the translated i18n string in html code, use the following syntax:

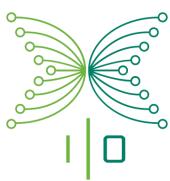
```
{{ i18n 'viewlinks.buyer' }}
```

To access the i18n string in javascript use:

```
i18n('viewlinks.buyer');
```

To switch between languages, pass in language code('en', 'ja', 'ko', 'zh') as parameter to this function:

```
i18n.setLanguage('ja');
```



Missing translations are marked with !!! and the language code for ease of searching:

```
i18n.map('en', {
  viewlinks: {
    partner: 'Partner Links',
  }
});

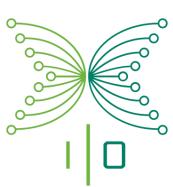
i18n.map('ja', {
  viewlinks: {
    partner: '!!!ja Partner Links',
  }
});

i18n.map('ko', {
  viewlinks: {
    partner: '!!!ko Partner Links',
  }
});

i18n.map('zh', {
  viewlinks: {
    partner: '!!!zh Partner Links',
  }
});
```

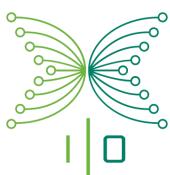
Unneeded translations are marked with !!!- and the language code:

```
i18n.map('ko', {
  viewlinks: {
    partner: '!!!-ko Partner Links',
  }
});
```



References:

- I18n package: <https://atmospherejs.com/anti/i18n>
- Anti:i18n: <https://github.com/anticoders/meteor-i18n>
- Gwendall: simple-schema-i18n: <https://github.com/gwendall/meteor-simple-schema-i18n>
- Gwendall: auto-form-i18n: <https://github.com/gwendall/meteor-autoform-i18n>



DDP

Distributed Data Protocol (or DDP) is a client-server protocol created for use by the Meteor framework.

It used to power the querying and updating a server-side database and for synchronizing such updates among clients. In order to handle it, DDP uses the publish-subscribe messaging pattern.

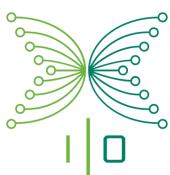
Example:

```
// On the Server: '/server/publications.js'

Meteor.publish('jobs', function() {
  if (Roles.userIsInRole(this.userId, ['sysop'])) {
    return Jobs.find({}, secureJobsOptionsByRole({}, 'sysop'));
  }
  return null;
});
```

```
// On the Client: '/client/views/jobs-monitor/jobs-monitor.js'

TemplateController('jobsMonitor', {
  onCreated() {
    Meteor.subscribe('jobs');
  },
  helpers: {
    jobs() {
      return Jobs.find({}, {sort: {type: 1, created: -1, updated: -1}});
    }
  },
  // file continues..
});
```



With this, the implementation of DDP in Meteor handles the rest; It intelligently polls the database to pick up changes on the jobs collection and pushes them down to the client. DDP can also simulate the model code on the client side, so the screen is updated immediately without waiting for the network response.

References:

- DDP:
<https://github.com/meteor/meteor/blob/master/packages/ddp/DDP.md#ddp-specification>

Languages

ECMAScript 6

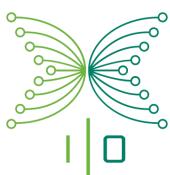
EcmaScript 6 is the latest (written at 21 December 2016) version of the JavaScript standard. The application uses many features that come with this version.

Example:

```
import { markJobAsReady } from '/imports/server/workers';
import { preCancelInvoicesOnReject } from '/server/invoice-manager';
const checkRevieweeExistance = (userId, callerMethod) => {
  const reviewee = Meteor.users.findOne(userId);
  if (reviewee === undefined || reviewee === null) {
    Logger.error('[RPC][Warning] ', callerMethod, ': User not found.', 'userId:', userId, 'customerServiceId:', this.userId);
    throw new Meteor.Error('userNotFound');
  }
  return reviewee;
};
```

References:

- ES6: <http://es6-features.org>

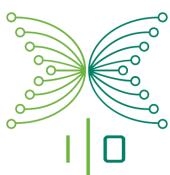


HTML

HTML is the main markup language used in the project. Every template used, uses this markup.

Example:

```
<template name="buyers">
<div id="buyers-screen" class="data-screen">
  <header>
    <div class="headline">
      <h1>{{i18n "buyers.headline"}}</h1>
      {{#if showAddBuyerButton}}
        <button class="add-buyer">{{i18n "buyers.addBuyer"}}</button>
      {{/if}}
    </div>
    <p class="buyers-count">
      {{i18n "buyers.count.youHave"}}
      {{buyersCount}}
      {{i18n "buyers.count.buyers"}}
    </p>
    <a href="#" class="toggle-all">
      {{#if areAllOpen}}
        {{i18n "buyers.closeAll"}}
      {{else}}
        {{i18n "buyers.openAll"}}
      {{/if}}
    </a>
  </header>
  <div class="buyers-list">
    {{#each buyer in currentUser.buyers}}
      <div class="row">
        {{> buyerPanel buyer=buyer isOpen=areAllOpen }}
      </div>
    
```



```
    {{/each}}
```

```
</div>
```

```
{#{if hasMoreToLoad}}
```

```
    <div class="load-more-results">{{> spinner}}</div>
```

```
  {{/if}}
```

```
</div>
```

```
</template>
```

LESS

Less is a CSS pre-processor, meaning that it extends the CSS language, adding features that allow variables, mixins, functions and many other techniques that allow you to make CSS that is more maintainable, themable and extendable.

LESS files are the main form of styling used for the templates in the code base.

Example: LESS

```
.account-info-panel {
```

```
  .panel-heading {
```

```
    padding-bottom: 0;
```

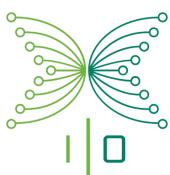
```
    background-color: white;
```

```
    h1 {
```

```
      margin-bottom: 1em;
```

```
    }
```

```
  }
```



Example: Template using the styles from LESS

```
<template name="accountInfoPanel">
<div class="account-info-panel panel panel-default">
<div class="panel-heading">
  <h1>{{ i18n 'accountInfo.account.title' }}</h1>
  <!-- Nav tabs -->
  <ul class="nav tabs" role="tablist">
    <li role="presentation" class="active">
      <a href="#information" aria-controls="information" role="tab">
        {{ i18n 'accountInfo.tabs.information' }}
      </a>
    </li>
    <li role="presentation">
      <a href="#agreements" aria-controls="agreements" role="tab">
        {{ i18n 'accountInfo.tabs.agreements' }}
      </a>
    </li>
  </ul>
</div>
```

References:

- LESS: <http://lesscss.org/#>

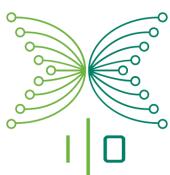
Cucumber

Cucumber is a tool for running automated tests written in plain language, allowing it to be easily read and understood by anyone on your team.

This tool understands and uses Gherkin to write test scenarios in a more ‘human’ language, and is used within Chimp.js.

References:

- Cucumber: <https://cucumber.io/>



Shell Script

A shell script is a computer program designed to be run by the Unix shell, a command-line interpreter. The various dialects of shell scripts are considered to be scripting languages. Many Sales application script are shell script commands, as they are intended to be run at SO level. For example there are shell scripts to run the application, tests, and to deploy.

Example: Sales application run shell script ./bin/run/dev

```
#!/usr/bin/env bash
echo "Sourcing ./bin/env/development.sh as environment ..."
source ./bin/env/development.sh
cd attainenroll/
echo "Running meteor with --settings ../config/develop.json ..."
meteor "$@" --settings ../config/develop.json --no-release-check
```

Perl

Perl is a family of high-level, general-purpose, interpreted, dynamic programming languages.

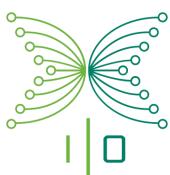
The Perl languages borrow features from other programming languages including C, shell script (sh), AWK, and sed. They provide powerful text processing facilities without the arbitrary data-length limits of many contemporary Unix command line tools, facilitating easy manipulation of text files. It is commonly used by Web programmers to create scripts for Web servers.

Perl commands are used internally in some of Sales application bash scripts. For example, in deploy script, we can found:

```
# Collaps settings JSON data one level
echo -ne "$WAIT Preparing settings and secrets..."
perl -i -pe 'BEGIN{undef $/;} s/\s*\{\{(.*)\}\}\s*/\1/sm' $TEMP_SETTINGS
```

References:

- Perl: <http://perldoc.perl.org/index.html>



PowerShell

Powershell is a console in Windows which can execute .ps1 scripts and works similarly, this is similar to Shell Scripts in unix/linux based systems.

Example: PowerShell script that starts the Meteor server

```
echo "Sourcing ./bin/env/development.ps1 as environment ..."  
./bin/env/development.ps1  
cd ./attainenroll  
echo "Running meteor with --settings ../config/development/settings.json ..."  
meteor --settings ../config/development/settings.json
```

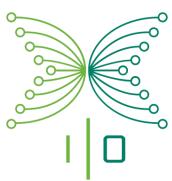
Frameworks

Meteor

Meteor is a full-stack JavaScript platform for developing modern web and mobile applications. It integrates with MongoDB and uses the Distributed Data Protocol (DDP) and a publish–subscribe pattern to automatically propagate data changes to clients without requiring the developer to write any synchronization code.

Meteor uses **data on the wire**, meaning the server sends data, not HTML, and the client renders it. It also provides **full stack reactivity**, allowing the UI to seamlessly reflect the true state of the world with minimal development effort.

- Meteor version: 1.4.2.3
- Node version: 4.6.0
- npm version: 3.10.8



Example: Publish-subscribe code

```
// Server
Meteor.publish('reviewRecords', function() {
  if (Roles.userIsInRole(this.userId, Meteor.users.getComplianceRoles())) {
    return ReviewRecords.find({});
  }
  return undefined;
});

// Client
Meteor.subscribe('reviewRecords');
```

Example: collection code

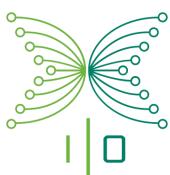
```
// To create a new collection
ReviewRecords = new Mongo.Collection('reviewRecords');

// Find, update, insert
ReviewRecords.find();
ReviewRecords.insert(reviewRecord);
ReviewRecords.update({_id: reviewRecordId}, { $set: {date: new Date()}});

```

References:

- Meteor: <https://www.meteor.com/>



Blaze

Blaze is the Meteor's default frontend rendering system.

- Its templates consists of views and controllers
 - Views are written in **Spacebars** which is handlebars-like templating language with reactively changing datacontext.
 - Controllers are written in **Javascript**.
- It uses Tracker to automatically keep track of when to recalculate each template.
Because of this, developers don't have to explicitly declare when to update the DOM, or even perform any explicit "data-binding."

We use Blaze to create sales application user interface. We also use template-controller which provides a very thin layer of syntactic sugar on top of the standard API.

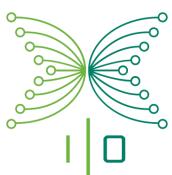
References:

- Blaze: <http://blazejs.org/index.html>
- Tracker: <https://docs.meteor.com/api/tracker.html>
- Template-Controller: <https://github.com/meteor-space/template-controller>

Example:

Investigator-tabs.html

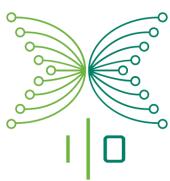
```
<template name="investigatorTabs">
  <div class="row">
    <div class="col-md-3">
      {{>} counterButton label=(i18n 'investigator.tabs.underInvestigation')
        value="underInvestigation"
        count=getCountValue
        isActive=isSelected
        onClicked=onTabSelected
        buttonId='underInvestigation' >}}
    </div>
    <div class="col-md-3">
      {{>} counterButton label=(i18n 'investigator.tabs.all')
        value="all"
        count=getCountValue
        isActive=isSelected
        onClicked=onTabSelected
        buttonId='all' >}}
    </div>
  </div>
</template>
```



```
        onClicked=onTabSelected  
        buttonId='all' }}  
  
    </div>  
  </div>  
</template>
```

Investigator-tabs.js

```
TemplateController('investigatorTabs', {  
  props: new SimpleSchema({  
    selected: {  
      type: String  
    },  
    onTabChanged: {  
      type: Function  
    },  
    counts: {  
      type: Object,  
      blackbox: true  
    }  
  }),  
  helpers: {  
    isSelected: function() {  
      return (tab) => this.props.selected === tab;  
    },  
    onTabSelected: function() {  
      return (tabClicked) => this.props.onTabChanged(tabClicked);  
    },  
    getCountValue() {  
      return (tab) => this.props.counts[tab];  
    }  
  }  
});
```



Node

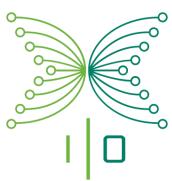
Node.js is a server-side JavaScript runtime environment based on Google's V8 JavaScript engine. Meteor.js is supported by Node.js and its' packages. We are using Node.js on its own in very few places in application, mostly connected with tests.

Example: Cache build and dependencies for CircleCI

```
#!/usr/bin/env node

const path = require('path');
const spawn = require('child_process').spawn;
const baseDir = path.resolve(__dirname, '../..');
const srcDir = path.resolve(baseDir, 'attainenroll');

const cacheMeteor = function() {
  console.log('Caching build & dependencies (can take a while the first time)');
  const childProcess = spawn('meteor', ['--raw-logs', '--settings',
  '../config/testing.json'], {
    cwd: srcDir,
    env: process.env
  });
  childProcess.stdout.setEncoding('utf8');
  childProcess.stderr.setEncoding('utf8');
  childProcess.stdout.on('data', function(line) {
    process.stdout.write(line);
  });
  childProcess.stderr.on('data', function(line) {
    process.stderr.write(line);
  });
  const exitAfterBuild = function exitAfterBuild(line) {
    if (line.indexOf('App running at') !== -1) {
      childProcess.kill();
      console.log('Done caching build & dependencies');
    }
  };
}
```

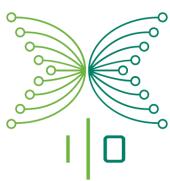


```
    } else if (
      line.indexOf('Your application is crashing') !== -1 ||
      line.indexOf('Errors prevented startup') !== -1) {
        childProcess.kill();
        console.error('There were issues whilst trying to cache build &
dependencies');
        throw new Error(line);
      }
    );
    childProcess.stdout.on('data', exitAfterBuild);
    childProcess.stderr.on('data', exitAfterBuild);
  };

cacheMeteor();
```

References:

- Node.js: <https://nodejs.org/en/>



Mocha.js

Mocha.js is a JavaScript test framework running on Node.js and in browser. We are using it for [unit](#) and [integrations tests](#).

Example: Testing if invoice ticket schema validation is working correctly.

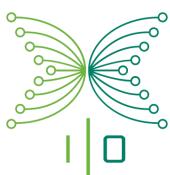
```
describe('Invoice Tickets Schema Validations', function() {

    it('ticket with piiHash is valid', function() {
        const obj = { adaVendingInvitation: { piiHash: 'n8fh3hfeHGfrv89hfuihv8HUIb43h' }};
        const isValid =
Schemas.InvoiceTicket.namedContext("piiHash_validation").validateOne(obj,
"adaVendingInvitation");
        expect(isValid).toBeTruthy();
    });

    it('piiHash gets removed after Schema clean', function() {
        const obj = { buyerId: '1', adaVendingInvitation: { piiHash: 'n8fh3hfeHGfrv89hfuihv8HUIb43h' } };
        const validated = Schemas.InvoiceTicket.clean(obj);
        expect(validated.buyerId).toEqual('1');
        expect(validated.adaVendingInvitation).toBeUndefined();
    });
});
```

References:

- Mocha.js: <https://mochajs.org/>



Chai.js

Chai is an assertion library for node and the browser. We are using it in combination with Mocha.js.

Example:

```
expect(validated.buyerId).equals('1');
expect(validated.adaVendingInvitation).to.be.undefined;
```

References:

- Chai: <http://chaijs.com/>

Chimp.js

Chimp.js is a test framework with good support for meteor and CI(continuous integration). It supports Mocha, Jasmine and Cucumber for writing the tests. In this project we are using Cucumber to write ours. We are using Chimp.js only for end-to-end tests.

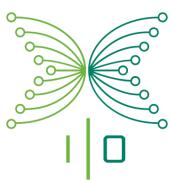
Example: Evaluate if the text in “reviewer-name” is empty. If empty, test will pass.

```
import { waitAndGetText } from '/test/end-to-end/tests/_support/webdriver';

module.exports = function() {
  this.Then(/^the reviewer name is empty$/, function() {
    const name = waitAndGetText(".reviewer-name");
    expect(name).to.eq('');
  });
};
```

References:

- Chimp.js: <https://chimp.readme.io/>



Databases

MongoDB

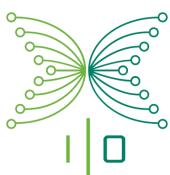
Meteor.js comes with MongoDB as the default database. MongoDB is a NoSQL open-source database, and it's using JSON like objects for schemas.

Example: MongoDB insertion

```
Commissions.insert({  
  distributorId: commission.distributorId,  
  ticketId: invoiceTicket._id,  
  invoiceNumber: invoiceTicket.invoiceNumber,  
  buyerId: invoiceTicket.buyerId,  
  originator: originator,  
  payoutAmount: commission.payoutAmount,  
  payoutTransactionId: invoiceTicket.payoutTransactionId,  
  paidAt: new Date(),  
  createdAt: invoiceTicket.satoshisReceivedAt  
});
```

References:

- MongoDB: <https://docs.mongodb.com/>



Improvement Suggestions

Test Suite Improvement

Because sales application has lots of functionality, testing thoroughly takes a long time. Especially end-to-end tests since running all scenarios takes about 30 minutes. Part of this is because some steps are repeatedly used (for example When I login) which takes substantial amount of time. Because of this, it is important to write a test code that would not fail or else someone who is not responsible for the task may have to fix the problem which wastes their working time.

It is hard to write tests that involves animations, because test suite ignores the animation and when trying to do the next step - it fails, because the contents that the animation should show are not yet displayed. Major example of this is modals. When modals pop up, you have to do an action, such as confirm or cancel. After waiting for the modal to disappear, you continue with the next action. Our tests occasionally fails here, because they don't wait for modal to disappear. We have tried to solve this with various ways and spend a lot of time, but it is still occurring sometimes.

CircleCI greatly helps us reduce our testing time but it is not as fast as running the tests on our own machines, which results in longer tests.

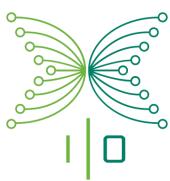
We have discussed about switching over to other packages but some were outdated or has negative impact on our testing environment. It would help us a lot if we can find a better testing suite for improving our test environment for future.

Middleman Attack Yielding

As you have seen, there is a two way communication going on between the sales application and backend-app in order to fulfill the payments and reflect the status on a user level. Given Galaxy environment characteristics, there is no way to set a point-to-point close network between those two applications, what would be ideal, and as a result of this flaw (Galaxy has dynamic IPs) both applications have open endpoints.

Although this was secured on Backend side by adding Token authentication and response encryption, sales application endpoints are completely open. This means that even if every payment is secure and there is no way to manipulate actual money, a middleman attack can mess around with the sales application quite bad, manipulating tickets payments status. Although that would require a deep understanding of its functionality and data.

A possible solution is to add authentication on sales application as well, to verify that any request on the Backend API is from a trusted third party.



Toggle Sorting Button for Invoice Managers

In the exporter, the HIM bundle list the tickets are sorted newest at the bottom, but sometimes it's useful to switch the sorting order to find newest tickets.

Add a sorting button that toggles ascending/descending sorting order to the invoice managers, to allow the invoice manager to choose how to sort the tickets/bundles.

Block Password Reset Emails for X Amount of Time

Currently "Send password reset email" button can be clicked infinite times, and infinite amount of emails will be sent. If a malicious person wants, he would be able to saturate the email service.

Reset email should be sent only when a certain time has been passed before the first email was sent (e.g. 30 mins).

Turn Pay Page into an Invoice Ticket Status Page

Convert the pay page into a ticket state page. The page will show the tickets current and previous stages.

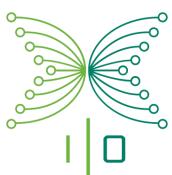
The key point is to show when the ticket

- is created (date, amount)
- has Ada reserved (date)
- need payment (date, amount, address)
- is expired (date)
- is paid (date, amount)
- has passcode assigned (date)

This page is read only.

Show The Buyer's Residence Country on the Invoice Ticket Card in the Invoice Queue

Currently in the invoice queue the card component for invoices doesn't show the buyer's residence country. It would be better if the invoice manager could see that field.



Add Backend Status Info to Sysop Dashboard

Currently some back-end status parameters are not visible in the sysop dashboard, they need to be queried specifically to detect any abnormal situation. A quick overview dashboard could be valuable to have an instant info about a potential issue.

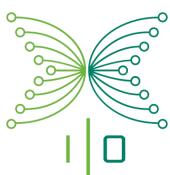
Some useful information could be:

- Health-check data (backend version for example)
- Sales-app notifications queue amount (also details perhaps?)
- White-list queue amount (also details perhaps?)
- Transaction queue amount (also details perhaps?)
- Holding Wallet balance
- Commission Wallet balance
- Search tool by invoice wallet or address

Extend the Region Filter for Compliance Officers

The number of countries where sales are available has increased, thereby we should extend the region filter to include China (and maybe also Vietnam) into the list.

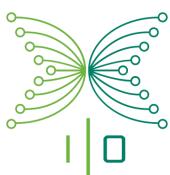
The screenshot shows the 'Account Approval Workflow' interface. At the top, there are several buttons: 'Recent (4)', 'Unreviewed (13)', 'Watching (1)', 'From' (with calendar icons), 'To' (with calendar icons), 'Funds Received (0)', 'CCO (3)', and 'All (782)'. Below these are dropdown menus for 'Q Surname' and a search bar. On the left, there's a 'Filter' section with checkboxes for 'Buyer', 'Distributor', 'Corporation', 'Individual', and four categories labeled '(A)' through '(D+)'. At the bottom, there are sections for 'Countries' with checkboxes for Japan, South Korea, China, Vietnam, and Others.



Sort the Invoice List in Review User Page

The list of invoice tickets in the user review page is seemingly unsorted. When a D+/D+ user is reviewed this can be very confusing for the compliance officer. A more reasonable listing would be a descending sorting by creation date (or similar).

アテイン・ADA交換取次システム ダッシュボード									ログアウト
名前	請求書ID	支払い方法	国	作成日時	金額	入金額（日本円）	サトシでの金額	受け取ったサトシ	
よ黒 今挙	1004040	Btc	JP	9月 8日 2016	\$60,000	¥0	฿95.95394210	฿95.95305526	
よ黒 今挙	220395	Btc	JP	1月 13日 2016	\$52,000	¥0	฿120.84872992	฿120.60208270	
よ黒 今挙	1001880	Btc	JP	8月 2日 2016	\$720,000	¥0	฿1,240.75893087	฿1,241.37931034	
よ黒 今挙	784775	Btc	JP	3月 23日 2016	\$88,000	¥0	฿211.31495533	฿211.42664936	
よ黒 今挙	938119	Btc	JP	3月 11日 2016	\$24,000	¥0	฿57.18370264	฿57.18370264	
よ黒 今挙	748928	Btc	JP	2月 3日 2016	\$23,000	¥0	฿61.68038831	฿61.68038831	

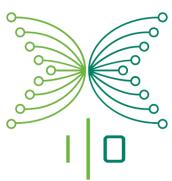


Hover in Birth Date Comparison Table Doesn't Line Up

The user listings under birth date comparison in review user page has a comparison table for the two users. When hovering a field it highlights the field in blue color for both users. But if one of the users has a field that the other lacks, then the comparison doesn't line up in the UI.

The screenshot shows a comparison table between two users, "のし 子よ" and "花ラ 丹ト". The table compares various fields like ID, Account Type, Name, Email, Status, Birth Date, and more. The "パートナー" (Partner) field for "のし 子よ" is highlighted in blue, while the "パートナー" field for "花ラ 丹ト" is also highlighted in blue, causing the alignment to be off.

現在閲覧中		生年月日が同じユーザー	
ID	WECKnLnKLXpBExnc	ID	T24quYHbuNLJnFpkG
アカウント種類	個人、交換希望者	アカウント種類	個人、交換希望者
名前	今參 よ黒	名前	のし 子よ
メールアドレス	test-email+Rebecca.Andrews6331@iohk.io	メールアドレス	test-email+Rebecca.Andrews6331@iohk.io
ステータス	承認待ち	ステータス	拒否済み
生年月日	1969/10/16	生年月日	1969/10/16
コンプライアンス種別 (旧/新)	D+ / D+	コンプライアンス種別 (旧/新)	- / A
交換希望額(米ドル)	\$60.000	パートナー	んさ し雲 <test-email+Gary.Burton655@iohk.io>
パートナー	り童 河飼 <test-email+Maia.Webster8573@iohk.io>	紹介元	花ラ 丹ト <test-email+Derek.Acosta918@iohk.io>
紹介元	り童 河飼 <test-email+Maia.Webster8573@iohk.io>	作成日	10月 31日 2016
作成日	1月 13日 2016	登録時のIP	111.22.333.444
登録時のIP	111.22.333.444	電話番号	090-7849-1121
電話番号	090-9774-4839	言語	ja
言語	ja	居住国	日本
居住国	日本	住所	958-6117 Duis Road 天長市, North Island 120-0034
住所	8865 Ultrices Avenue 안동시, Cartago 143-0013	ユーザーは郵便番号のAPIチェックを通過しましたか?	このユーザーのチェックはまだ終わっていません。
ユーザーは郵便番号のAPIチェックを通過しましたか?		このユーザーのチェックはまだ終わっていません。	



Better Ticket Information in User Summary View

Some information on the enlisted tickets are unnecessary and some are missing for the purpose of being shown in the user summary view.

Unnecessary:

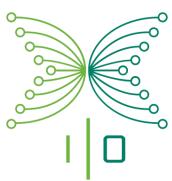
- Name
- State (user status)

Missing:

- Unreserved, Reserved, Paid, Canceled
- State (invoice ticket)

氏名	請求書ID	支払い方法	承認状況	作成日時	金額	入金額（日本円）	サトシでの金額	受け取ったサトシ
Hiroto Shioi	1008662	Bank	✓	2016-12-27	\$1,000	¥0	฿0	฿0

氏名	請求書ID	支払い方法	承認状況	作成日時	金額	入金額（日本円）	サトシでの金額	受け取ったサトシ
Hiroto Shioi	1008675	Btc	✓	2017-01-19	\$10	¥0	฿0.01132502	฿0

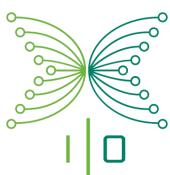


Special Holiday Settings

More than once manual work has been done to prevent ticket expiration due to holidays. These tasks are time consuming and prone to errors.

Add an array field `public.expireTickets.nonExpirationHolidays` to the settings file preventing ticket expiration for the given days. The program should handle the listed days in the same way, regarding ticket expiration, as it currently handles weekends when `public.expireTickets.X.afterInvoiceSent.businessDays` is set to `true`.

```
"expireTickets": {  
    "bank": {  
        "whilePendingCompliance": {  
            "days": 5,  
            "businessDays": true  
        }  
        "afterInvoiceSent": {  
            "days": 3,  
            "businessDays": true  
        }  
    },  
    "btc": {  
        "whilePendingCompliance": {  
            "days": 5,  
            "businessDays": false  
        }  
        "afterInvoiceSent": {  
            "days": 3,  
            "businessDays": false  
        }  
    },  
},
```

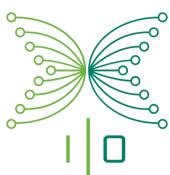


Add Expiration for Users Pending Compliance

This feature was already planned to be implemented some time ago, but has yet to be done.

Add an field `public.expireTickets.X.whilePendingCompliance` similar to `public.expireTickets.X.afterInvoiceSent` to the settings file, to expire tickets in the states `saleStartedBank` and `saleStartedBtc`. The program should handle the expiration in the same way as for `public.expireTickets.X.afterInvoiceSent`.

```
"expireTickets": {
    "bank": {
        "whilePendingCompliance": {
            "days": 5,
            "businessDays": true
        }
        "afterInvoiceSent": {
            "days": 3,
            "businessDays": true
        }
    },
    "btc": {
        "whilePendingCompliance": {
            "days": 5,
            "businessDays": false
        }
        "afterInvoiceSent": {
            "days": 3,
            "businessDays": false
        }
    }
},
```



Known Bugs

Zip Api Check not Always Saving During Enrolment

During enrolment, when a user fills in a zip code that the google api doesn't recognize, a checkbox shows up that forces a user to acknowledge the zip code.

This however, seems to fail to save to the user document sometimes for an unknown reason.

When the Pricing Providers are Offline, then the Application Will Set satoshisExpected, centsAskPrice and btcUsd to 0

This has its ups and downs, it can be good as it prevents purchases using outdated rate, but there is currently no way to for the frontend-backend interaction to solve a situation where a user pays a ticket while the pricing providers are down.

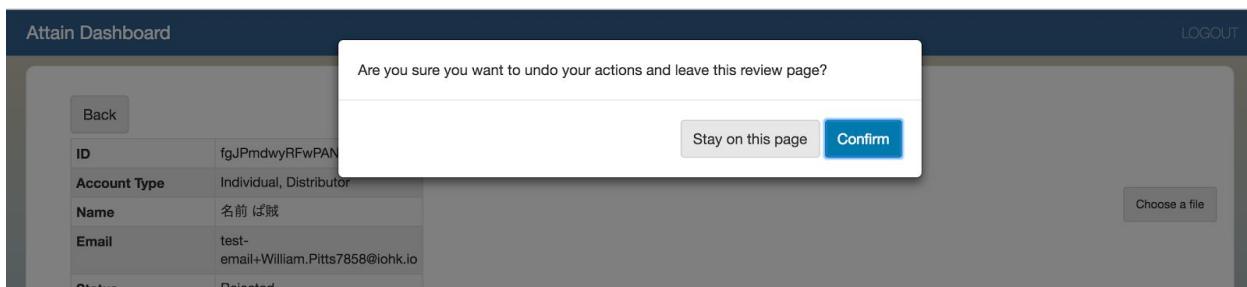
When Reviewing a User, and CO Rotates the Image, the Next Image Viewed Will Also Be Rotated

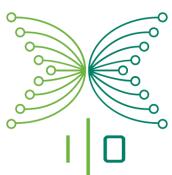
When you login as a compliance officer and rotate a picture in review-user view, all other pictures will be display with the same rotation when clicked.

When You Login as an Investigator and Press “Back” After Reviewing, Misleading Message Appears

As an Investigator, when you enter to review a user and then press Back, the message says:

“Are you sure you want to undo your actions and leave this review page?” which could be misleading.





When Commissions are Made, the Origin's Email are Stored in the Document and Remains Unchanged Even If User Change Their Email Address

When user pays the ticket and commissions are made, origin's email are stored in the commission document. This means that even if user changes their email address, commission's origin will remain unchanged. This may cause confusion to the distributors.

During Logout or When Switching Views, Console Errors Can Appear

When logging out as an user, error messages appear on the console log. However, this does not affect the application's performance.

The Payment Information on the Email and Payment Link for Bank Tickets are Inconsistent

Here is the current format of the payment link for bank tickets.

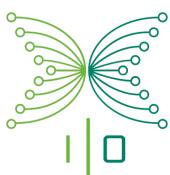
Email

Payment amount: Requested amount + Attain fee.

Payment Link

Payment amount: Requested amount.

Even if the payment link isn't sent out any more, the conflicting information can be confusing if someone enters it.



'RegenerateAdaPasscode' Button is Still Visible Even Though the Ticket is in the State Where Passcode Cannot Be Regenerated

User can see the 'Regenerate ADA Code' button on the tickets where the state is not 'receiptSent' and cannot regenerate passcodes. They can try to regenerate them but they'll get an error 'Please specify a message' on the console log.

A screenshot of a ticket detail page. The page has a header with columns: INVOICE #, PAYMENT METHOD, CREATED AT, ORDERED AMOUNT, YEN RECEIVED, SATOSHIS EXPECTED, SATOSHIS RECEIVED, and ADA PASSCODE GENERATIONS. Below the header is a table row with data: 1008701, Btc, 2017-01-26, \$10, ¥0, ₿0.01117256, ₿0.02117256, and 2. At the bottom of the table, there is a red button labeled 'Regenerate ADA Code'.

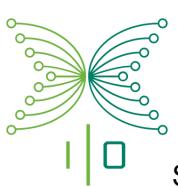
Counter On the Tab and Search Result Counter Show Different Number in Compliance View

When you login as a Compliance officer, the counter on the tab and search result show different numbers. This is because the counter on the tab is not counting the users in which their residenceCountry is neither "Japan" or "Korea".

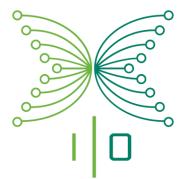
A screenshot of the Compliance view. At the top, there is a title 'アカウント承認リスト'. Below it is a filter section with checkboxes for '交換者' (checked), '代理人' (checked), '法人' (checked), '個人' (checked), '(A)' (checked), '(B)' (checked), '(C)' (checked), '(D+)' (checked), '日本' (checked), '大韓民国' (checked), and 'その他' (checked). To the right of the filters are buttons for '最近 (0)', '未評価 (10)', '監視中 (4)', and a date range selector 'から' and 'まで'. Below the filters is a search bar with a dropdown for '姓' and a '検索' button. A red box highlights the '全部 (1067)' button. At the bottom left is a link '承認記録を確認する'. At the bottom right, it says '検索結果: 1128'.

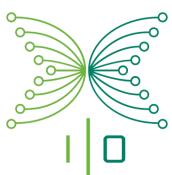
User Can Enter String on Phone Field on Enrollment

Buyer and Distributor can enter String on phone number field in enrollment. This means that user can enter emails, sentences etc. and still get their personal information valid on enrollment.

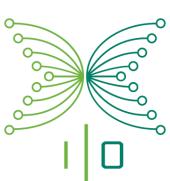


Frontend - 3rd Party Software and Services

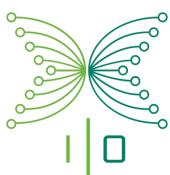




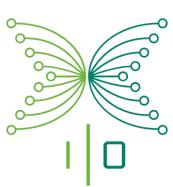
Services	6
Loggly	6
Mailgun	6
Compose	6
Galaxy	6
Firepoker.io	7
Trello	7
Github	7
S3	7
Slack	8
Trello	8
Hangouts	8
Packages	9
Meteor	9
alanning:roles	9
aldeed:autoform	9
aldeed:autoform-bs-datepicker	10
craphtex:autoform-wizard	10
craphtex:autoform-wizard-iron-router	10
aldeed:simple-schema	11
aldeed:collection2	11
anti:i18n	11
dburles:collection-helpers	11
reactive-dict	12
simple:reactive-method	12
space:template-controller	12
iron:router	12
rajit:bootstrap3-datepicker	13
jparker:crypto-core	13
jparker:crypto-md5	13
mizzao:bootboxjs	13
meteorhacks:ssr	14
momentjs:moment	14
sacha:spin	14
zimme:active-route	14
huttonr:bootstrap3	15



edgee:slingshot	15
peerlibrary:aws-sdk	15
jjman505:wheelzoom	15
stevezhu:lodash	16
ongoworks:security	16
natestrauser:statemachine	16
numeral:numeral	16
jjman505:bitcoin-address	17
jjman505:moment-business	17
reywood:publish-composite	17
vsivsi:job-collection	17
rzymek:moment-locale-ja	18
gwendall:simple-schema-i18n	18
gwendall:autoform-i18n	18
froatsnook:sleep	18
sales:invoice-manager	19
meteorhacks:subs-manager	19
u2622:persistent-session	19
tsega:bootstrap3-datetimepicker	19
icellan:sjcl	20
meteorhacks:aggregate	20
practicalmeteor:mocha	20
practicalmeteor:chai	20
dburles:factory	21
xolvio:cleaner	21
practicalmeteor:mocha-xunit-reporter	21
Meteor (Potentially Unused)	21
ca333:bitcoinjs	21
classcraft:meteor-wkhtmltopdf	22
jparker:crypto-aes	22
jparker:crypto-sha256	22
tmeasday:publish-counts	22
matb33:collection-hooks	22
meteorhacks:kadira	23
meteorhacks:kadira-profiler	23
percolate:migrations	23
xolvio:backdoor	23

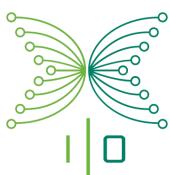


Node	24
babel:babel-runtime	24
winstonjs:loggly	24
Node (Dev)	25
michaelzoidl:babel-root-import	25
chaijs:chai-http	25
xolvio:chimp	25
marak:faker	26
lodash:lodash	26
moment:moment	26
jesselane:shell-source	27
practicalmeteor:spacejam	27
eslint:eslint	27
Node (Potentially Unused)	28
yannickcr:eslint-plugin-react	28
babel:babel-eslint	28
Software	29
Code Editors	29
Visual Studio Code	29
Webstorm	29
Atom	29
Sublime	30
Database Admin UIs	30
Mongobooster	30
Robomongo	30
Git	30
Git Bash	31
GPG	31
Hub	31
Browser Plugins	32
Trello	32
Card Numbers for Trello	32
Trelabels	32
Github	32
Github Projects Like Trello Cards	32
Minimongo	32
Minimongo Explorer for Chrome	33



Licenses

33



Services

Summary of 3rd party services which the sales application is relying on.

Loggly

Loggly is a cloud-based log management service provider. It does not require the use of proprietary software agents to collect log data. The service uses open source technologies, including Elasticsearch, Apache Lucene and Apache Kafka.

To use Loggly you only need to create an account on their web and configure some settings in the configuration file (for more information see the Loggly section under System Configuration).

References:

- Loggly: <https://www.loggly.com/>

Mailgun

Mailgun is an API that allows Sales application to send emails. This feature is widely used throughout the entire application. Mailgun starts charging a fee when more than 10.000 emails are sent in a month, so it's very important to be cautious with the usage.

References:

- Maingun: <https://www.mailgun.com/>

Compose

Compose is a database as a service platform for securely hosting and managing MongoDB databases. It's easy to setup and use. You can manage access rights for developers to the database, do monitoring, manage backups and lookup data using the web GUI.

References:

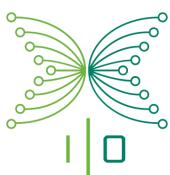
- Compose:
<https://www.ibm.com/analytics/us/en/technology/cloud-data-services/compose/>

Galaxy

Galaxy is a platform as a service built specifically for Meteor that allows ease of deployment on AWS servers and scaling.

References:

- Galaxy: <https://www.meteor.com/hosting>



Firepoker.io

Fire poker is used in the [SCRUM](#) method of working on projects. It can help making time-estimates if your team is spread out over different locations. It uses cards as representatives for time.

References:

- Firepoker.io: <http://firepoker.io/>

Trello

Trello is a collaboration tool that organizes your projects into boards. In one glance, Trello tells you what's being worked on, who's working on what, and where something is in a process.

References:

- Trello: <http://trello.com>

Github

GitHub is a web-based Git repository hosting service. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.

References:

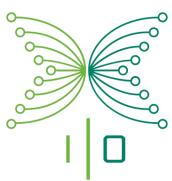
- Github: <https://github.com/>

S3

Amazon Simple Storage Service (Amazon S3) is object storage with a simple web service interface to store and retrieve any amount of data from anywhere on the web.

References:

- Amazon S3: <https://aws.amazon.com/s3/>



Slack

Slack is a tool to communicate with all members in a company, it supports chat groups and a multitude of different functions regarding notifications of other platforms.

References:

- Slack: <https://slack.com/>

Trello

Trello is a collaboration tool that organizes your projects into boards. In one glance, Trello tells you what's being worked on, who's working on what, and where something is in a process.

References:

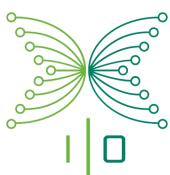
- Trello: <http://trello.com>

Hangouts

Hangouts can be used to have online audio/video conferences. It can be used to share your screen and have others looks.

References:

- Hangouts: <https://hangouts.google.com/>



Packages

For the application we use many 3rd party modules for either Meteor or Node. Any pieces of software used from sources other than our own will be listed here.

Meteor

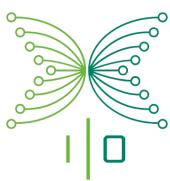
This is a list of all 3rd party meteor packages used in the project, their description, use cases in our application, version number and important legal notes.

alanning:roles

<i>Description</i>	Authorization package for meteor, which allows generating roles and attaching permissions to them.
<i>Usage</i>	Throughout the entire application as the basis for the role management.
<i>Version</i>	1.2.15
<i>License</i>	MIT

aldeed:autoform

<i>Description</i>	Autoform is a meteor package for automatically creating form UI elements based on database schema.
<i>Usage</i>	Enrollment page, reorder page and many other pages, where input forms are used.
<i>Version</i>	5.8.1
<i>License</i>	MIT



aldeed:autoform-bs-datepicker

Description Add-on for the autoform package. Adds in bootstrap-datepicker as custom input type.

Usage Enrollment and user info pages.

Version 1.1.1

License MIT

craphtex:autoform-wizard

Description AutoForm Wizard is a multi step form component for AutoForm (based on the work of forwarder).

Usage Allowing a multi step-form in enrollment.

Version 0.9.1

License MIT

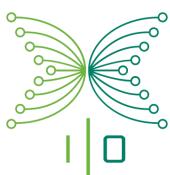
craphtex:autoform-wizard-iron-router

Description Iron router binding package for autoform wizard, based on the work of forwarder.

Usage Allowing a multi step-form in enrollment.

Version 0.1.4

License MIT



aldeed:simple-schema

Description Simple reactive schema validation package, that can be used together with Collection2 and AutoForm packages or on its own.

Usage In combination with collection2 and also for props validation in template controller.

Version 1.5.3

License MIT

aldeed:collection2

Description This package allows you to attach schema to a collection, so that every time the collection is updated, schema validation is run.

Usage All of our collections are using this package.

Version 2.9.1

License MIT

anti:i18n

Description Simple reactive i18n package for meteor.

Usage All of our translations are using this i18n package.

Version 0.4.3

License MIT

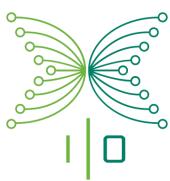
dburles:collection-helpers

Description Allows adding helper functions for collections.

Usage Helper functions for users, refs and many other collections.

Version 1.0.4

License MIT



reactive-dict

Description Adds a general-purpose reactive datatype for use with tracker.

Usage In some cases used as replacement for session variable.

Version 1.1.8

License MIT

simple:reactive-method

Description Allows Meteor calls from helpers.

Usage Various parts of application

Version 1.0.2

License MIT

space:template-controller

Description Syntactic sugar for blaze templates.

Usage All blaze templates are using this.

Version 0.3.0

License MIT

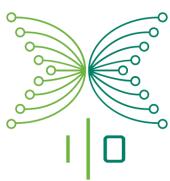
iron:router

Description Router for meteor.

Usage Managing routes for our application.

Version 1.0.12

License MIT



rajit:bootstrap3-datepicker

Description Bootstrap datepicker meteor package.

Usage Dependency for aldeed:autoform-bs-datepicker.

Version 1.5.1

License MIT

jparkr:crypto-core

Description Meteor package for CryptoJS library (collection of standard and secure cryptographic algorithms).

Usage As dependency for other crypto related packages.

Version 0.1.0

License New BSD License

jparkr:crypto-md5

Description md5 algorithm from CryptoJS.

Usage Generating refcodes.

Version 0.1.1

License New BSD License

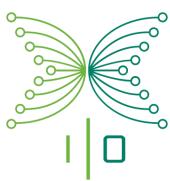
mizzao:bootboxjs

Description Bootboxjs allows us to generate modal dialogs.

Usage It is used to create dialogs in the sales application.

Version 4.4.0

License MIT



meteorhacks:ssr

Description Server side rendering for blaze templates.

Usage Rendering emails.

Version 2.2.0

License MIT

momentjs:moment

Description JavaScript date library for parsing, validating, manipulating, and formatting dates

Usage Various date related operations.

Version 2.13.1

License MIT

sacha:spin

Description Adds spinning icon to the application.

Usage We use a spinning icon as a loading indicator on application.

Version 2.3.1

License DWYWWWT(Do Whatever You Want With This)

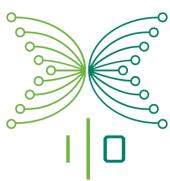
zimme:active-route

Description This package provides helpers for figuring out if some route or path is or isn't the currently active route.

Usage Sidebars for Buyer, Distributor, Admin to indicate which page they're currently viewing.

Version 2.3.2

License MIT



huttonr:bootstrap3

Description Bootstrap3 (HTML, CSS, and JS framework) package for meteor.

Usage Design for the application.

Version 3.3.6_13

License MIT

edgee:slingshot

Description Package that upload files to Amazon S3.

Usage Upload and store documents that buyers and distributors need for compliance.

Version 0.7.1

License MIT

peerlibrary:aws-sdk

Description AWS SDK Meteor package. Adds AWS object for use with multiple AWS services into the global scope.

Usage Authorization for connecting to AWS.

Version 2.2.42_1

License Apache 2.0

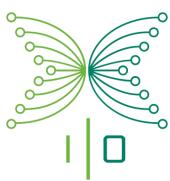
jjman505:wheelzoom

Description A simple tool allowing you to zoom images with your scroll wheel.

Usage Zoom in document pictures.

Version 1.0.0

License Built in-house



stevezhu:lodash

Description A meteor package for lodash.

Usage Various parts of application (JavaScript utility library).

Version 3.10.1

License Lodash

ongoworks:security

Description API for defining write security on your MongoDB collections.

Usage Preventing updates to user db.

Version 1.3.0

License MIT

natestrauser:statemachine

Description StateMachine.js packaged for meteor.

Usage Controlling state of invoice tickets.

Version 2.3.5

License MIT

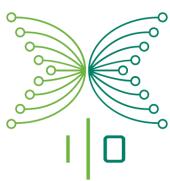
numeral:numeral

Description A library for formatting and manipulating numbers and is used for formatting currencies.

Usage Formating btc, ada, usd amounts.

Version 1.5.3

License MIT



jjman505:bitcoin-address

Description Package for simple bitcoin address validation.

Usage Bitcoin address validation in various parts of application.

Version 0.1.0

License Built in-house

jjman505:moment-business

Description Package for working with weekdays and weekends in Moment JS.

Usage Calculating invoice ticket expiration.

Version 2.0.0

License MIT

reywood:publish-composite

Description Allows publishing a set of related documents from various collections using a reactive join.

Usage In multiple publications used in the application.

Version 1.4.2

License LGPL 3.0

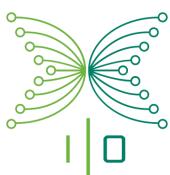
vsivsi:job-collection

Description Reactive job queue for Meteor, supporting distributed workers that can run anywhere.

Usage Our workers are using the job collection.

Version 1.3.3

License MIT



rzymek:moment-locale-ja

Description Localization package for moment.js in Japanese.

Usage This package is used when user selects Japanese as an language.

Version 2.12.0

License MIT

gwendall:simple-schema-i18n

Description Package for internationalization of alded:simple-schema error messages. Using this package will show translated error message on auto generated forms.

Usage All of our forms are using this package.

Version 0.2.1

License None

gwendall:autoform-i18n

Description This package will automatically attach translated labels, placeholders and select options to forms that are created through Autoform.

Usage All form validations are using this for getting the translations.

Version 0.1.9

License None

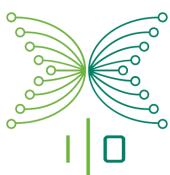
froatsnook:sleep

Description This package lets you sleep the application while doing an async operation.

Usage Used in reorder page to put in a pause before throwing out error about email not being found. Used to prevent spamming of the page to try and find emails that exist.

Version 1.2.0

License MIT



sales:invoice-manager

Description Package to calculate order fulfillment amount.

Usage Updating payment exports as a invoice manager.

Version 1.0.0

License MIT

meteorhacks:subs-manager

Description Subs-manager allows you to cache subscriptions on the client side. This will prevent the Sales application from slowing down when resubscribing the same data.

Usage Subs manager are used in views where the same data is fetched multiple times; such as Invoice Manager and Compliance Officer queue.

Version 1.6.4

License MIT

u2622:persistent-session

Description This package will allow Session variable persistence through page refresh.

Usage Session variables are used to save temporary data on client side of the sales application.

Version 0.4.4

License MIT

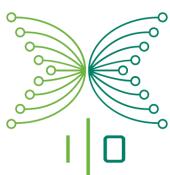
tsega:bootstrap3-datetimepicker

Description Bootstrap 3 DateTime picker from Eonasdan, packaged for Meteor.js.

Usage Invoice queue.

Version 4.17.37_1

License MIT



icellan:sjcl

Description Sjcl stands for Stanford Javascript Crypto Library. It is a Javascript library for cryptography.

Usage This is used for encrypting API response for the backend.

Version 1.2.0

License BSD-2-Clause

meteorhacks:aggregate

Description This package allows you to aggregate collections on server side.

Usage Aggregations are used in many parts of the applications such as query for suspicious distributors, calculating compliance tier and such.

Version 1.3.0

License MIT

practicalmeteor:mocha

Description A package for running meteor app and package tests with Mocha.

Usage Integration, unit tests.

Version 2.4.6-rc.3

License MIT

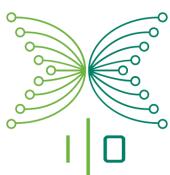
practicalmeteor:chai

Description The Chai Assertion Library, packaged for meteor.

Usage Assertions in integration, unit tests.

Version 2.1.0_1

License MIT



dburles:factory

Description A package for creating test data or for generating fixtures.

Usage Generating users, invoice tickets etc, for test cases.

Version 1.1.0

License MIT

xolvio:cleaner

Description This package clears your entire database for testing purposes.

Usage Clearing up the database before running test cases.

Version 0.3.1

License MIT

practicalmeteor:mocha-xunit-reporter

Description XUnit reporter for mocha.

Usage When running integration tests.

Version 0.1.0-rc.2

License None

Meteor (Potentially Unused)

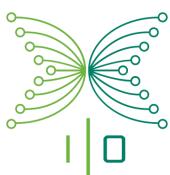
This is a list of potentially unused 3rd party meteor packages. We decided to keep them, just to make sure that we don't break any functionality because of the chance that they might be dependencies of other packages.

ca333:bitcoinjs

Description Javascript library for bitcoin packaged for meteor.

Version 1.6.0

License MIT



classcraft:meteor-wkhtmltopdf

Description Allows PDF creation from HTML.

Version 0.3.1

License MIT

jparkr:crypto-aes

Description AES algorithm from CryptoJS.

Version 0.1.0

License New BSD License

jparkr:crypto-sha256

Description SHA256 algorithm from CryptoJS.

Version 0.1.0

License New BSD License

tmeasday:publish-counts

Description Reactive counter for publications.

Version 0.7.3

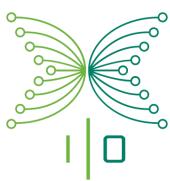
License MIT

matb33:collection-hooks

Description Add before and after hooks for the collection.

Version 0.8.1

License MIT



meteorhacks:kadira

Description Performance Monitoring for Meteor.

Version 2.30.0

License MIT

meteorhacks:kadira-profiler

Description This package makes CPU profiles of the app for inspection and analyzation.

Version 1.3.0

License MIT

percolate:migrations

Description Localization package for moment.js in Japanese.

Version 0.9.8

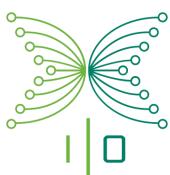
License MIT

xolvio:backdoor

Description Allows running arbitrary code on the server using a Meteor method.

Version 0.2.0

License None



Node

These packages are required to run the application.

babel:babel-runtime

Description The package does three things:

- Automatically requires `babel-runtime/regenerator` when you use generators/async functions.
- Automatically requires `babel-runtime/core-js` and maps, ES6 static methods and built-ins.
- Removes the inline babel helpers and uses the module `babel-runtime/helpers` instead.

Usage Essential for being able to run the application.

Version 6.20.0

License MIT

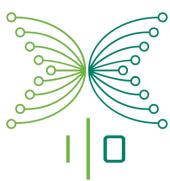
winstonjs:loggly

Description The package is a client implementation for [Loggly](#) in node.js.

Usage We are using it to log all the server-side activity.

Version 1.1.1

License MIT



Node (Dev)

These packages are required only for development purposes as: Javascript transpilation, tests, code-styling, etc.

michaelzoidl:babel-root-import

Description This package is a babel plugin to add the opportunity to use “import” and “require” with root based paths.

Usage Setting the root path suffix of babel to “attainenroll”.

Version 3.2.2

License MIT

chaijs:chai-http

Description HTTP integration testing with Chai assertions.

Usage Used in our project for integration tests.

Version 2.0.1

License MIT

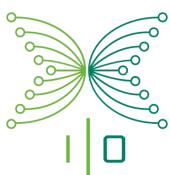
xolvio:chimp

Description This package allows us to write and run unit and integration automation tests for the application.

Usage End-to-end tests.

Version 0.41.2

License MIT



marak:faker

Description Fake contextual data generator.

Usage We use this package to generate fake contextual data that serves to set up automation tests.

Version 3.1.0

License MIT

lodash:lodash

Description JavaScript library for easing work with arrays, numbers, objects, strings, etc.

Usage We use lodash to write tests in a more readable way when we manipulate arrays, objects and composite functions in JS.

Version 4.16.2

License MIT

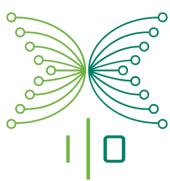
moment:moment

Description This is a date library for parsing, validating, manipulating, and formatting dates.

Usage We use this for any date related solutions.

Version 2.15.1

License MIT



jesselane:shell-source

Description This package provide source environment variables from a shell script into a Node.js process.

Usage End-to-end test launch script

Version 1.1.0

License WTFPL

practicalmeteor:spacejam

Description Package to run meteor package tests and mocha tests from the command line with phantomjs.

Usage We use this in our continuous integration environment (CircleCI).

Version 1.6.2-rc.4

License MIT

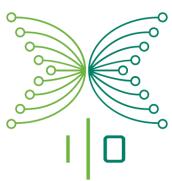
eslint:eslint

Description ESLint is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code.

Usage This gives us a nice error or warning when the code is deviating from the standard.

Version 2.11.1

License MIT



Node (Potentially Unused)

yannickcr:eslint-plugin-react

Description React specific linting rules for ESLint.

Usage This package is useless, the application doesn't use any react package.

Version 5.1.1

License MIT

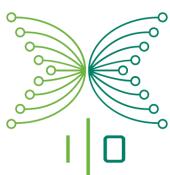
babel:babel-eslint

Description ESLint using Babel as the parser.

Usage This package is useless, the eslint package actually supports ES2015/ES2016/ES2017, JSX, and object rest/spread by default now.

Version 6.0.4

License MIT



Software

These softwares are recommendations based on what the development team used for this project during the year of 2016. These tools may change in the future, so see these as something to look at while considering other options.

The following descriptions and references represent current tools state (the end of 2016) and might be out of date.

Code Editors

Editors are used to create and maintain the code, a good editor will help speed up the development process.

Visual Studio Code

Visual Studio Code is a Cross-Platform source code editor developed by Microsoft. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring.

Despite it being free and open-source, the official download is under a proprietary license. Also any third party component or plugin can have different license agreements than the ones published from Microsoft.

References:

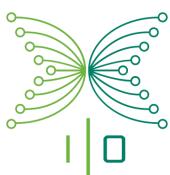
- Visual Studio Code: <https://code.visualstudio.com/>

Webstorm

Webstorm is one of the most popular tools for web-development using Javascript, it's an IDE that works for all OS's. However this tool is not free.

References:

- Webstorm: <https://www.jetbrains.com/webstorm/>



Atom

Atom is a Cross-Platform text and source code editor. It has support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. It is free and open-source, and also most of the extending packages are community-built and maintained.

References:

- Atom: <https://atom.io/>

Sublime

Sublime is another Cross-Platform source code editor. It has a very good performance and its functionality can be extended with plugins too.

Despite being a proprietary software, the free version is good enough.

References:

- Sublime: <https://www.sublimetext.com/>

Database Admin UIs

Mongobooster

Mongobooster is the database management tool that most of the project's developers used. It has support for useful Javascript libraries such as *ES2015*, *Lodash*, *ShellJs*, *Moment.js* and *Mathjs*. It has a free version and a paid one for extra features. It also available for all popular Operating Systems.

References:

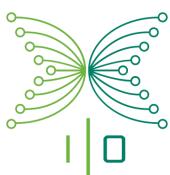
- Mongobooster: <https://mongobooster.com/>

Robomongo

Robomongo is a fast and lightweight database management tool, it is faster than *Mongobooster*, however it lacks some of its features. Should not be used for updating large amount of documents at once.

References:

- Robomongo: <https://robomongo.org/>



Git

Git is the versioning tool used for this project, to be able to easily maintain the project there are a few programs that can be used.

Git Bash

Git bash nice feedback on which branch you are, if there are changes, etc. It gives a console that you can enter your commands in.

References:

- Git bash: <https://git-scm.com/downloads>

PGP

PGP allows git users to sign their commits, if you want to be sure that the correct person committed changed to the code, this tool is necessary.

References:

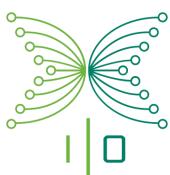
- For Linux: <https://www.gnupg.org/download/index.html>
- For Mac: <https://gpgtools.org/>
- For Windows: <https://www.gpg4win.org/download.html>

Hub

Hub is used to convert a github issue to a pull-request. This tool is not necessary, but very handy if you want to keep the project clean. Also note that the conversion feature is deprecated and might not be usable in future.

References:

- Hub: <https://github.com/github/hub>



Browser Plugins

Trello

Plugins for trello are almost a necessity as they add good functionality that is still missing.

Card Numbers for Trello

This is handy to easily find cards by card number in the overview.

References:

- Card numbers for trello:
<https://chrome.google.com/webstore/detail/card-numbers-for-trello/ddadhlcejiholmdihihdcoapdfkhicn>

Trelabels

This shows labels with the text in there instead of just the colors.

References:

- Trelabels:
<https://chrome.google.com/webstore/detail/trelabels-for-trello/annjdmkbhchmobehkcflecnlhibedbj>

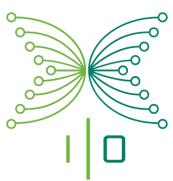
Github

Github Projects Like Trello Cards

This plugin shows a modal instead of a new page when clicking on an issue in a project, it also adds easy to copy commands.

References:

- Github projects like Trello cards:
<https://chrome.google.com/webstore/detail/github-projects-trello-li/cpdlcidglpjcamneedpcbjbkffigkhg>



Minimongo

Minimongo Explorer for Chrome

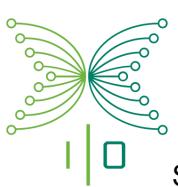
This Chrome Extension provides a lightweight manager to the mini-mongo database (meteor publication structure) which contains client side served data, it's very useful tool to find data leaks.

References:

- Minimongo explorer for chrome:
<https://chrome.google.com/webstore/detail/meteor-minimongo-explorer/bpbalpgdnkieljogofnfjmgcnjcaiheg?hl=en>

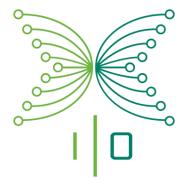
Licenses

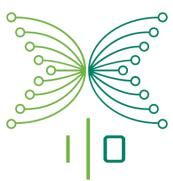
The licenses used in the application for any of the packages and 3rd party software will be supplied in the code base as an additional folder.



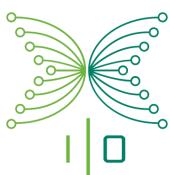
Sales App | Frontend - 3rd party software and services

Backend - Transaction Fees structure





Introduction	3
BackEnd Transactions Types	3
Fees Calculation	3
Fees Application	4



Introduction

This document gives a basic idea of the fee management for the IOHK Backend System. The reader should know the structure of that system in order to follow the explanation.

BackEnd Transactions Types

There are essentially three Transactions Types in the app:

1. **Holding**: Moves funds from one Holding Wallet address to one or more Invoice Wallets Addresses (could be either to the same Wallet or different ones)
2. **Invoice Split**: Moves funds from one invoice address to the Exodus address and the corresponding Commission address
3. **Commission Payout**: Moves funds from one Commission address to Distributors addresses (up to four).

Each of these movements requires bitcoin transaction fees to be paid, and as each order Balance must close to zero, it's important to handle them carefully.

Fees Calculation

Currently Bitgo Service is used estimate the “*Fees per kilobyte*”, using the method described in Bitgo documentation¹.

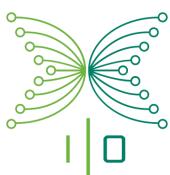
And then, the Tx Kbs are calculated using the following formula:

```
txKb = inputsAmount * 148 + outputsAmount * 34 + 10 + inputsAmount
```

Based on [this](#)² Stackexchange post.

¹ <https://www.bitgo.com/api/#estimate-transaction-fees>

² <http://bitcoin.stackexchange.com/questions/1195/how-to-calculate-transaction-size-before-sending>



Fees Application

Each Transaction Type takes the fees amount from different places, let's review each one:

1) Holding Transaction Fee

As Holding movements are supposed to be spaced over time, meaning that by the time a transaction is performed the previous one is completed and confirmed, the [risks](#)³ of using a fixed “fee address” to takes fees from, and using that same address as “change address” (to be able to reused it) should be mitigated; by the “atomic” and isolated nature of these transactions.

There is a fixed holding address (in Wallet chain = 1, internal) with some funds, and this address unspents are appended to the Transactions Inputs. Where in position one, you have the Holding Address with the amount you want to move. This same address is set as the change address, so when the transactions is completed, the change amount will be put back in the fee address for next transaction.

Example:

Fees: 0.001 BTC

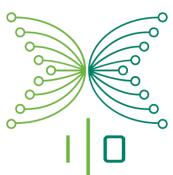
Input:

Holding Address X	50 BTC
Holding Fee Address	1 BTC

Output:

Invoice Address 0	Sum
...	50 BTC in Total
Invoice Address N	
Change Address	0.999 BTC

³ <http://bitcoin.stackexchange.com/questions/40670/spending-unconfirmed-output-from-a-change-address>



2) Invoice Split Fee

This movement splits an Invoice Address **totals** into an exodus address (80%) and a particular commission address (20%) associated with it. The fees are taken from the commission portion.

Example:

Fees: 0.001 BTC

Input:

Invoice Address X	10 BTC
-------------------	--------

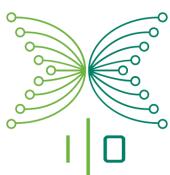
Only one Invoice Address, all funds are split

Output:

Exodus Address	8 BTC
Commission Address X	1.999 BTC

Note that there is no Change Address, cause the Transaction closes to zero

Invoice Address Amount = Exodus Address Amount + Commission Address Amount + Tx Fee



3) Commission Payout Fee

This Transaction spreads a Commission Address amount **total** into some (one to four) distributor's address/es. The fees paid for these transactions, as well as the Split Tx's one, are distributed evenly for each distributor.

Example:

Imagine this ideal scenario; Invoice receives 10 BTC and the Commission must be paid equally amongst two distributors. In an ideal scenario with no fees, the exodus should receive 8 BTC, Commision 2 BTC and then Distributor 1 BTC each. In reality, each Tx costs a fee, so that the operations would be:

Invoice Receives	Invoice Splits	Commission Payout
	Fee = 0.001	Fee = 0.003
Invoice X 10 BTC	Exodus 8 BTC Commission X 1.999 BTC	Distrib 1 0.998 BTC Distrib 2 0.998 BTC

Where 0.998 comes from

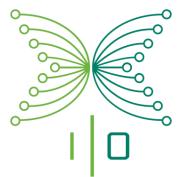
$$\text{Dist Total} - (\text{Split Fee} + \text{Payout Fee}) / \# \text{Distributors} = \\ 1 - (0.001 + 0.003) / 2 = 0.998$$

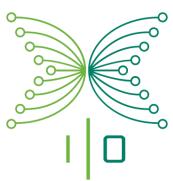
In this case $(\text{Split Fee} + \text{Payout Fee}) / \# \text{Distributors}$ is a decimal number, it would be floored, and the last distributor would compensate the extra reminder. For example, if we had 4001 satoshis of fees to be distributed between two address, the first one would take 2000 satoshis and the last one the remaining 2001.

Notice that these Transactions are balanced, all inputs are spend completely, meaning no change address is needed.

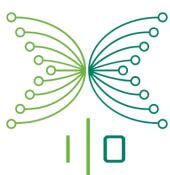
$$\text{Commission X Amount} = \sum_i (\text{Distrib}_i \text{ Address Amount}) + \text{Tx Fee}$$

Backend - System Setup





System requirements	3
Operating system	3
Node	3
NPM	3
PostgresQL	3
System Configuration	4
Settings file	4
Settings breakdown	4
Routes Access Permissions	7
Deployment	8
Encrypting Private Settings	9
Access Token generation	11
Recommendations	11



System requirements

Operating system

Any OS that runs NodeJs and Postgres might be used. Otherwise, if Docker container service is going to be used to run the application, docker should be installed.

We recommend Ubuntu to be used to run the backend app.

Node

The Node version we're currently using in this project is 4.2.3. In order to download this version, you can either download it from their web or use Node Version Manager (nvm), following the tutorial listed below.

References:

- Node download: <https://nodejs.org/download/release/v4.2.3/>
- NVM tutorial:
<http://stackoverflow.com/questions/7718313/how-to-change-to-an-older-version-of-node-js>

NPM

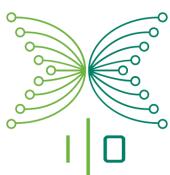
Node Package Manager is the best tool for importing and updating external packages. The current version in this project is 2.14.7, and it's automatically installed when installing Node, but in case you need to install it manually, you can download it (see the references for the link).

Please make sure to run `npm install --save` once it's installed, so it downloads all the dependencies defined in `package.json`.

References:

- NPM download: <https://registry.npmjs.org/npm/-/npm-2.14.7.tgz>

PostgresQL



The database used is psql (PostgreSQL) version 9.4.8. PostgreSQL is a powerful, open source object-relational database system.

References:

- PostgresSQL download for Ubuntu: <https://www.postgresql.org/download/linux/ubuntu/>

System Configuration

Settings file

Backend application settings are divided in two sections when sensitive data needs to be added. A plain javascript file contains all the basic structure and configuration, and a private encrypted json file that extends those basic settings with private ones.

Each environment will have its corresponding section; *development*, *test* and *ci* are all public and don't have encrypted data, while *staging* and *production* does include them. The configuration can be found in *src/conf/environment/*.

Any key on the private section overrides the public one while merging, so any key can be arbitrarily moved from public to private as desired.

Settings breakdown

DB Connection

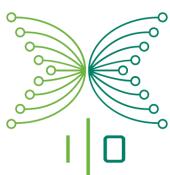
Defines the postgres connection, localhost in this case.

```
db: {  
  connectionString: 'postgres://backendapp_user:XXXXXX@localhost/backendapp',  
},
```

Logger

The application has a very detailed logging level, here you can define that level and define the output style, *prettyStdOut* in this case.

```
// Logger Streams
```



```
logStreams: [ {  
    level: 'info',  
    stream: prettyStdOut  
} ],
```

Bitgo

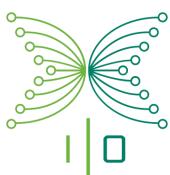
Bitgo is the Bitgo Blockchain API provider used, this configuration will normally be private, as accessToken allows handling the Wallets and transactions operations.

```
bitgo: {  
    useProduction: false,  
    accessToken:  
    '*****',  
    previous: 'http://*****/bitgotest/index.php',  
    webhook : 'http://localhost',  
    env: 'test'  
},
```

Sales App

In this section you should define the sales app api url and each of the endpoints used to communicate with.

```
const SALES_APP_ENDPOINT = 'http://localhost:8880/api/v1'  
...  
salesapp: {  
    endpoint: SALES_APP_ENDPOINT,  
    healthCheckEndpoint: SALES_APP_ENDPOINT + '/' + 'healthCheckEndpoint',  
    holdingWalletNotificationEndpoint: SALES_APP_ENDPOINT + '/' +  
    'holdingWalletReceivedBtc',  
    invoiceFundsReceivedNotificationEndpoint: SALES_APP_ENDPOINT + '/' +  
    'invoiceAddressReceivedBtc',
```



```
commissionReadyEndpoint: SALES_APP_ENDPOINT + '/' +
'commissionsMovedToWallet',
commissionPayoutDeliveredEndpoint: SALES_APP_ENDPOINT + '/' +
'commissionsPaid'
},
```

Backup Public Key

Every Wallet in the app is a MultiHD Wallet that requires 2 out of its 3 signatures to operate. One signature is kept by bitgo, and the other ones are stored encrypted in the DB to operate. But the third signature, the backup one, it's not needed to operate and it's store separately in a safe place. The Public Key is needed while creating new Wallets.

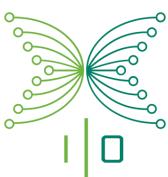
```
defaultBackupPub:
'*****',
*****'
```

Wallets and Addressed

Here you can configure the Commission and Holdings Wallets Ids, as well as the exodus Address and percentage and the *minConfirmations* required to consider a Transaction confirmed.

```
exodusAddress: '*****',
exodusPercentage: 0.8,
holdingWalletId: '*****',
commissionWalletId: '*****',
holdingFeeAddresses: [ '*****',
'*****'],
customRefundAddress: '*****',
minConfirmations: 1,
```

Secret



All sensitive data in the DB is stored encrypted by a combination of this secret, code, and environment as well as a salt on each entry.

```
secret: '*****',
```

AWS Configuration

Logs are currently stored using AWS Cloud Watch services. here you can define the corresponding credentials.

```
AWS_accessKeyId: 'AWS_accessKeyId',  
AWS_secretAccessKey: 'AWS_secretAccessKey',
```

Queue

Many processes are handled by an asynchronous queue. Here you can define the iteration time (in milliseconds) it runs on:

```
queuePollTime: 1000 * 60 // 1 min
```

New Relic

New Relic is used as a monitoring tool for the backend, it's very useful for detecting any possible downtime, network and application errors and making an analysis of application requests.

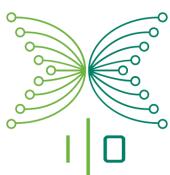
```
newRelicLicenseKey: '*****',
```

Routes Access Permissions

There are three level of permissions to access any endpoint of the backend API:

- Public: open to the network
- Read only (R): Requires at least a Read or ReadWrite Token as authorization header.
- Read & Write (RW): Requires a ReadWrite Token as authorization header.

Each route is mapped to an authorization requirements in the *src/conf/routeAccess.js* file:



```
module.exports.routes = [
  /** PUBLIC */
  { key : 'GET //healthcheck', access : '' },
  { key : 'POST //bitgoCallback', access : '' },
  /** PRICES */
  { key : 'PUT //price/runService', access : 'RW' },
  { key : 'GET //price/makeRequest/*', access : 'RW' },
  { key : 'GET //price/*', access : 'R' },
  /** SALESAPP */
  { key : 'GET //wallets/*', access : 'R' },
  { key : 'GET //invoiceWallets/*', access : 'R' },
  { key : 'POST //wallets.*', access : 'RW' },
  { key : 'PUT //wallets.*', access : 'RW' },
  { key : 'POST //invoiceWallets.*', access : 'RW' },
  { key : 'PUT //invoiceWallets.*', access : 'RW' },
  { key : 'PUT //holdingWallet/newAddress', access : 'RW' },
  /** MANAGEMENT */
  { key : 'PUT //fixHoldingWhitepolicies', access : 'RW' },
  { key : 'PUT //cleanCommissionWalletWhitelist', access : 'R' },
  { key : 'GET //notifications/*', access : 'R' }
]
```

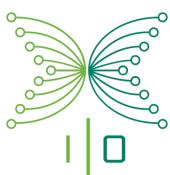
Each key represents an endpoint and the HTTP protocol {PUT, GET, POST} matching and the access code { "", "R", "RW" } defines the access level. The "*" wildcard matches any character; the array is processed in sequence, so it will take the first match for each endpoint.

Note: Refer to “**IOHK Backend - SalesApp Authentication**” document to further details on the authorization process.

Deployment

Deployment process is as simple as starting up a NodeJs application. In order to do so the following steps are needed:

1. Get source code to be deployed



2. Run `npm install`
3. Export the following environment variables:
 - a. `NODE_ENV={environment}` where environment is { staging, production }
 - b. `SECRET={secret}` see next section (*Encrypting Private Settings*)
4. Run `node app.js`

In order to improve the deployment process, it's suggested to use Docker as a container service. The following steps are required:

1. `export RELEASE_VERSION={release_version}` where `release_version` is a tag, for example `1.0.0`, `1.1.3`, etc
2. `docker build inputoutput/backend-app:$RELEASE_VERSION`
3. `docker stop backend-app`
4. `docker run -d -e "NODE_ENV=${environment}" -e "SECRET={secret}" --restart=on-failure:10 -p 8888:8888 --name backend-app inputoutput/backend-app:$RELEASE_VERSION`

The following Dockerfile can be used to achieve the steps stated above:

```
FROM node:4.2.3

RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

COPY package.json /usr/src/app/
COPY . /usr/src/app

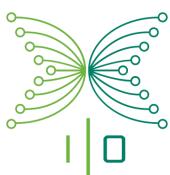
RUN npm install && npm install -g pm2 && pm2 install pm2-logrotate && pm2 set pm2-logrotate:max_size 500M

EXPOSE 8888

CMD ["pm2", "start", "app.js", "--no-daemon"]
```

Encrypting Private Settings

As mentioned in the Settings section, any sensitive key should be uploaded in an encrypted manner. The encryption key is sliced in two, the first part should go to a "`SECRET`" environment variable, and the second part fixed in the code. This process can be better explained by the line in the settings itself:



```
const decrypted = encryptionService.decrypt( process.env.SECRET + '*****',
, data )
```

Even if the encryption process is standard following the *sjcl*¹ one, there are two grunt² scripts to help the user in charge of deploy.

```
grunt encrypt_env_config --env environment
grunt decrypt_env_config --env environment
```

Example for staging.json:

Given this staging.json file:

```
→ src git:(master) ✘ grunt decrypt_env_config --env staging
Running "prompt:secret" (prompt) task
? Please enter decrypt/encrypt secret *****
Running "decrypt_env_config_task" task

Done, without errors.
→ src git:(master) ✘ cat conf/environment/staging.json
{
  "bitgo": {
    "useProduction": false,
    "env": "test",
    "enterpriseId": "████████████████████████████████████████",
    "accessToken": "████████████████████████████████████████████████████████",
    "webhook": "https://████████████████████████████████████████/api/v1/bitgoCallback"
  },
  "secret": "████████████████████████████████████████",
  "AWS_accessKeyId": "████████████████████████████████████████",
  "AWS_secretAccessKey": "████████████████████████████████████████████████████████",
  "customRefundAddress": "████████████████████████████████████████████████████████"
}
```

We can run the encryption process (using both key parts):

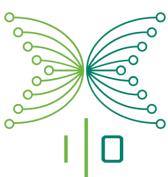
```
→ src git:(master) ✘ grunt encrypt_env_config --env staging
Running "prompt:secret" (prompt) task
? Please enter decrypt/encrypt secret *****
Running "encrypt_env_config_task" task

Done, without errors.
```

The file now contains an encryption of the previous:

¹ <https://crypto.stanford.edu/sjcl/>

² <https://gruntjs.com/>



```
+ src git:(master) x cat conf/environment/staging.json
{"iv": "",
 "v": 1,
 "iter": 10000,
 "ks": 256,
 "ts": 64,
 "mode": "ccm",
 "adata": "",
 "cipher": "aes",
 "salt": "",
 "ct": ""}
```

And if we run the decrypt script, we get the same initial file:

```
→ src git:(master) x grunt decrypt_env_config --env staging
Running "prompt:secret" (prompt) task
? Please enter decrypt/encrypt secret ****
Running "decrypt_env_config_task" task

Done, without errors.
→ src git:(master) x cat conf/environment/staging.json
{
  "bitgo": {
    "useProduction": false,
    "env": "test",
    "enterpriseId": "████████████████████████████████████████",
    "accessToken": "████████████████████████████████████████",
    "webhook": "https://████████████████████████████████/api/v1/bitgoCallback"
  },
  "secret": "████████████████████████████████████████",
  "AWS_accessKeyId": "████████████████████████████████████████",
  "AWS_secretAccessKey": "████████████████████████████████████████",
  "customRefundAddress": "████████████████████████████████████████"
}
```

Access Token generation

As mentioned in the “Routes Access permissions” in the System Configuration section, each endpoint can have access restriction. This permission is granted by a specific Token that needs to be previously generated and delivered to the allow third parties, the sales app should have RW access for example.

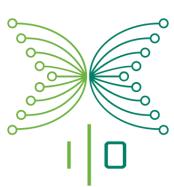
To generate Read (R) and ReadWrite (RW) access tokens, the user will need a grunt script, complete secret key, and have write access to the database to insert the new Token. As the token is salted and encrypted, there is no risk on data manipulation on a DB level.

For further details on Token generation please refer to the “*Backend - SalesApp Authentication*” Document, section “*Access Token Generation*”.

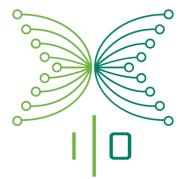
Recommendations

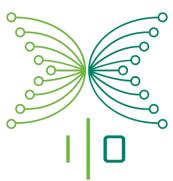
For deploying Backend in an easy and practical way, we suggest using Docker Containers.

Also, don't forget to set a fixed version in the nvm packages to prevent any possible errors inserted by a package update.

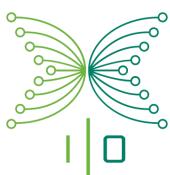


Backend - Authentication





Introduction	3
Applies for	3
Authentication	3
Requests	3
Responses	4
Revoke Access	4
Security Concerns	4
Access Token Generation	4
Example:	5
Backend Configuration	5
Sales-App Configuration	6



Introduction

This document explains the authentication process between the sales app and the backend, in every request and response. As well as the process needed to generate the tokens for that interaction.

Applies for

- Backend: Version > 1.0.0
- Sales App: Release > 1.15.0

Authentication

Requests

The authentication process from a sales app instance to the Back-end is made following the **Bearer Token Protocol** defined in [RFC6750](#)¹. In our case, tokens are statically generated, no generation endpoint is available.

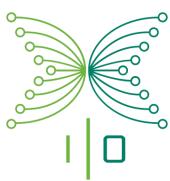
Essentially, every sensitive request will need the Header:

Authorization	Bearer {TOKEN}
---------------	----------------

There are 3 types of endpoint access:

- **Public**: No token is required (example: /healthcheck or bitgo callback)
- **Private ReadOnly**: This endpoints will required a **Read Only Token** or **Read-Write Token**: Returns error if not defined or incorrect. (example: GET -> /price/rates)
- **Private ReadWrite**: Only **Read-Write Token**, will work. (example POST -> /invoiceWallet)

¹ <https://tools.ietf.org/html/rfc6750>



Responses

Sometimes, it's the backend the one that initiates the interaction with the sales app, and in particular cases there is sensitive information sent back in the payload response. In these cases, the payload needs to be encrypted with a special key (SIGNATURE) associated with that client TOKEN, and include the client ID in the Header under the Header Key : “**X-Client-Id**”.

The encryption will need to follow the standard defined for the **Stanford Javascript Crypto Library (SJCL)**.

Revoke Access

To revoke access to a particular token, it's as simple as deleting the entry from the “access_token” table.

Security Concerns

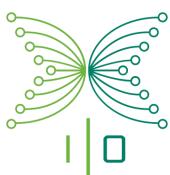
If the database gets compromised by enabling someone read-write permissions, they won't be able to compromise any access, only a deny of service is possible. Detail:

- They won't be able to read a token, as only a hash is stored
- Any new “fake” entry entered won't work as token, even if the same hash function was applied to it, as the payload is encrypted with the server deploy key. Even if they copy another registry encrypted data, that won't work because it's salted.
- Altering an existing entry would invalidate it, for the same reasons.

Access Token Generation

The Access Token is composed of the following elements:

- **Token:** The TOKEN itself **[Sensitive]**
- **Token Hash:** A salted hash of the token, this needs to be stored in the Database
- **Client ID:** A descriptive unique id of the token client. (example: “Sales-App-JPN”, “Atix-Read-Only”)
- **Token Access Type:** { R=ReadOnly, RW: ReadWrite }
- **Access Signature:** The signature associated with this Token, needs to encrypt payload responses **[Sensitive]**



- **Access Payload:** Encrypted Salted Payload Data, containing the Access Type and the Signature

The Token generation process can be performed by a *grunt script* ("generate_token") created for this purpose. It will ask for the application secret², the client-id and the access type, and will generate a completely new Access Token item. Which will then be needed to be entered into the 'access_token' table in the corresponding database *{token hash, client-id and access payload}*.

IMPORTANT: *The data will be returned by the script as text console output, so it's extremely important to run this script on a secured environment, the token & Access Signature must be kept safe after the process.*

Example:

```
→ src git:(feature/token-login) grunt generate_token
Running "prompt:clientId" (prompt) task
? Please enter the public ClientId to identify this token: sales-app-demo

Running "prompt:access" (prompt) task
? Please select the access type for this token: Read & Write

Running "prompt:secret" (prompt) task
? Please enter decrypt/encrypt secret ****

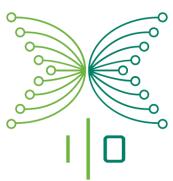
Running "generate_token_task" task
BitGo SDK env not set - defaulting to testnet at test.bitgo.com.
Your TOKEN data is:
{ token: '',
  tokenHash: '',
  clientId: 'sales-app-demo',
  access: '{"iv":"Yk41QsAIVvuPbVp//77gKA==","v":1,"iter":10000,"ks":256,"ts":64,"mode":"ccm","ad":"",
  accessSignature: '' }'

Done, without errors.
```

Backend Configuration

- Insert the new Access Token data into the database table "access_data" *{token: "token_hash", client_id : "client_id", access: "access payload"}*

² The secret value asked as "Please enter encrypt/decrypt secret can be found in development.json encrypted file as "secret"

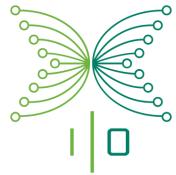


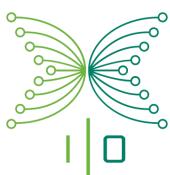
Sales-App Configuration

- Set up the Backend **TOKEN**
- Set up the Backend **ClientId**
- Set up the **Backend Access Signature**

These are currently plain text settings, but would need to be encrypted and turned unreachable from repository access.

Backend - Api Doc





Introduction	4
Distributor Named Invoice Wallet	4
Description	4
Diagram	4
Endpoint	5
Workflow	5
Get New Invoice Addresses	6
Description	6
Diagram	6
Endpoint	6
Workflow	6
Holding Wallet Funds Received	8
Description	8
Diagram	9
Endpoint	9
Sales App Notifications	10
Holding Wallet Received Bitcoins	10
Workflow	10
Bitcoin Received on Invoice Address	11
Description	11
Diagram	11
Endpoint	12
Sales App Notifications	12
Workflow	12
Commission Wallet Funds Received	14
Description	14
Diagram	14
Endpoint	15
Sales App Notifications	15
Commissions Moved To Wallet	15
Commissions Paid	15
Workflow	15
Fix Wrong Amount and Invoice Refunds	17
Description	17

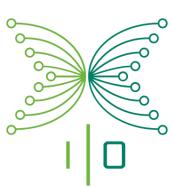
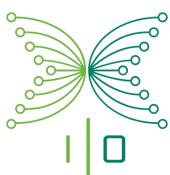


Diagram	17
Endpoint	18
Payload fields description and restrictions:	18
Workflow 1: Buyer Sends less than expected	18
Workflow 2: Buyer Sends more than expected, needs refund	19
Workflow 3: All funds needs to be refunded	19



Introduction

Backend provides a number of blockchain services to the Sales Application, in this document you'll find the documentation of the different endpoints that are needed to interact with it.

Every particular API endpoint has a description section, a flow diagram with involved actors, technical definitions of the endpoint itself and a sequence explanation of the different workflows it might follow.

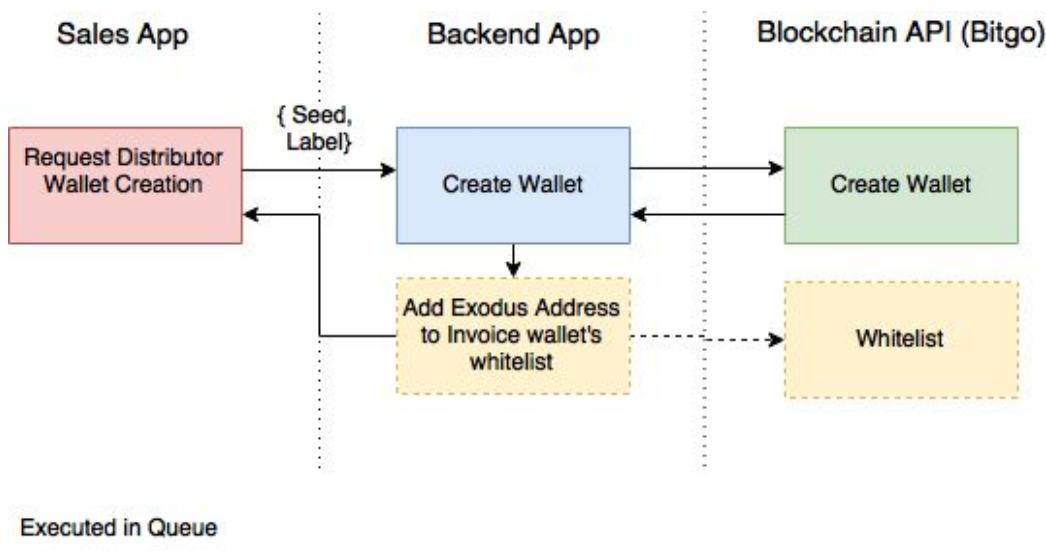
Distributor Named Invoice Wallet

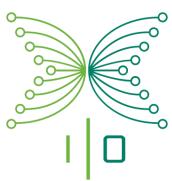
Description

Each distributor in the network will have a wallet seeded with account specific information and named after the user or user's email.

Making the seed (Sales-App):
sha256("UUID + 1st or original Email address")

Diagram





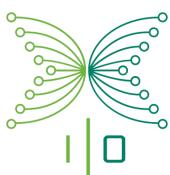
Endpoint

[**Security:** Requires RW Authorization Token]

```
POST /api/v1/invoiceWallets body
{
    "seed": [String],
    "walletName": [String](Optional)
}
→ { walletId : [String] }
```

Workflow

1. Sales App requests for a new Invoice Wallet
2. Backend App calls bitgo to create a wallet
3. Backend App queues a whitelist operation in order to add exodus wallet into new wallet's whitelist
4. Backend App returns:
 - a. 200 and the wallet id if everything is ok
 - b. 400 and the corresponding error message if there was a problem

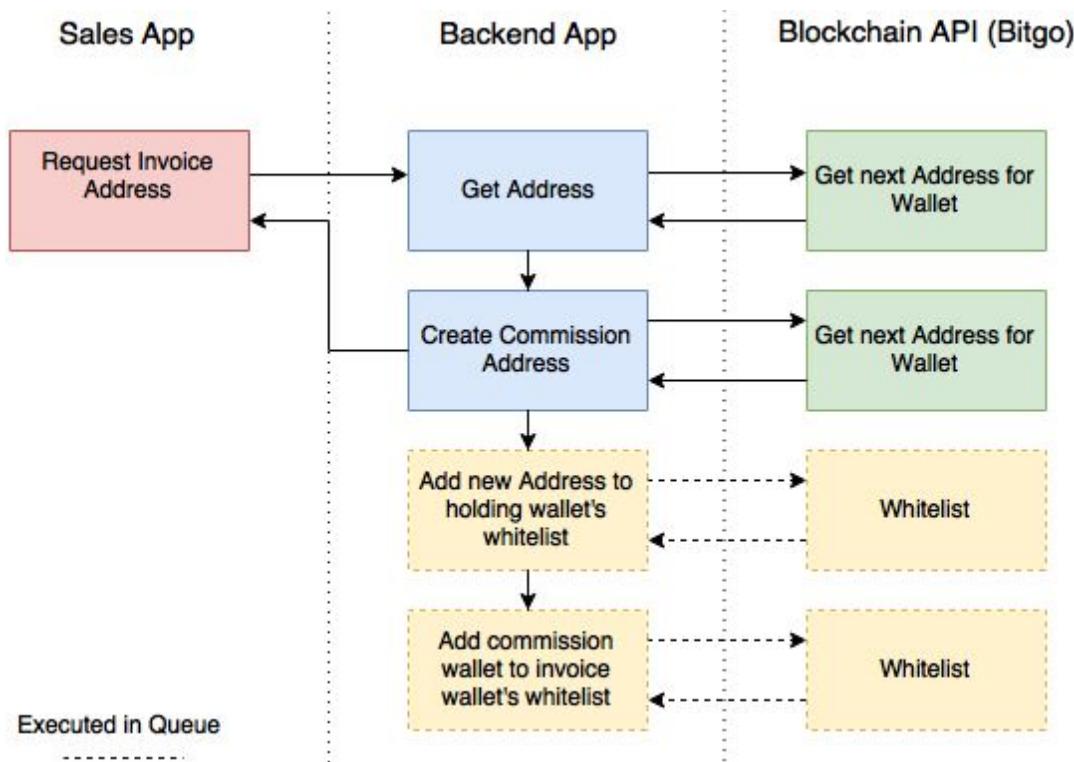


Get New Invoice Addresses

Description

Each order will request an invoice address from the buyer's distributor's wallet
The address will be attached to the order. Each address should be watched by the backend.

Diagram



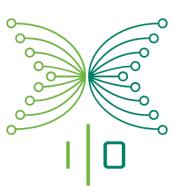
Endpoint

[**Security:** Requires RW Authorization Token]

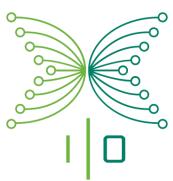
`PUT /api/v1/invoiceWallets/WALLET_ID/newAddress → { address : [String] }`

Workflow

1. Sales app requests for a new address given an invoice wallet Id
2. Backend App request for a new address



3. Backend App creates a new commission address and associates it with the new invoice address
4. Backend app queues an operation to add the new address to the holding wallet's whitelist, in order to be able to receive funds if this address is used in a bundle
5. Backend app queues an operation to add the commission address to the invoice wallet's whitelist, in order to be able to do commission payout



Holding Wallet Funds Received

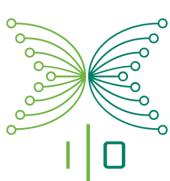
Description

Bank transactions are all bundled together in one single bitcoin Transaction that then gets split into individual invoices. This single transaction is handled in a particular Wallet, named Holding Wallet.

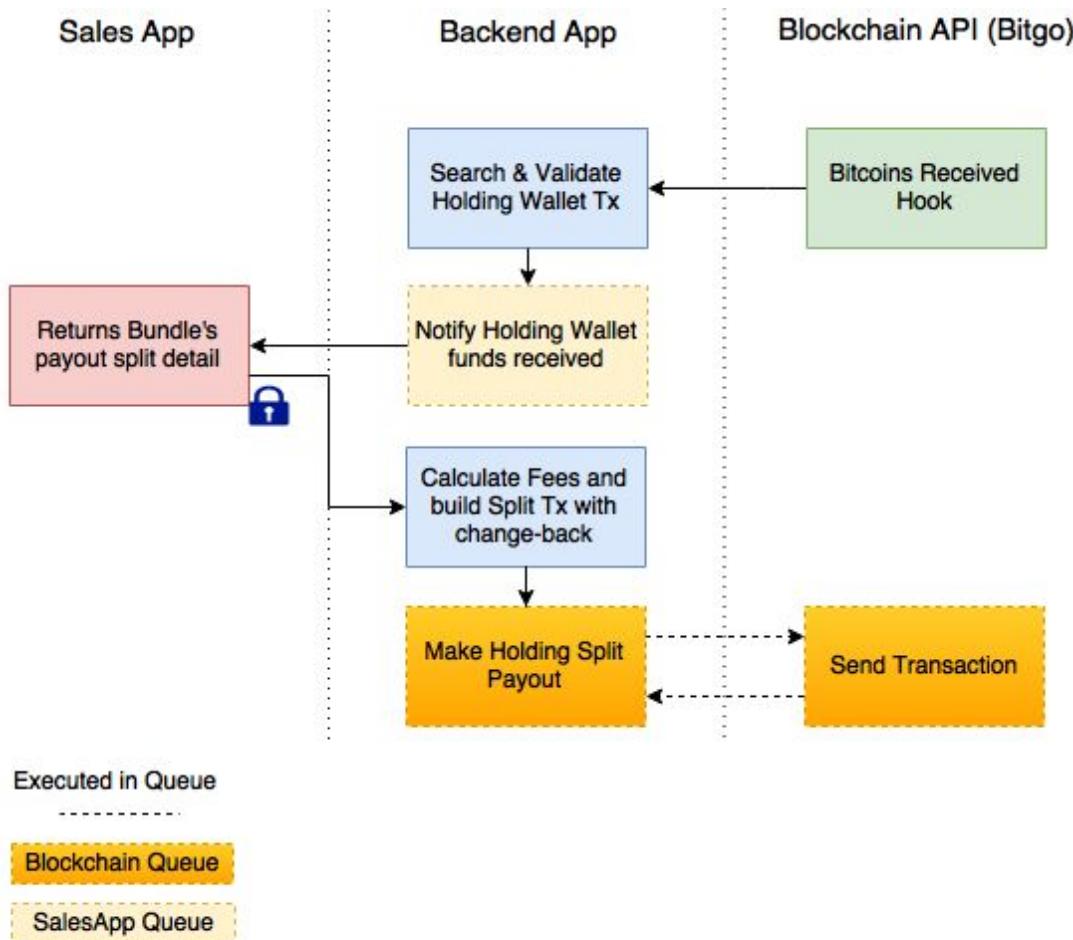
When a transaction is received for this Wallet, the backend will notify the sales app and expect a response containing all the invoices with each amount the funds should be split in. The sum should be the total of the amount received in the transaction.

The Fees for this transaction are taken from a specific *fee address* which is also used as *change-back* to keep funds available for following transactions. This design requirement restricts the amount of parallel transactions that can be done for this Wallet. **It's very risky and should be avoided at all terms** to start a new Holding Transaction if previous one has not yet been confirmed.

Note: All Invoice addresses are previously white-listed (on creating) for the Holding Wallet, this prevents the Split Tx to be complete if not every payout address is an invoice address.



Diagram

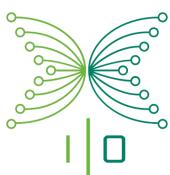


Endpoint

[**Security:** Public, doesn't need Authorization Token]

`POST /api/v1/bitgoCallback body { "hash": [String], "type": "transaction", walletId: [String] } → 200`

Condition: `walletId == HOLDING_WALLET_ID`



Sales App Notifications

1) Holding Wallet Received Bitcoins

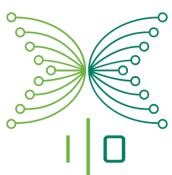
[**Security:** Requires Payload Signature]

```
POST /api/holdingWalletReceivedBtc body
{
    "holdingWalletAddress": "address",
    "satoshisReceived": "#ofsatoshisreceived",
    "transactionId": "transactionId"
}
→ 200 signed(*) body
{ "payout" : [ { invoiceAddress": "invoiceAddress", "amount": "amount"}, ... ] }
→ < 555 => (any error < 555 triggers retry)
→ > 555 => Stop Call
```

(*) Note: The payload body must be signed with the backend given key.

Workflow

1. Bitgo notifies backend app that a transaction has been detected for the Holding Wallet
2. Backend app checks if the transaction is confirmed, if not, finishes the execution
3. Backend queues the call to the Sales-App `holdingWalletReceivedBtc` with the transaction info.
4. Once the Notification gets executed, Sales-App responds 200 with a signed payload containing the invoices split payout, address and amounts for each one.
5. The Backend builds the transactions, calculates the fees and assigns the holding fee address as secondary input, and that same address as charge-back. (See [fees document](#) for details)
6. The Transaction is queued for it to be processed
7. Once the Worker processes the Tx and the network confirms it, each invoice notification will be received and it will start the flow described in [Bitcoin Received on Invoice Address](#)

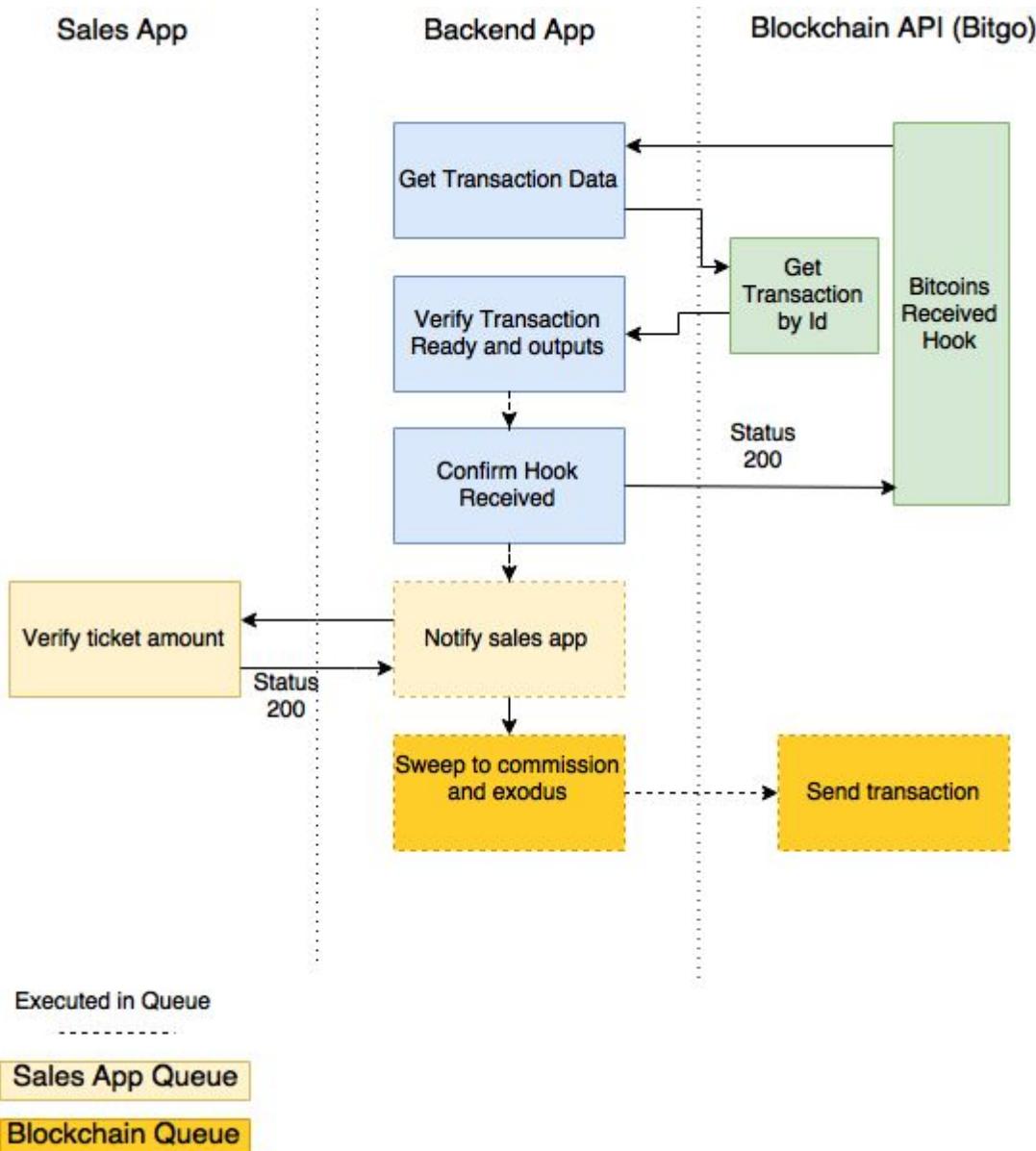


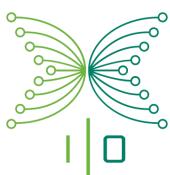
Bitcoin Received on Invoice Address

Description

When one of the requested invoice addresses receives funds, the sales app needs to be notified in order to split funds between exodus and commission wallets.

Diagram





Endpoint

[**Security:** Public, doesn't need Authorization Token]

```
POST /api/v1/bitgoCallback body
{
    "hash": [String],
    "type": "transaction",
    "walletId: [String]
} → 200
```

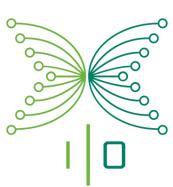
Sales App Notifications

[**Security:** Only response code needed]

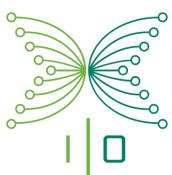
```
POST /api/invoiceAddressReceivedBtc body
{
    "invoiceAddress": [String],
    "satoshisReceived": [Integer],
    "transactionId": [String],
    "date" : [DateTime],
    "confirmations" : [Integer],
} → { 200 | 4xx | 5xx}
```

Workflow

1. Bitgo notifies backend app that a transaction has been detected
2. Backend app checks the transaction output and queues a job for each output corresponding to an invoice address
3. Backend App (Sales app queue) queries sales app in order to ask permission to split funds between exodus and commission
4. If Sales app returns:
 - a. Code 200 → go to step (6)
 - b. Code < 555 → funds are not split but this job will be retried (go back to step 4)
 - i. 400: Malformed request body, Invalid bitcoin address, invoiceWalletAddress, satoshisReceived must be a positive value
 - ii. 549: Amount received is invalid for ticket ##### having address XXXX
 - c. Code > 555 → funds are not split and this job is marked as failed and won't be executed again
 - i. 556: Ticket not found for address
 - ii. 586: Ticket has been canceled
 - iii. 587: Ticket has already received funds
 - iv. 589: Bitstamp prices not valid for ticket



5. Backend App (Sales App queue) queues an Invoice Split Job in Blockchain Queue
6. Backend App (Blockchain queue) splits 80-20% to exodus and commission

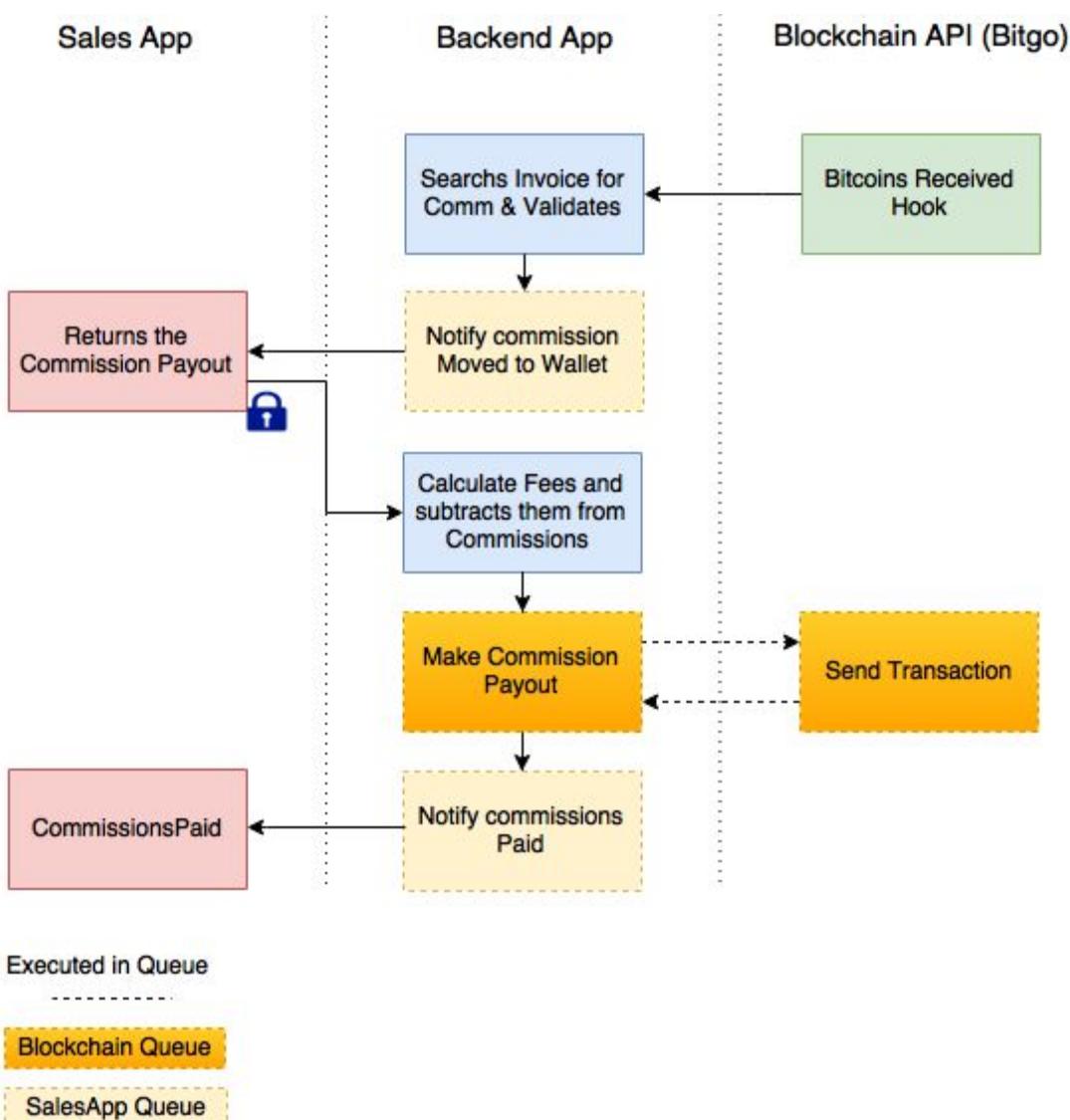


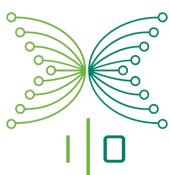
Commission Wallet Funds Received

Description

Once the exodus/commission split transaction is confirmed, the back-end app receives the hook and broadcasts it to the sales-app, It adds the invoice and commission address for this transaction info in the payload. The sales-app responds with the payout detail.

Diagram





Endpoint

[**Security:** Public, doesn't need Authorization Token]

```
POST /api/v1/bitgoCallback body { "hash": [String], "type": "transaction", walletId: [String] } → 200
```

Sales App Notifications

- 2) Commissions Moved To Wallet

[**Security:** Requires Payload Signature]

```
POST /api/commissionsMovedToWallet body
  {"invoiceAddress": "invoice-address",
   "commissionWalletAddress": "commission-address",
   "satoshisAmount": "satoshis-amount",
   "transactionId": "transactionId",
   "date": "tx-date"}
→ 200 signed(*) body
  {"commissions": [{"commissionAddress": "address", "amount": "#ofsatoshis"}]}
→ < 555 => (any error < 555 triggers retry)
→ > 555 => Stop Call
```

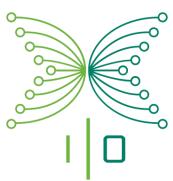
(*) Note: The payload body must be signed with the backend given key.

- 3) Commissions Paid

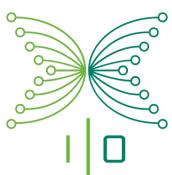
```
POST /api/commissionsPaid body
  {"invoiceAddress": "invoice-address", "transactionId": "transactionId"}
→ 200 => Success/Confirmation
→ < 555 => (any error < 555 triggers retry)
→ > 555 => Stop Call
```

Workflow

1. Bitgo notifies backend app that a transaction has been detected for the Commission Wallet
2. Backend app checks if the transaction is confirmed, if not, finishes the execution



3. Backend app checks the transaction output, as it's a Commision event, the address should be stored in the DB associated with the corresponding Invoice Address.
4. It queues the calls the Sales-App `commissionMovedToWallet` with the transaction info and the invoice address.
5. Once the Notification gets executed, the Sales-App responds 200 with a signed payload containing the commissions payouts, address and amounts to pay.
6. The Backend builds the transactions, calculates the fees and subtracts those (and the split ones) to make the amounts close-up to "0". (See [fees document](#) to details)
7. The Transaction is queued to be processed
8. Once the Worker processed it, the Sales-App Notification is queued to notify the `commissionsPaid` for this Transaction and Invoice Address.



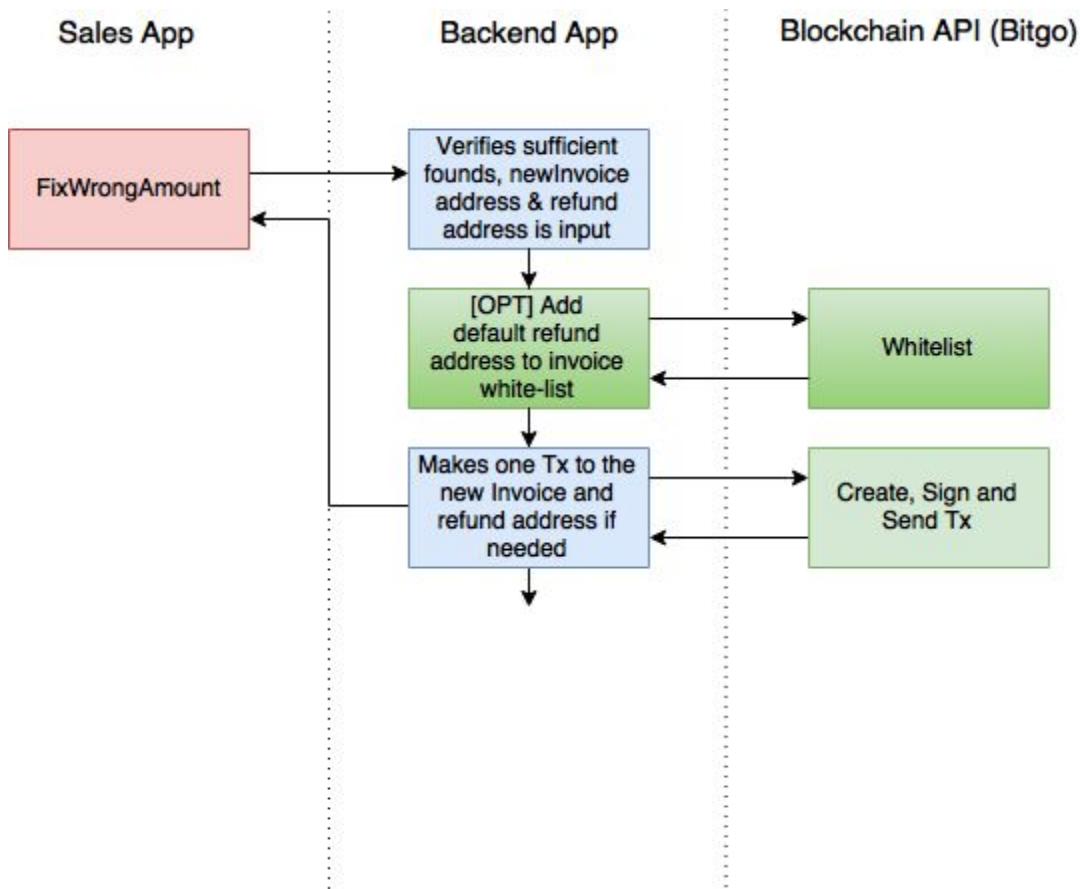
Fix Wrong Amount and Invoice Refunds

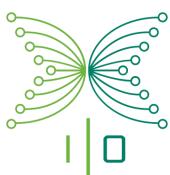
Description

This endpoint is intended to amend the case where the buyer sends a wrong Amount of bitcoins to one Invoice Address, this applies to:

1. Buyer sends less than expected and another Tx is made to complete the amount
2. Buyer sends more than expected and, if wanted, some funds needs to be refunded. This case includes the double sent scenario.
3. Buyer sends either less or more and the funds needs to be completely refunded.

Diagram





Endpoint

[**Security:** Requires RW Authorization Token]

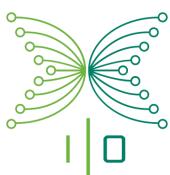
```
POST /api/v1/wallets/WALLET_ID/fixWrongAmount body
{
    "invoiceAddress": [String] required,
    "expectedAmount": [Integer] required,
    "newInvoiceAddress": [String] optional,
    "fee": [Integer] optional,
    "refundAddress": [String] optional
} → 200
```

Payload fields description and restrictions:

- **Invoice Address:** The invoices where the wrong amount was sent to.
- **Expected Amount:** The total amount of satoshis that was originally expected for this order. Can be zero if all funds would be refunded.
- **New Invoice Address:** A new Invoice address generated for the same Wallet, this address should also be updated in the Sales-app as the new Order Invoice Address. In the case 'expectedAmount' is zero, meaning that everything will be refunded, this parameter can not be included. But is required if expected amount is bigger than zero.
- **Fee:** The fee can be optionally supplied in the payload to be payed in the Tx, this is intended to allow a zero balance for this wallet once the funds are moved. If not included, the fees will be assigned by the backend, but there are no guarantees that current funds will be enough to cover them or in the other hand some coins be left in the wallet.
- **Refund Address:** If included, the extra funds (Balance - ExpectedAmount - Fee) will be returned to this address. This address should be the same as the one configured in the backend app secret configs. This restriction blocks the possibility of anyone trying to send funds to a different place.

Workflow 1: Buyer Sends less than expected

1. Buyer sends less satoshis than expected (example 10000 expected, 8000 received) to the invoice A.
2. The transaction is confirmed, the back-end receives the hook and forwards it to the sales-app, which rejects the funds because of an incorrect amount ($8000 < 10000$).
3. The buyer is asked to send the extra missing funds plus an extra to cover the fees ($2000 + 10$)



4. The second transaction is completed, the hook is received and once again the sales-app rejects it because of an invalid amount ($2010 < 10000$)
5. A sales-app operator creates a new Invoice Address B for this same Distributor's Wallet (using backend create address [functionality](#)) and then updates the sales app ticket Invoice wallet.
6. The *fixWrongAmount* is called with the payload
`{ invoiceAddress: A, newInvoiceAddress: B, expectedAmount: 10000, fee: 10 }`
7. The Backend will validate this data and if everything is ok, it will make the Transaction from A to B for 10000 satoshis paying 10 fee, and leaving A with zero balance.

Workflow 2: Buyer Sends more than expected, needs refund

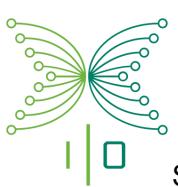
1. Buyer sends more satoshis than expected (example 10000 expected, 11000 received) to the invoice A, from the address C.
2. The transaction is confirmed, the back-end receives the hook and forwards it to the sales-app, which rejects the funds because of an incorrect amount ($11000 > 10000$).
3. A Sales-app operator creates a new Invoice Address B for this same Distributor's Wallet.
4. The *fixWrongAmount* is called with the payload:
`{ invoiceAddress: A, newInvoiceAddress: B, expectedAmount: 10000, refundAddress: SPECIAL_ADDRESS }`
5. The Backend will validate this data and if everything is ok, calculate the fees (let's say 8) and it will make the Transaction from A to B for 10000, 992 to *SPECIAL_ADDRESS* paying 8 fee, leaving A with zero balance.

Workflow 3: All funds needs to be refunded

1. Buyer sends 10000 to the invoice A from the address C; but for any reason, the funds needs to be refunded.
2. The transaction is confirmed, the back-end receives the hook and forwards it to the sales-app, which rejects the funds because of an incorrect amount or other reason.
3. The *fixWrongAmount* is called with the payload:
`{ invoiceAddress: A, expectedAmount: 0, refundAddress: SPECIAL_ADDRESS }`
4. The Backend will validate this data and if everything is ok, calculate the fees (let's say 7) and it will make the Transaction from A to *SPECIAL_ADDRESS* for 9993 paying 7 fee, and again leaving A with zero balance.

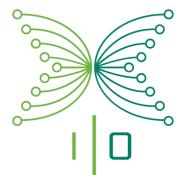
Notes:

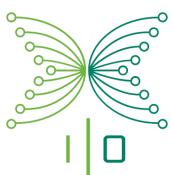
- If the refund address is not provided, the funds will be left in the InvoiceAddress
- If funds are not enough to pay fees, the Tx will be rejected



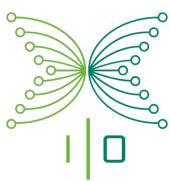
Sales App | Backend - Api Doc | 01/28/2016 | V - 0.11 | app vs 0.1.X

Backend - Api Troubleshooting





What is this document?	3
General considerations	3
Funds are stuck in an address	4
Scenario 1: Bitgo wallet funds received notification wasn't called	4
Cause	4
How to fix	4
Scenario 2: Sales app rejected funds movement	5
Cause	5
How to fix	5
Scenario 3: Funds cannot be moved because a whitelisting denial	6
Cause	6
How to fix	6
Bitgo API token max spend exhausted	7
Cause	7
How to fix	7
Exodus address needs to be changed	8
Considerations	8
Steps to Proceed	8

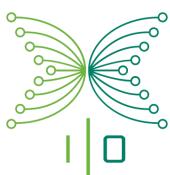


What is this document?

In this document are specific backend problems that might happen and the troubleshooting steps in order to fix them.

General considerations

- Backend App's goal is to minimize the amount of errors that require manual intervention. Some fixes presented in this document might be deprecated.
- Always try to look for information in the logs first.
- Every request has a unique id. It's supposed to be used as a filter in logs. Even workers that run enqueued, will keep track of the request_id that originally triggered them.
- Manually triggering funds movements is safe because funds are taken from specific input addresses linked to singular operations; for example, paying one order distributor's commissions can only be funded by that same order payment, even if the Distributor Wallet has funds from other invoices, those funds won't be involved in this Transaction. This maintains accountability very simple as every payment can be linked with each Transaction and address.



Funds are stuck in an address

Scenario 1: Bitgo wallet funds received notification wasn't called

Cause

We have found that sometimes Bitgo doesn't call us when a wallet address has received funds. We are not sure why this happens but we have already reported this to them. As the endpoint and the Tx are public, it's very simple to make the request ourselves.

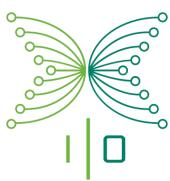
How to fix

1. Search in the logs if the notification was received. To do so use *Hook received \${TX_HASH}* expression. The transaction hash can be searched using any blockchain explorer¹. For example: Hook received
dbbe58dde6e583a13c1dc78274af2ebcd1c81d59c07fd78669a2ccf21bbb415c
2. Based on the search result
 - a. If there are no results check if you are searching in a date range containing the transaction date and go back to (1). If it's ok, the notification wasn't received and continue to (3)
 - b. If there is only one result, it would probably be the notification without any confirmations. In order to be sure, search for the req_id and you should see *Transaction is unconfirmed*. In this case the notification wasn't received and continue to (3)
 - c. If you see one or more notifications and there is at least one that doesn't have *Transaction is unconfirmed* message, this is probably not the error stated in this scenario
3. Using an http client² you need to create a post to APP_ENDPOINT/api/v1/bitgoCallback (don't forget to set the header Content-Type: application/json) with the following body:

```
{  
    "hash": "${TX_ID}",  
    "type": "transaction",  
    "walletId": "${WALLET_ID}"  
}
```

¹ <http://blockchain.info/>, <https://live.blockcypher.com/>

² <https://www.getpostman.com/>



For example,

```
{  
  "hash": "*****",  
  "type": "transaction",  
  "walletId": "*****"  
}
```

4. Once done you will receive *STATUS 200* and funds will be moved. You will be able to see the corresponding transaction in a blockchain explorer.

Scenario 2: Sales app rejected funds movement

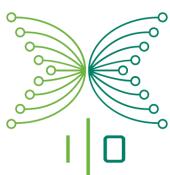
Cause

If the amount received does not correspond to the one expected by the sales app (based on ticket amount) the transaction won't be placed.

How to fix

1. Search in the logs if the error actually happened. To do so use *Hook received \${TX_HASH}* expression. The transaction hash can be search using any blockchain explorer³. For example: *Hook received ******
2. Searching using the *req_id* found in (1) you should be able to see "*Amount received is invalid for ticket*"
3. Update Sales App db in order to allow fund movement
4. Trigger the notification (as described in Scenario 1 - Step 3). Note: *This will be done automatically once sales app allows this error to be retried.*

³ <http://blockchain.info/>, <https://live.blockcypher.com/>



Scenario 3: Funds cannot be moved because a whitelisting denial

Cause

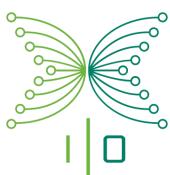
When Trying to move funds bitgo rejects the transaction because one of the destination addresses is not in the whitelist. This seems to be a bitgo error when calling add to whitelist simultaneously.

How to fix

1. Search in the logs if the error actually happened. To do so use *Hook received \${TX_HASH}* expression. The transaction hash can be search using any blockchain explorer⁴. For example: *Hook received ******
2. You should be able to see *Error: denied by policy*
3. Perform `GET APP_ENDPOINT/api/v1/invoiceWallets/${WALLET_ID}` and check if all the outputs are contained in `admin.policy.rules[0].condition.addresses`
4. Using IOHK custom Bitgo-cli⁵ whitelist the missing addresses
 - a. Login: `bin/bitgo login`
 - b. Select the wallet to add whitelist to: `bin/bitgo addWhitelist`
 - c. Input the address to whitelist `*****`
 - d. Add whitepolicy: `bin/bitgo *****`
 - e. You could see a 'Done!' message
5. Backend App will retry and send the funds

⁴ <http://blockchain.info/>, <https://live.blockcypher.com/>

⁵ <https://github.com/atixlabs/bitgo-cli>



Bitgo API token max spend exhausted

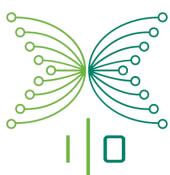
Cause

For each API Token Bitgo allows a certain amount of total BTC to be sent.

How to fix

1. Search in the logs if the error actually happened. To do so use *Hook received \${TX_HASH}* expression. The transaction hash can be search using any blockchain explorer⁶. For example: *Hook received ******
2. You should be able to see *Error: needs unlock*
3. There are 2 possibilities
 - a. Contact Bitgo and ask them to increase the limit
 - b. Issue a new token
 - i. Decrypt production.json file
 - ii. Change the Bitgo token with the new one
 - iii. Encrypt production.json file
 - iv. Push changes
 - v. Release and deploy a new version

⁶ <http://blockchain.info/>, <https://live.blockcypher.com/>



Exodus address needs to be changed

Considerations

When the exodus address is changed, all existing white-policies from invoices Wallets need to be updated as well.

Depending on the amount, we could ask Bitgo people to make a massive update, or run a script ourselves. Last time (Oct 2016), we had ~6000 wallets and they ask us to run the script.

Steps to Proceed

1. Make sure there are no current transactions and sales are stopped
2. Update [production.js](#) settings with the new exodus in **master** branch
3. Deploy the new version to production (Build with *Jenkins* and deploy with *Rundeck*)
4. Export all Production Wallets Ids to a text file. (*)
 - a. Connect to **Backend** Production DB
 - b. execute: `SELECT DISTINCT walletid FROM keys`
 - c. Export the result to a text file (each line should be a wallet Id, nothing else).
5. Run Script [fix_exodus_whitelist.sh](#) (the script is in `./scripts/` folder, not `./src/scripts/`)
 - a. Param \$1: File Path to Wallet Ids list
 - b. Param \$2: File Path to output activity log (will be created if not exist)
 - c. Param \$3: Environment to run, Options = {'STAGING', 'PRODUCTION'}
 - d. The script will ask for the **Production RW TOKEN**

Example: `bash ./fix_exodus_whitelist.sh /tmp/walletIds.txt /tmp/output.txt STAGING``

```
→ scripts git:(master) ✘ bash ./fix_exodus_whitelist.sh /tmp/walletIds.txt /tmp/output.txt STAGING
Please enter the RW Token? █
```

6. **Validate Whitelistings:** the script hits a Backend endpoint that enqueues all whitelisting requests, so the script finishing is not an indicator that we are ready to go. You should also check the whitelisting queue evolution. That can be checked in the DB, whitelisting table.

(*) Actually this can also be found in Sales-App DB, and probably these wallets can be sorted in a way that the first ones are the Most Recently Used. If run in that order, they would be processed first and can be ready for use without waiting the whole script and the whitelisting process to finish. This could be particularly beneficial if the exodus address switching needs to be done on a live sales scenario, where MRU Wallets are more prone to receive new orders first.