z

Abstract

The Dapps.ai Ethereum Blockchain Application Management Suite enables Salesforce customers to build, deploy, and manage decentralized applications on the Salesforce platform. This managed package is a composite application built to help companies quickly develop blockchain applications that can be integrated with existing customer data, business processes and other mission critical revenue driving applications.

Dapps.ai Release Notes

Ethereum Blockchain Application Management Suite v1.0

A picture containing clipart

Description generated with very high confidence

# Table of Contents

## Table of Contents ………………………………………………………………………………………………………. 1

## Dapps.ai Background …………………………………………………………………………………………………. 2

## Dapps.ai Technical Architecture …………………………………………………………………………………. 3

## Dapps.ai Salesforce Object Model ………………………………………………………………………… 4- 5

## Blockchain Technology ……………………………………………………………………………………………… 6

## Ethereum Virtual Machine ……………………………………………………………………………………… 7-9

## Package Install …………………………………………………………………………………………………….10 -12

## Dapps Dashboard …………………………………………………………………………………………………… 13

## Smart Contract Interface ……………………………………………………………………………………… 14-15

## Token Transfer Interface …………………………………………………………………………………… 16 -17

## Key Management Interface ……………………………………………………………………………………… 18

## Blockchain Analytics ……………………………………………………………………………………………… 19

# Dapps.ai Company Background

Dapps.ai is an award winning Blockchain Technology software company founded by Marc Wolenik and Dominic Steil on January 17th, 2017. Dapps.ai builds enterprise blockchain technology applications and blockchain enabled solutions. The company was built in Barcelona, Spain in 2017.

Dapps.ai builds blockchain enabled solutions for public, private, and permissioned blockchain protocols. These solutions are focused specifically on the Bitcoin, Ethereum and Hyperledger blockchains.

Dapps.ai flagship product, the Blockchain Application Management Suite, was released in May at Consensus 2017 in New York City. The Blockchain Application Management Suite enables customers to build, deploy and manage blockchain applications on the Salesforce platform.

Dapps.ai is a member of the Enterprise Ethereum Alliance and a Registered ISV on Salesforce.

Dapps.ai has offices in both San Francisco, CA and Barcelona, Spain.

A screenshot of a cell phone

Description generated with high confidence

Figure 1: Dapps.ai Homepage

# Blockchain Application Management Suite

The Dapps.ai Blockchain Application Management enables organizations to deploy and manage blockchain technology applications on the Salesforce platform. It provides a Solidity Compiler and testing environment for developers to write, compile, and save their smart contracts in a Smart Contract Library. The Smart Contract interface is used to dynamically generate interfaces based on the parameters of the smart contract, reducing development time and testing of contracts deployed to test nets. The Token Transfer interface enables companies to track the creation of ERC20 compliant tokens that are used for loyalty points for retail, airlines, and other digital assets. The interface can also be used for internal onboarding in addition to running reports and dashboards on point distributions and token transfers. Lastly all Smart Contracts deployed from the platform are filtered and watched on our nodes so if the contract is called outside of the platform it will write the transaction data back to the application suite.

# Dapps.ai Technical Architecture

Dapps.ai’s BAM provides software for companies to build and manage blockchain applications. We want enable companies to quickly prototype the use case and not worry about the cost of infrastructure. To achieve this our product addresses the application layer and the infrastructure layer. This includes the following:

**Application Layer:**

* Installed Managed Package
* Salesforce Lightning GUI (Angular and Salesforce Lightning Design System)
* Solidity Compiler
* Smart Contract ABI to HTML Interface
* Token Transfer Interface
* Transaction Signing Module
* Reports and Dashboards
* Key Management Interface
* Account Creating

**Infrastructure Layer:**

* Node.js NodeServer
* Ethereum GETH Node (Blockchain Node)
* Virtual Machine on Heroku, AWS, Azure, or IBM Bluemix
* HSM integration

A screenshot of a social media post

Description generated with very high confidence

Figure 2: Dapps.ai Technical Architecture Diagram

This is a set of tools that allows a Salesforce developer or admin to quickly build and manage blockchain applications. Dapps.ai handles the infrastructure level requirements of running an Etheruem Node, Key Management Policies, and encryption and decryption of transactional data. Dapps.ai provides the applications layer graphical user interfaces built with the Salesforce Lightning Design System, creating an experience that seamlessly integrates with the rest of the Lightning platform. To goal is to enable Salesforce developers, administrators and users to be able to build, manage and use blockchain applications that integrate with their existing customer data and apps that have built on the platform.

# Dapps.ai Salesforce Object Model

The Blockchain Application Management Suite is a composite application built on the Salesforce platform. The objects used are the following:

|  |
| --- |
| dapps\_\_Address\_\_c |
| dapps\_\_Smart\_Contract\_\_c |
| dapps\_\_Token\_\_c |
| dapps\_\_Token\_Transfer\_\_c |
| dapps\_\_Library\_\_c |
| dapps\_\_Transaction\_\_c |
|  |

A screenshot of a social media post

Description generated with very high confidence

Figure 3: Dapps.ai Salesforce Object Model

# A map of a computer Description generated with high confidence

Figure 4: Dapps.ai Schema

# Blockchain Technology

Blockchain technology is a system that enables distributed consensus protocols; verifiable, tamperproof, state machines that can be used to create decentralized application networks. The first decentralized application network was created for a global digital cash and transactional value layer, Bitcoin.

The Bitcoin protocol facilitates the transfer of digital cryptocurrency payments by using a (Unspent Transaction Output) UTXO digital bearer architecture that is globally accessible and driven by proof of work and the largest deployment of public/private cryptography in human history. It is open source game theory driven, censorship resistant, cryptoeconomic incentivized technology system that is operated by users, developers and miners.

Other decentralized application networks have been created to facilities use cases other then digital value and payments. One of these protocols is the Ethereum Blockchain. Ethereum is a world computer, a generalized computational singleton that enables applications to be built and deployed to update the state of this one global virtual machine. Rent is extracted to update and run your application on this machine in the form or Ether, a gas to the platform that fuels computational resource extensive operations. Ethereum has enabled developers and users to create tokens on this global machine and these tokens can be offered in crowd sales known as in an initial coin offering. Private implementations of the Ethereum Blockchain can be used and there are many forked or modified version of the protocol as well.

A screenshot of a cell phone

Description generated with very high confidence

Figure 5: Blockchahin Technology diagram; blocks hashed referring to the previous block header

Distributed Ledgers that have similar consensus properties to the Bitcoin and Ethereum Blockchain ledgers have been created with more of a focus on commercial and enterprise infrastructure level applications. One of these protocols is the Hyperledger Fabric Blockchain infrastructure. Hyperledger is a permissioned blockchain ledger that has a module consensus model and focuses on validating and ordering transactions between multiple parties.

Respectively, Bitcoin is a known as a public permission less blockchain. Ethereum is a public blockchain and has private blockchain implementations as well and Hyperledger is a permissioned Blockchain Distributed Ledger. There are other blockchain technology protocols that are focused on commercial and enterprise applications.

Our Focus for this flagship product release is on the Ethereum Blockchain. The applications developed will point to a private Ethereum test net while in the development and poc stage. When ready to switch to a production deployment on the public Ethereum network, the infrastructure layer is directed a Dapps.ai production node.

# Ethereum Virtual Machine

**A gl**o**bal cryptographic, decentralized, immutable, permissionless world computer.**

And in relation to the protocols of that which it is built on.

**Internet = Communications**

**Bitcoin = Money**

**Ethereum = Computation**

**The Ethereum Computer**

The EVM is stack-based execution environment that uses Smart Contracts (similar to object-oriented classes) and HTML, CSS, and JavaScript to create dapps. When you are running a decentralized application (dApp), every instruction is executed on every node of the network. This means the operands for the instructions are taken from the stack, and it is also where the results are added. This is the low level assembly code of the EVM and the resulting flow-control functions that can be found in the [Ethereum Yellow Paper](http://gavwood.com/paper.pdf). Items are pushed to the Stack and can be manipulated using POP (remove top item from the stack), SWAP (Swap item order on the stack / Limit of 16 elements), and DUP (copy and order new item on top of the stack).

**Memory and Calldata**

The stack also extends Memory and Calldata to be used in during program execution. Memory is an expandable byte-array used to store data during program execution. It can be accessed using the MSTORE and MLOAD instructions. Calldata is a byte-array, just like memory, but it is read-only. It contains the data from the **transaction** that triggered the code execution in the Contract.

**Storage**

Storage is a map used for fields in contracts. A contract can neither write nor read any storage other that its own. Essentially it is permanently storing the state variables within the contract.

**Elements of the Ethereum Smart Contract**

Contracts in Solidity are similar to classes in object-oriented languages. Each contract can contain declarations of:

* State Variables
* Functions
* Function Modifiers
* Events
* Structs Types
* Enum Types

with Parameters of:

* The gas-price I want to pay (gasPrice).
* The maximum amount of gas that may be spent (gas).
* The amount of ether I want to transfer (value).
* My **account** address (from).
* The target **account** address (to).
* The nonce (nonce).
* The **transaction** data (data).

A smart contract’s code resides in a Contract Account. It is unalterable once deployed.

**Accounts in Ethereum**

There are two kinds of accounts on the Ethereum network: Externally Owned Accounts (Public-Private Key) and Contract Accounts.

**Externally Owned Account (EOAs):**an account controlled by a private key, and if you own the private key associated with the EOA you have the ability to send ether and messages from it.

* Can have an Ether balance.
* Can send transactions.
* Are controlled by private keys.
* Has no code.

**Contract Accounts** (CA): an account that has its own code, and is controlled by code.

* Can have an Ether balance.
* Can send transactions.
* Can send messages.
* Contracts are controlled by their contract code.
* Only send transactions in response to other transactions that they have received. Therefore, all action on the Ethereum blockchain is set in motion by transactions fired from Externally Owned Accounts.
* Every time a contract account receives a transaction its code activates, allowing it to read and write to internal storage and send other transactions/messages or create contracts.

EOA0 —-**Transaction**–> CA1 = Activate Code in CA1

CA1 —– **Messages**—> CA2, CA3, CA4= Perform functions in CA2, CA3, CA4

EOA1 —–**Transaction** —-> EOA2 = Send Ether EOA2

**Ether**

Ether, the currency used in Ethereum, is exchanged for computation on the platform. Gas is the name for the execution fee for every operation made on an Ethereum blockchain. Its price is expressed in ether and it’s decided by the miners, which can refuse to process transaction with less than a certain gas price. To get gas you simply need to add ether to your account. The Ethereum client automatically converts Ether to gas and gas to Ether when transactions are processed.

When a **transaction** is sent from an EOA to a CA, it is sent with input code and Ether. The input code and ether is then processed by the Contract Account, thereby activating itself and thus executes its code. A CA can then send a **message** out to other (CA) contracts. The contracts are essentially talking and passing messages between themselves, in this way, sending a message is exactly like calling a function. This could produce another smart contract, send Ether to another address, register a vote of confidence, open a door to a house; virtually any sort of call to or update to the state of the decentralized network.

The total cost of a **transaction** is based on 2 factors:

* gasUsed is the total gas that is consumed by the transaction
* gasPrice price (in ether) of one unit of gas specified in the transaction

**Total cost = gasUsed \* gasPrice**

An EOA can also send a **transaction** to another EOA and nothing happens accept transfer some Ether (P2P payment or ether).

# Configuration and Package Installation

**Configure Remote Site Settings**

Add <https://eth.dapps.network> to the Remote Site Settings

Once you have added this URL to the Remote Site Settings

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

Click **View in another browser** and Accept for the Connected App Access. You will now be redirected to the page below.

A screenshot of a cell phone

Description generated with very high confidence

Now you are on the Package Installation Screen.

Click **Configure** or go to the **Dapps Dashboard Page**

# Dapps Dashboard

Here you will be able to create a new account and set a password for the account. Once you have created your new account you can return to the home screen, remember the password you used. When you sign transactions from the Smart Contract Interface you will be able to query for the Accounts that have been created in your org.

A screenshot of a cell phone

Description generated with very high confidence

Figure 6: Dapps.ai Dapps Dashboard page; Create accounts and set passwords for use when making state changes

After you have successfully created the account you will see the green banner below:

A screenshot of a social media post

Description generated with very high confidence

Figure 7 Dapps.ai Account Setup

# Solidity Compiler

Solidity is the programming language that is used to write Ethereum Smart Contracts. Solidity is a statically typed contract oriented language that compiles into Ethereum Virtual Machine (EVM) Bytecode. When the contract is compiled using out Solidity Compiler GUI; the smart contract code, EVM bytecode and contract ABI is saved in the Smart Contract Library. Smart Contracts that have already been written locally can be uploaded into the GUI to be edited, compiled and saved. The Smart Contracts that are saved in the Library well be queried and deployed by users in the Smart Contract Interface.

Click on the Solidity Compiler Tab from the Dapps.ai Home Screen.

A screenshot of a computer

Description generated with very high confidence

Figure 8: Dapps.ai HomeScreen in the Salesforce Lightning Expirience

Import your Smart Contract from your Local Machine

A screenshot of a cell phone

Description generated with very high confidence

Figure 9 Dapps.ai Solidity Compiler; Write and Compile Smart Contracts

Once you have compiled your Smart Contract you can save it in the Smart Contract Library

A screenshot of a social media post

Description generated with very high confidence

Figure 10: Dapps.ai Solidity Compiler; Save you Smart Contract to the Library in Salesforce

The smart contract solidity code, the bytecode, and the abi will be save the Smart Contract Library.

# Smart Contract Interface

The Dapps.ai Smart Contract Interface allows users to select and use Smart Contracts from the Smart Contract Library. Based on the parameters of the Smart Contract defined in the ABI, an interface will be dynamically generated for the user. The interface allows a user to enter data into input fields and deploy a Smart Contract to the blockchain. When deploying the Smart Contract the user will be able to sign the transaction with an account password or the private key for the address. A module will prompt the user for the input to sign and create the blockchain transaction.

To select a Smart Contract from the Library, enter the name of the Smart Contract in the Search Box:

A screenshot of a social media post

Description generated with very high confidence

Figure 11: Dapps.ai Smart Contract Deployer; Dynamic ABI to HTML interface based on Smart Contract parameters

When you select the Smart Contract the input parameters are dynamically generated.

A screenshot of a cell phone

Description generated with very high confidence

Figure 12: Enter in an existing Smart Contract Address to retrieve the latest state and its functions

If you have already created and deployed a Smart Contract you can past in the contract Address to pull up the current state and functions of that Smart Contract.

# Transaction Signer

When you want to deploy a transaction to the blockchain you need to sign a message using an Ethereum Account. Dapps.ai provides a Transaction Signer module that will enable you to pick which account you would like to deploy from. You can sign the message using the Ethereum Password you set on the Account Management Screen or using the Private Key for the Account.

A screenshot of a computer

Description generated with very high confidence

Figure 13: Dapps.ai Transaction Signer

Make sure to Select your account and use the Ethereum Password Input with the password you set in the Account Page.

# Token Transfer Interface

The Dapps.ai Token Transfer Interface allows users to select and transfer ERC20 Tokens that have been created via a deployed Smart Contract. An ERC20 Token is a digital asset that is created via a Smart Contract on the Ethereum Blockchain. The token can be sent to other Ethereum Addresses that are on the Ethereum Blockchain. All of the data on the Token Transfer is stored in Salesforce, this is data such as the amount of the token transferred, the age of the transfer and the transaction hash of the transfer. This enables visibility into the token transfer events that are happening via that Smart Contract.

A screenshot of a cell phone

Description generated with high confidence

Select which Token you would like to transfer. You can then paste the recipients address and the amount of the token you would like to transfer.

# Dapps.ai Blockchain Analytics

Blockchain Technology offers powerful time based, unalterable anyltics that can be queried and used with Salesforce’s native reporting and dashboard capabilities, in addition to Salesforce Einstein Discovery. Dapps.ai manages all of the Addresses, Smart Contracts, Transactions, Tokens, and Token Transfers; reports and Dashboards can be generated for all of the data that is stored in these objects.

A screenshot of a cell phone

Description generated with very high confidence