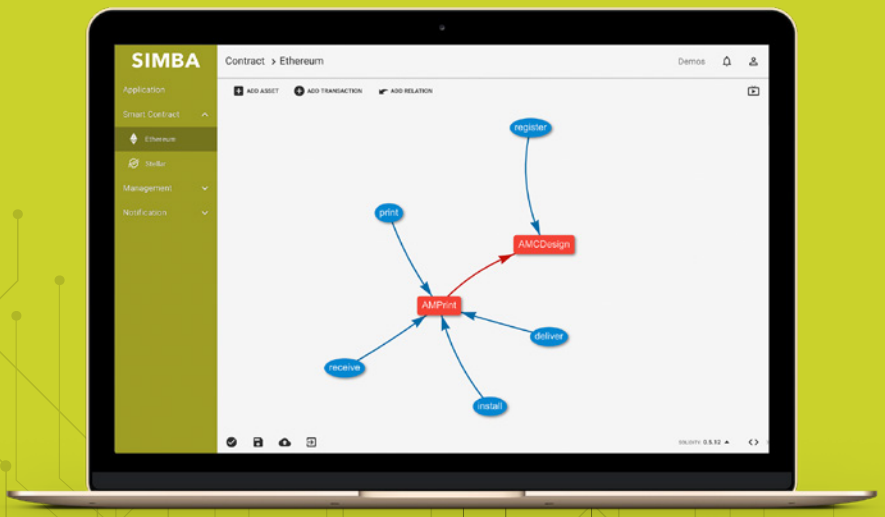
SIMBA

WHAT IS SIMBA?



SIMBA Chain brings Blockchain to anyone with an idea. We make it easy to build and deploy Blockchain systems across multiple Blockchains and data stores.

SIMBA CHAIN PROVIDES A SIMPLIFICATION LAYER TO DEVELOP APPLICATIONS ON A BLOCKCHAIN

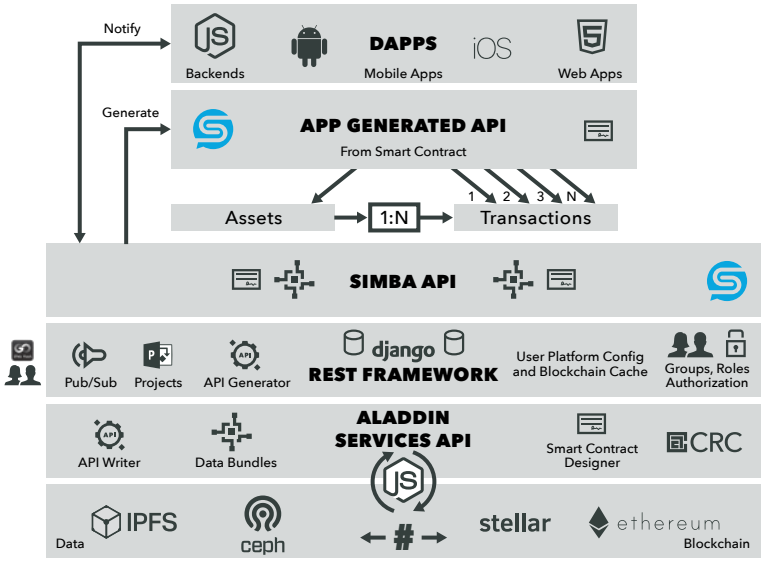
In order to understand the value of SIMBA Chain, let's define a Blockchain. A Blockchain is a distributed ledger in which an immutable, and therefore non-repudiable record of transactions or events can be stored permanently and verifiably, without the need for a central authority. Blockchains allow digital information to be distributed, but not copied, meaning each individual transaction can only have one owner. This is enabled through the use of wallets, containing (multiple) special asymmetric cryptosystem keys that are used to sign each transaction, making all transactions identifiable and authentic.

Blockchains that employ the use of Smart contracts (programmable pieces of code) can ensure certain transactions are only performed when specified conditions are met without the need for a human to manually intervene – smart contracts have been described as "cryptographic 'boxes' that contain value and only unlock if certain conditions are met.". Smart contracts are also deployed and written as a transaction onto the Blockchain, which also makes them immutable and signed by the user that deployed them to ensure the authenticity of the code is not compromised.

From an application standpoint however, this underlying tooling is not enough to build production applications.

SIMBA Chain addresses these needed features and more, the architecture of the system is shown to the right. It has a core underlying goal of providing tooling for developers to make it easy to build and deploy Blockchain systems across multiple Blockchains and data stores. It provides a design tool for generating a business process model for tracking assets, which auto-generates a smart contract and corresponding API for integration with external applications. The resulting API provides REST based access to smart contract methods that transact on the ledger and provides access control using groups for reading or writing to such resources. All data smart contract API endpoints allow multiple data files to be attached and stored in an off-chain data store, such as IPFS or Ceph. The diagram below expands on this to illustrates the core features of the system for each level of the development Blockchain stack.

SIMBA ARCHITECTURE



	BLOCKCHAIN	SIMBA
ACCESS CONTROL	Private – External Tools | Public – None	Permissioned Data and Blockchain in API
WALLET	Key Sign Transactions	Fully Integrated Wallet (Browsers APIs)
DATA STORE	External Data, Hashcode On-chain	Automated Off-chain Storage
SMART CONTRACT	Non Repudiable Code and Automation	Write Smart Contracts with Zero Code
BLOCKCHAIN	Non Repudiable Transactions	App Interface to Multiple Blockchains

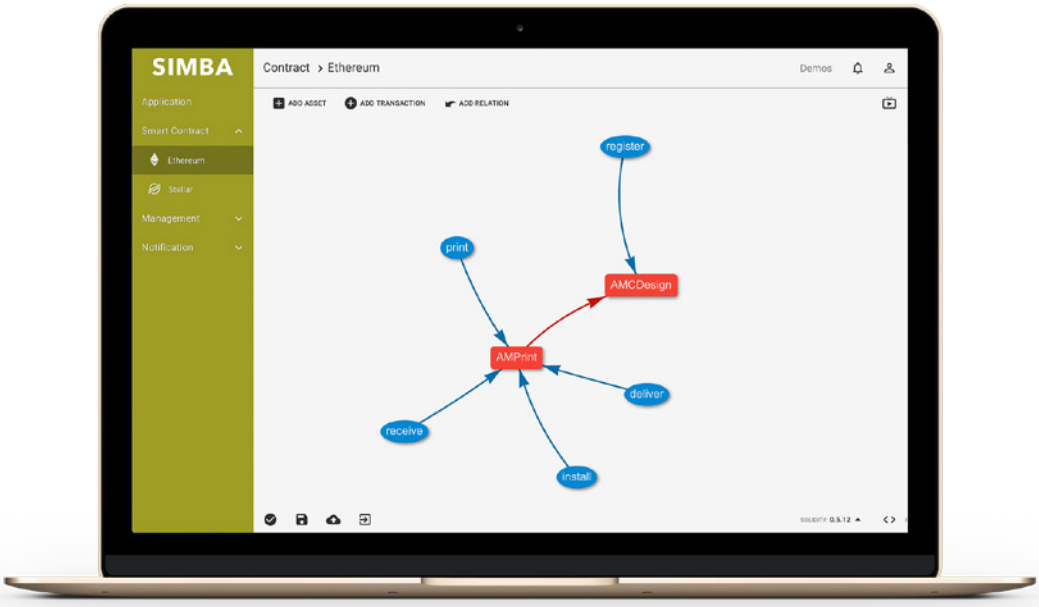
BLOCKCHAIN

For the Blockchain layer, SIMBA provides a generic API to multiple Blockchain systems, thus the system does not have a dependency on a single Blockchain or DLT implementation. Currently, the platform supports Ethereum and Stellar but several more are on the roadmap. Furthermore, although SIMBA Chain is proprietary, it will soon release a code generation capability that auto-generates source code that binds to the specific Blockchain system and Data Store the user has configured. This makes it possible for developers to export the application into an enterprise environment or to use SIMBA Chain to create open source APIs for public use.

SMART CONTRACTS

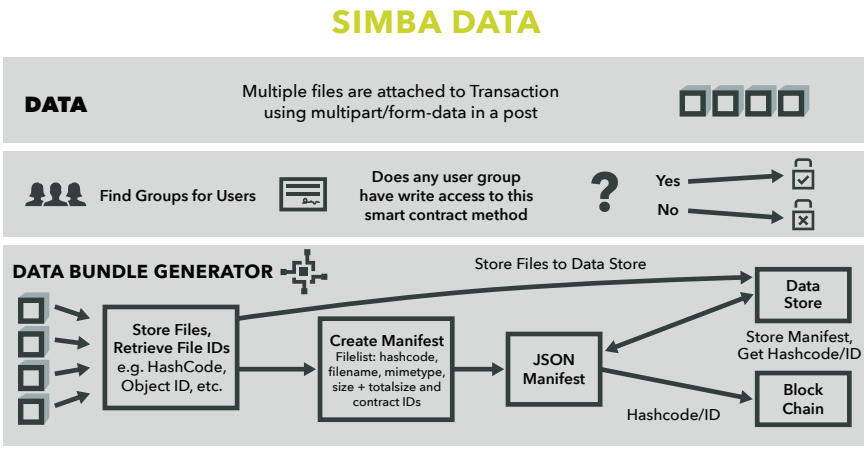
Smart Contracts provide the interface, and business logic, to what is written on the Blockchain and what rules need to be satisfied for this write operation to take place. SIMBA's Smart Contract Designer (SCD) allow anyone to quickly and easily create smart contracts. The SCD generates Smart Contracts are automatically from conceptual models that define the Assets and Transactions that transact on those Assets. Such models are specified using SIMBA's Web App's UI shown below.

A user simply uses the SCD to add Asset or Transactions along with their methods and parameters, and SIMBA Chain automatically generates the smart contract for the platform they select (in this case, Solidity code for Ethereum). It also generates a graph of the relationships for the model as shown. The resulting smart contract, once deployed on the Blockchain, is dynamically exposed as an application REST interface for simpler external application interaction with the Blockchain.



DATA STORES

Data Stores in SIMBA Chain uses the same adapter pattern as the Blockchain one, that is, a single generic REST interface can support the simple integration of different data stores. Currently SIMBA supports the Ceph, IPFS and flat file system based data stores. As shown, data flows into the system through the application's REST API (generated by SIMBA) by attaching one or multiple files to the transaction. This is achieved by using a simple multipart form post. Transactions are then checked for access (see below) before being passed to the data bundling mechanism, which stores all files into the Data Store and collects each hashcode into a JSON manifest file. The manifest is then stored into the Data Store and its hashcode is stored onto the Blockchain. Using this mechanism, the system can easily retrieve all files by first retrieving the manifest, and then using each hashcode to retrieve the files. The hash also serves as a digest to guarantee the integrity of the data.



WALLET

A private key of a Wallet is required to sign a transaction. In SIMBA Chain we do not store user's private keys, instead, we provide a callback mechanism that allows the developer's application to sign transactions on behalf of their users. When a transaction request is made on SIMBA using the application API, it sends it off to the Blockchain adapter to generate the transaction payload. This payload is then returned to the sender for signing.

Along with the transaction payload the headers include the external user ID so that user in the developer's external system can be extracted and their private key applied to the transaction, for signing. This signed transaction is returned to SIMBA for submission to the Blockchain. An example of this implementation is in the SIMBA Dashboard, which implements this functionality in the "Make Transactions" tab of the Application view, using an in-browser implementation of a digital wallet.

SIMBA Chain enables seamless utilization and integration of Blockchain technology to bolster trust, security, and risk mitigation for enterprise and government

simbachain.com
574-914-4446
info@simbachain.com

SIMBA