Eris Legal Markdown: Putting The Contracts in Smart Contracts

Introduction

The Eris Legal Markdown system is the backbone of the Distributed Application Software Stack's legal compliance strategy. When combined with EPM (https://epm.io) and the DeCerver (https://decerver.io), ELM provides an ability to link legal contractual agreements to smart contract architecture so that the legal contract's provisions can be incorporated into the smart contract mechanics.

To bridge the gap between the legal niceties required in order to enforce an actual contract in an actual court with the automation assistance which comes when using smart contracts to administer an agreement we have built a Go port of Legal Markdown and integrated it into the Decerver (https://decerver.io) portion of Eris' **Distributed Applications Software Stack**.

The process of ensuring that there is a real world legal contract overlay fused onto a specific smart contract which is built to administer a data-driven interaction we call **dual integration**. This process, which is explained below, is a multiple step process which is very simple to perform in the Decerver.

What is Dual Integration?

Dual integration is the process of integrating a specific legal contract (which can be built with Legal Markdown or any other contract building system) into a specific smart contract which runs on a distributed data store such as Eris Industries' Thelonious (https://thelonious.io). The idea of dual integration is to allow users to be able to have the certainty of having a real world contract which can be taken to a court and enforced using established dispute resolution processes in the jurisdiction(s) of the user(s) while also using a smart contract as the primary mechanism for administering the data-driven interaction which attends to the

agreement between the parties. For another system which is also seeking to bridge this gap see Primavera de Filippi's Draft Legal Framework For Crypto-Ledger Transactions (http://p2pfoundation.net/Legal_Framework_For_Crypto-Ledger_Transactions).

The reason Eris Industries recommends to all users of Distributed Technology, and particularly smart contracts, to dual-integrate their smart contracts with real world legal contracts built by lawyers qualified in the jurisdiction(s) that will be relevant to the agreement, is simple. Smart contracts are necessarily limited. As they are, at their core, just scripts which live in a distributed data store, the pure code of a smart contract has a limited ability to "reach" outside the context of their data store to incorporate a legally-binding contractual understanding. While they are capable of being structured in a manner which would automatically administer a data-driven interaction and ensure harmony of the data set in which the smart contracts reside (if they have permissions to do so), judges are unlikely, for the foreseeable future, to be able to easily resolve disputes stemming from smart contracts solely on the basis of their coded parameters (meaning without an integrated legal contract) without simply applying the commercial defaults for the agreement in the jurisdiction – an end that is unlikely to reflect the intention of the parties to the agreement in question. For these reasons of the limited reach of smart contracts and the limited enforceability of smart contracts, we **highly** encourage all smart contract systems developers to utilize dual integration of some kind.

Once the dual integration process is finished the result will be a smart contract which references a specific fingerprint of a real world contract (this is called the `hash` of a file, or sometimes it is called the `checksum` of a file) and a real world contract which integrates the specific fingerprint of a smart contract (this utilizes the `chainID` of the distributed data store which the smart contract runs on as well as the `contractAddress` of the specific smart contract). This circular reference ensures – **to a cryptographic certainty** – that the specific file which contains the real world contract and the specific smart contract both properly reference one another.

To perform a dual integration one would simply take the following steps.

1. Deploy a smart contract
2. Reference the `chainId` and `contractAddress` of the deployed smart contract in the final draft of the real world contract.
3. Finalize the real world contract and find its digital fingerprint.
4. Send a transaction logging the `checksum` of the real world contract into the storage of the smart contract.

That's it. If those steps are followed then there will be a cryptographically-certain dual integration.

ELM's Role in Dual Integration

Eris' Legal Markdown greatly assists in the dual integration process by providing an interface for lawyers to work with coders to develop automated transactional solutions for clients. In general, most transactional law follows a similar pattern to how templating works in software development. The idea generally is that a template is developed and then an "instance" of that template is deployed based on a set of "parameters". ELM embraces this idea by providing a very simple way to send a set of parameters to both an ELM template and a smart contract and dual integrate those.

Legal Markdown is one of the few legal templating systems which works from plain-text documents which can be **much more easily** dealt with programmatically than a document system like PDFs or DOCXs, but which also integrates prosaic language which is useful to (and understood by) most lawyers.

ELM's Design Philosophy

This library was built specifically to empower the creation of structured legal documents using simple text based documents rather than bloated word processor files. This library provides the user with access to structured headers, internal cross references, optional clauses, mixins, and other features that will greatly empower the use of text based documents to create and maintain structured legal documents from template files in simple text based documents.

This library will parse YAML and JSON Front Matter of Markdown and other Text Based Documents. **It will work with more than only markdown**. Despite its name, Legal Markdown is not actually dependent upon markdown if used as a preprocessor. If used as a full text processor it is linked to common markdown syntax.

Designed with Lawyers in Mind

Some may argue that lawyers are incapable of learning *any system* which is not a popular, closed-source word processor tool. At Eris Industries we do not accept that. Lawyers are craftshumans and documents are (predominantly) our craft. Word processors are fine tools, but they are tools which were not engineered specifically for lawyers and they are not very adaptable to different contexts.

Text editors, on the other hand, are more simple and purposefully built to be extendable and configurable, and are a joy to work with for text-based documents. Text editors can also be much more easily integrated with process and workflow

tools than a closed-source word processor that only has a limited API. While working from a text editor does require a cognitive load to learn, for lawyers who do make the plunge the payoff is a drastically improved working experience.

ELM, Eris Industries' implementation of Legal Markdown, not only assists with dual integration of smart contracts and real world contracts, but at its core it was built to simplify the workflow for any transactional documents. It has a range of features which are not only simple to use from a text editor but also will make transactional document creation and maintenance simple and fun.

Try ELM for yourself

Eris' Legal Markdown is **real**. We're not talking about vaporware or white papers here – we're talking about real software that developers can use, today. Head over to our tutorials page (https://lmd.io/tutorials/) and see what you can build!