(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0005090
(43) 공개일자 2025년01월09일

(51) 국제특허분류(Int. Cl.)
H04L 9/00 (2022.01) *G06F 9/451* (2018.01)
G06F 9/54 (2018.01)
(52) CPC특허분류
H04L 9/50 (2022.05)
G06F 9/451 (2018.02)
(21) 출원번호 10-2024-7033067
(22) 출원일자(국제) 2023년03월01일
심사청구일자 없음
(85) 번역문제출일자 2024년10월02일
(86) 국제출원번호 PCT/CA2023/050265
(87) 국제공개번호 WO 2023/164767
국제공개일자 2023년09월07일
(30) 우선권주장
17/653,085 2022년03월01일 미국(US)

(71) 출원인
케이엔엔엑스 코퍼레이션
캐나다 엠5에이치 3엠7 온타리오 토론토 유니버시티 애비뉴 181 스위트 1410
(72) 발명자
스리바스타바, 니라즈
캐나다 엠5제이 2에이치7 온타리오주 토론토 유니버시티 애비뉴 55 스위트 1200
(74) 대리인
양영준, 임규빈, 백만기

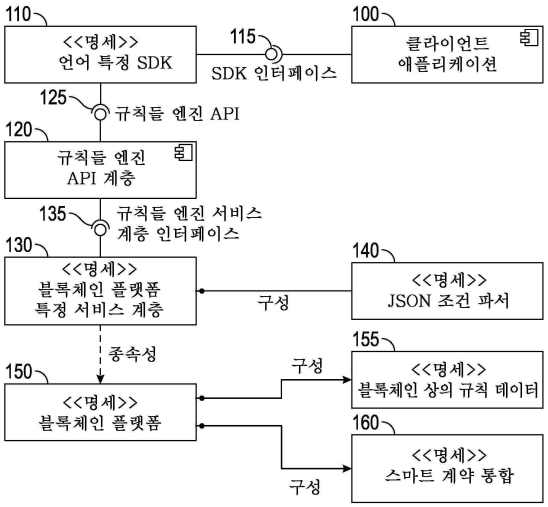
전체 청구항 수 : 총 25 항

(54) 발명의 명칭 **블록체인 규칙들 엔진**

(57) 요 약

블록체인 규칙들 엔진은 블록체인 및 비 블록체인 애플리케이션들에 의한 사용을 위해 규칙들, 거래들, 및 결과들을 블록체인 상에 저장하는 규칙들 엔진의 실제 구현을 묘사하는 플러그가능한 서비스로서 제공된다. 블록체인 규칙들 엔진은 블록체인을 관리하는 블록체인 플랫폼과 상호 작용하는 클라이언트 애플리케이션에 의해 구현되는 규칙들을 관리한다. 블록체인 규칙들 엔진은 클라이언트 애플리케이션에 인터페이스를 제공하고 클라이언트 애플리케이션에 규칙 실행 기능성을 노출시키는 언어 특정 소프트웨어 개발자 키트(SDK), 언어 특정 SDK에 규칙들 엔진 API를 제공하고 언어 특정 SDK로부터 커맨드들을 수신하여 프로세싱하는 규칙들 엔진 애플리케이션 프로그래밍 인터페이스(API) 계층, 및 블록체인 플랫폼과 상호 작용하고 규칙들 엔진 API 계층으로부터 입력을 받아 규칙들에서 구현되는 비즈니스 로직을 실행하는 블록체인 플랫폼 특정 서비스 계층을 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 9/541 (2013.01)

명 세 서

청구범위

청구항 1

블록체인을 관리하는 블록체인 플랫폼과 상호 작용하는 클라이언트 애플리케이션에 의해 구현되는 규칙들을 관리하는 블록체인 규칙들 엔진으로서,

상기 클라이언트 애플리케이션에 인터페이스를 제공하고 상기 클라이언트 애플리케이션에 규칙 실행 기능성을 노출시키는 언어 특정 소프트웨어 개발자 키트(SDK);

상기 언어 특정 SDK에 규칙들 엔진 API를 제공하고 상기 언어 특정 SDK로부터 커맨드들을 수신하여 프로세싱하는 규칙들 엔진 애플리케이션 프로그래밍 인터페이스(API) 계층; 및

상기 블록체인 플랫폼과 상호 작용하고 상기 규칙들 엔진 API 계층으로부터 입력을 받아 상기 규칙들에서 구현되는 비즈니스 로직을 실행하는 블록체인 플랫폼 특정 서비스 계층 - 데이터 동작들은 상기 블록체인 플랫폼 특정 서비스 계층 및 상기 블록체인 플랫폼에 의해 함께 관리됨 -

을 포함하는, 블록체인 규칙들 엔진.

청구항 2

제1항에 있어서,

상기 블록체인 플랫폼 특정 서비스 계층은 규칙이 적용되는 객체들에서 존재하는 조건들을 파싱하고 평가하는 자바스크립트 객체 표기법(JSON) 조건 파서를 포함하는, 블록체인 규칙들 엔진.

청구항 3

제2항에 있어서,

상기 블록체인 규칙들 엔진에 의해 지원되는 각각의 타입의 기능성의 각각의 인스턴스는 표준 블록체인 규칙들 엔진 포맷의 JSON 문서에 의해 표현되고, 각각의 JSON 문서는 다른 블록체인 거래들과 동일한 방법 및 합의를 사용하여 상기 블록체인에 기입되는, 블록체인 규칙들 엔진.

청구항 4

제3항에 있어서,

각각의 타입의 기능성에 대한 상태의 각각의 수정 또는 변경은 상기 수정된 기능성에 대한 상기 JSON 문서의 완전히 업데이트된 버전을 상기 블록체인에 첨부하는 것에 의해 표현되는, 블록체인 규칙들 엔진.

청구항 5

제1항에 있어서,

상기 언어 특정 SDK는 상기 언어 특정 SDK에 의해 지원되는 프로그래밍 언어로 개발되는 블록체인 애플리케이션에 의해 상기 클라이언트 애플리케이션이 규칙 실행 기능성에 액세스하는 것을 가능하게 하기 위한 인터페이스를 제공하는, 블록체인 규칙들 엔진.

청구항 6

제1항에 있어서,

상기 규칙들 엔진 API 계층에 의해 제공되는 상기 인터페이스는 상기 블록체인 규칙들 엔진과 인터페이싱하기 위해 다중 테넌트들 및 다중 애플리케이션들을 지원하는, 블록체인 규칙들 엔진.

청구항 7

제6항에 있어서,

상기 규칙들 엔진 API 계층은 상기 언어 특정 SDK로부터 또는 사용자 인터페이스 콘솔을 통해 상기 블록체인 규칙들 엔진의 규칙 신택스들 및 로직 커맨드들을 받아들이고, 상기 규칙들 엔진 API 계층은 상기 신택스들 및 로직 커맨드들을, 상기 블록체인 플랫폼 상에서 구현되는 상기 블록체인 플랫폼 특정 서비스 계층에 대한 인터페이스를 통해 상기 블록체인 플랫폼 특정 서비스 계층 상으로 전달될 수 있는 포맷으로 변환하는, 블록체인 규칙들 엔진.

청구항 8

제1항에 있어서,

상기 블록체인 플랫폼 특정 서비스 계층은, 상기 블록체인 플랫폼 특정 서비스 계층에서 또는 상기 블록체인 플랫폼 상에서 실현될 수 있는 기능성들 및 피처들의 서브세트를 포함하여, 상기 블록체인 규칙들 엔진의 모든 실현에서 지원되어야 하는 인터페이스 플러스 기능성들을 설명하는, 블록체인 규칙들 엔진.

청구항 9

제8항에 있어서,

상기 피처들은: 함수 생성, 함수 업데이팅, 함수 삭제, 규칙 생성, 규칙 업데이팅, 규칙 삭제, 규칙 활성화 및 비활성화, 규칙 가져오기 및 내보내기, 또는 규칙 롤백 중 적어도 하나를 포함하는, 블록체인 규칙들 엔진.

청구항 10

제1항에 있어서,

상기 블록체인 플랫폼 특정 표면 계층은 상기 블록체인 상에서의 프로세싱 결과들의 실행 및 저장을 위해 전처리기 함수들 및 후처리기 함수들을 파싱하는 조건 파서를 포함하는, 블록체인 규칙들 엔진.

청구항 11

제10항에 있어서,

상기 전처리기 및 후처리기 함수들은 규칙들과 링크되어 상기 전처리기 및 상기 후처리기 함수들의 거동을 변경하는, 블록체인 규칙들 엔진.

청구항 12

제10항에 있어서,

상기 후처리기 함수들은, 규칙이 실행되고 상기 실행된 규칙에 의해 통과 또는 실패 결과가 생성된 이후 수행되는 통과시 후처리기 함수 또는 실패시 후처리기 함수 중 적어도 하나를 포함하는, 블록체인 규칙들 엔진.

청구항 13

제1항에 있어서,

상기 블록체인 플랫폼 특정 서비스 계층은, 상기 블록체인 플랫폼 상에서 스마트 계약에 의해 사용될 규칙 관련 기능성들을 위해 상기 블록체인 플랫폼 상에 직접 구현되는 기능성을 제공하는, 블록체인 규칙들 엔진.

청구항 14

제1항에 있어서,

상이한 규칙 실행 기능성이 상이한 데이터 객체들에 적용되고, 상기 데이터 객체들은 테넌트, 애플리케이션, 규칙 그룹, 함수, 또는 규칙 중 적어도 하나를 포함하는, 블록체인 규칙들 엔진.

청구항 15

제14항에 있어서,

각각의 타입의 규칙 실행 기능성의 각각의 인스턴스는 각각의 타입의 규칙 실행 기능성에 공통인 포맷의 JSON

문서에 의해 표현되고, 각각의 JSON 문서는 다른 블록체인 거래들과 동일한 방법 및 합의를 사용하여 상기 블록체인에 기입되는, 블록체인 규칙들 엔진.

청구항 16

제15항에 있어서,

규칙 실행 기능성에 대한 상태의 각각의 수정 또는 변경은 상기 규칙 실행 기능성의 타입에 대한 JSON 문서의 완전히 업데이트된 버전을 상기 블록체인에 첨부하는 것에 의해 표현되는, 블록체인 규칙들 엔진.

청구항 17

제14항에 있어서,

상기 규칙 실행 기능성은 스마트 계약으로부터 직접 액세스 가능한, 블록체인 규칙들 엔진.

청구항 18

제1항에 있어서,

상기 클라이언트 애플리케이션에서 상기 규칙들 엔진 API를 활용하여 애플리케이션들에서 동적 규칙 생성, 동적 규칙 수정, 및 규칙들의 동적 삭제를 가능하게 하는 신택스, 인터페이스들 및 요건들을 약술하고(outline) 상기 블록체인 규칙들 엔진이 규칙들 데이터를 상기 블록체인 상에 저장하는 방법을 약술하는 블록체인 규칙들 엔진 명세를 더 포함하는, 블록체인 규칙들 엔진.

청구항 19

제18항에 있어서,

상기 블록체인 규칙들 엔진 명세는 상기 블록체인 규칙들 엔진이 상기 블록체인 상에서 규칙 실행 이전에 실행되는 명령어들의 세트를 저장하는 방법을 명시하는, 블록체인 규칙들 엔진.

청구항 20

제18항에 있어서,

상기 블록체인 규칙들 엔진 명세는 다수의 테넌트들이 규칙들을 생성 및 관리하는 것 및 자격증명 데이터 및 규칙 함수들을 상기 블록체인 상에 저장하는 것을 가능하게 하는 것에 의해 상기 블록체인 상에서 다중 테넌시를 구현하는, 블록체인 규칙들 엔진.

청구항 21

제20항에 있어서,

상기 블록체인 규칙들 엔진 명세는 다수의 레벨들의 액세스 제어를 통합하고, 그에 의해, 규칙에 대해 임의의 동작을 수행하기 위해, 상기 클라이언트 애플리케이션은 테넌트의 자격증명뿐만 아니라 상기 클라이언트 애플리케이션의 자격증명을 제공하는, 블록체인 규칙들 엔진.

청구항 22

제18항에 있어서,

상기 블록체인 규칙들 엔진 명세는 상기 규칙 실행 기능성을 활용하기 위해 상기 블록체인 규칙들 엔진에 의해 지원되는 애플리케이션 레벨 계층 구조를 제공하고, 그에 의해, 상기 클라이언트 애플리케이션은 생성된 규칙들 및 함수들을 사용할 수 있고 자격증명 및 메타데이터를 상기 블록체인 상에 저장할 수 있는, 블록체인 규칙들 엔진.

청구항 23

제18항에 있어서,

상기 블록체인 규칙들 엔진 명세는, 체인간 코드 통신을 확립하고 비 블록체인 애플리케이션들에 대한 연결을

확립하기 위해, 상기 블록체인 규칙들 엔진과의 블록체인 애플리케이션들의 통신을 위한 플러그인을 구현하는, 블록체인 규칙들 엔진.

청구항 24

제18항에 있어서,

상기 블록체인 규칙들 엔진 명세는, 규칙 실행을 수행하기 위해 상기 클라이언트 애플리케이션과 상기 블록체인 규칙들 엔진 사이의 연결을 생성하는 함수들을 제공하는 것에 의해, 생성된 규칙들을 실행하는 능력을 상기 클라이언트 애플리케이션에게 제공하는, 블록체인 규칙들 엔진.

청구항 25

제18항에 있어서,

상기 블록체인 규칙들 엔진 명세는, 클라이언트 애플리케이션 코드에 직접 이용 가능한 함수들을 정의하는 함수 인터페이스들의 세트, 및 상기 블록체인 규칙들 엔진 명세에 대한 호출들을 상기 규칙들 엔진 API에 대한 호출들로 변환하는 것에 의해 상기 함수 인터페이스들을 구현하는 함수 구현 코드를 포함하는, 블록체인 규칙들 엔진.

발명의 설명

기 술 분 야

[0001] 본 출원은 블록체인 상에서 비즈니스 프로세스들을 구현하기 위한 규칙들 엔진에 관한 것으로, 특히, 블록체인 및 비 블록체인 애플리케이션들에 의한 사용을 위한 플러그가능한 서비스(pluggable service)로서 구현되는 블록체인 규칙들 엔진에 관한 것이다.

배 경 기 술

[0002] 규칙들은 임의의 프로세스들 또는 시스템의 중요하고 의미 있는 역할을 한다. 규칙들은 비즈니스 프로세스들이 성공하는 기반이다. 규칙들은 많은 소스들 - 법적 규정 또는 회사 정책들 또는 프로세스들 및 사람들을 수반하는 임의의 조직 또는 그룹 - 을 가질 수 있다. 규칙들은 통상적으로 결과들을 결정하는 로직으로서 구현된다. 규칙들은 결정을 구현하기 위해 사용될 수 있거나, 또는 이벤트, 우선 순위 점수, 또는 다른 함수들에 기초한 계산일 수 있다.

[0003] 규칙들은 기술 시스템들 및 애플리케이션들이 처음 등장한 이래로 그들의 일부였다. 1세대 시스템들 및 애플리케이션들에서, 비즈니스 규칙들은 모든 결정들, 계산들 및 조건들이 실행되었던 시스템들 또는 애플리케이션 코드 베이스의 일부를 형성하였다. 규칙들이 복잡하게 되고 코드 베이스의 큰 부분을 형성함에 따라, 1990년대 초에 Pegasystems, Fair Isaac Corp, 및 ILOG로부터의 제품들에서 규칙들 엔진이 도입되었다. 규칙들 엔진은 정확한 의사 결정을 가능하게 하는 별개의 시스템 또는 애플리케이션이며 규제 및 조직 규칙들이 빈번하게 변경되어 로직 변경들을 필요로 하는 인스턴스들에서 복잡한 종속성들에 특히 유용하다. 별개의 규칙들 엔진은 새로운 규칙을 포함시키기 위한 또는 규칙들을 편집하거나 또는 비활성화하기 위한 용이한 방식을 제공한다.

[0004] 현존하는 규칙들 엔진 플랫폼들은 단지 규칙들 그대로인 것으로부터 관리 및 사용하기에 쉽고 빠른 체계적인 규칙들 엔진으로 진화하였다. 불행하게도, 현존하는 규칙들 엔진들은 블록체인의 패러다임에 맞지 않으며, 규칙들 및 규칙들로부터의 거래들은 단지 개발자의 의도대로만 블록체인에 고정될 수 있다.

발명의 내용

[0005] 본원에서 설명되는 시스템들 및 방법들은 블록체인 및 비 블록체인 애플리케이션들에 의한 사용을 위해 규칙들, 거래들, 및 결과들을 블록체인 상에 저장하는 규칙들 엔진의 실제 구현을 묘사하는 플러그가능한 서비스로서 블록체인 규칙들 엔진을 제공하는 것에 의해 기술 분야에서의 요구들을 해결한다. 블록체인 규칙들 엔진은 블록체인을 관리하는 블록체인 플랫폼과 상호 작용하는 클라이언트 애플리케이션에 의해 구현되는 규칙들을 관리한다. 블록체인 규칙들 엔진은 클라이언트 애플리케이션에 인터페이스를 제공하고 클라이언트 애플리케이션에 규칙 실행 기능성(rule execution functionality)을 노출시키는 언어 특정 소프트웨어 개발자 키트(software developer's kit)(SDK), 언어 특정 SDK에 규칙들 엔진 API를 제공하고 언어 특정 SDK로부터 커맨드들을 수신하

여 프로세싱하는 규칙들 엔진 애플리케이션 프로그래밍 인터페이스(application programming interface)(API) 계층, 및 블록체인 플랫폼과 상호 작용하고 규칙들 엔진 API 계층으로부터 입력을 받아 규칙들에서 구현되는 비즈니스 로직을 실행하는 블록체인 플랫폼 특정 서비스 계층을 포함한다. 데이터 동작들은 블록체인 플랫폼 특정 서비스 계층 및 블록체인 플랫폼에 의해 함께 관리된다.

[0006] 샘플 구성들에서, 블록체인 플랫폼 특정 서비스 계층은 규칙이 적용되는 객체들에서 존재하는 조건들을 파싱하고 평가하는 자바스크립트 객체 표기법(JavaScript Object Notation)(JSON) 조건 파서를 포함한다. 블록체인 규칙들 엔진에 의해 지원되는 각각의 타입의 기능성의 각각의 인스턴스는 표준 블록체인 규칙들 엔진 포맷의 JSON 문서에 의해 표현되는데, 여기서 각각의 JSON 문서는 다른 블록체인 거래들과 동일한 방법 및 합의를 사용하여 블록체인에 기입된다. 각각의 타입의 기능성에 대한 상태의 각각의 수정 또는 변경은 수정된 기능성에 대한 JSON 문서의 완전히 업데이트된 버전을 블록체인에 첨부하는 것에 의해 표현된다.

[0007] 다른 샘플 구성들에서, 언어 특정 SDK는 언어 특정 SDK에 의해 지원되는 프로그래밍 언어로 개발되는 블록체인 애플리케이션에 의해 클라이언트 애플리케이션이 규칙 실행 기능성에 액세스하는 것을 가능하게 하기 위한 인터페이스를 제공한다. 또한, 규칙들 엔진 API 계층에 의해 제공되는 인터페이스는 블록체인 규칙들 엔진과 인터페이싱하기 위해 다중 테넌트들 및 다중 애플리케이션들을 지원한다. 규칙들 엔진 API 계층은 언어 특정 SDK로부터 또는 사용자 인터페이스 콘솔을 통해 블록체인 규칙들 엔진의 규칙 신택스들 및 로직 커맨드들을 추가로 받는다. 규칙들 엔진 API 계층은, 신택스들 및 로직 커맨드들을, 블록체인 플랫폼 상에서 구현되는 블록체인 플랫폼 특정 서비스 계층에 대한 인터페이스를 통해 블록체인 플랫폼 특정 서비스 계층 상으로 전달될 수 있는 포맷으로 또한 변환한다.

[0008] 또 다른 샘플 구성들에서, 블록체인 플랫폼 특정 서비스 계층은, 블록체인 플랫폼 특정 서비스 계층에서 또는 블록체인 플랫폼 상에서 실현될 수 있는 기능성들 및 피처들의 서브세트를 포함하여, 블록체인 규칙들 엔진의 모든 실현에서 지원되어야 하는 인터페이스 플러스 기능성들(interface plus functionalities)을 설명한다. 샘플 구성들에서, 피처들은 다음의 것들 중 적어도 하나를 포함한다: 함수 생성, 함수 업데이팅, 함수 삭제, 규칙 생성, 규칙 업데이팅, 규칙 삭제, 규칙 활성화 및 비활성화, 규칙 가져오기 및 내보내기, 또는 규칙 롤백.

[0009] 추가적인 샘플 구성들에서, 블록체인 플랫폼 특정 표면 계층의 조건 파서는 블록체인 상에서 프로세싱 결과들의 실행 및 저장을 위해 전처리기 함수들 및 후처리기 함수들을 파싱한다. 전처리기 및 후처리기 함수들은 규칙들과 링크되어 전처리기 및 후처리기 함수들의 거동을 변경할 수 있다. 후처리기 함수들은, 규칙이 실행되고 실행된 규칙에 의해 통과 또는 실패 결과가 생성된 이후 수행되는 통과시 후처리기 함수(on-pass post-processor function) 또는 실패시 후처리기 함수(on-fail post-processor function) 중 적어도 하나를 포함할 수 있다.

[0010] 여전히 추가적인 샘플 구성들에서, 블록체인 플랫폼 특정 서비스 계층은, 블록체인 플랫폼 상에서 스마트 계약에 의해 직접 액세스될 규칙 관련 기능성들을 위해 블록체인 플랫폼 상에 직접 구현되는 기능성을 제공한다. 상이한 규칙 실행 기능성이 상이한 데이터 객체들에 적용될 수 있는데, 여기서 데이터 객체들은 테넌트, 애플리케이션, 규칙 그룹, 함수, 또는 규칙 중 적어도 하나를 포함할 수 있다. 각각의 타입의 규칙 실행 기능성의 각각의 인스턴스는 각각의 타입의 규칙 실행 기능성에 공통인 포맷의 JSON 문서에 의해 표현될 수 있다. 또한, 각각의 JSON 문서는 다른 블록체인 거래들과 동일한 방법 및 합의를 사용하여 블록체인에 기입될 수 있다. 규칙 실행 기능성에 대한 상태의 각각의 수정 또는 변경은 규칙 실행 기능성의 타입에 대한 JSON 문서의 완전히 업데이트된 버전을 블록체인에 첨부하는 것에 의해 표현될 수 있다.

[0011] 또 다른 추가적인 샘플 구성들에서, 클라이언트 애플리케이션에서 규칙들 엔진 API를 활용하여 애플리케이션들에서 동적 규칙 생성, 동적 규칙 수정, 및 규칙들의 동적 삭제를 가능하게 하는 신택스, 인터페이스들 및 요건들을 약술하고 블록체인 규칙들 엔진이 규칙들 데이터를 블록체인 상에 저장하는 방법을 약술하는 블록체인 규칙들 엔진 명세(blockchain rules engine specification)가 제공된다. 블록체인 규칙들 엔진 명세는 블록체인 규칙들 엔진이 블록체인 상에서 규칙 실행 이전에 실행되는 명령어들의 세트를 저장하는 방법을 명시할 수 있다. 또한, 블록체인 규칙들 엔진 명세는 다수의 테넌트들이 규칙들을 생성 및 관리하는 것 및 자격증명 데이터 및 규칙 함수들을 블록체인 상에 저장하는 것을 가능하게 하는 것에 의해 블록체인 상에서 다중 테넌시(multi-tenancy)를 구현할 수 있다. 블록체인 규칙들 엔진 명세는 다수의 레벨들의 액세스 제어를 추가로 통합할 수 있고, 그에 의해, 규칙에 대해 임의의 동작을 수행하기 위해, 클라이언트 애플리케이션은 테넌트의 자격증명뿐만 아니라 클라이언트 애플리케이션의 자격증명을 제공한다. 블록체인 규칙들 엔진 명세는 규칙 실행 기능성을 활용하기 위해 블록체인 규칙들 엔진에 의해 지원되는 애플리케이션 레벨 계층 구조를 또한 제공할 수 있고, 그에 의해, 클라이언트 애플리케이션은 생성된 규칙들 및 함수들을 사용할 수 있고 자격증명 및 메타데이

터를 블록체인 상에 저장할 수 있다. 블록체인 규칙들 엔진 명세는, 체인간 코드 통신을 확립하고 비 블록체인 애플리케이션들에 대한 연결을 확립하기 위해, 블록체인 규칙들 엔진과의 블록체인 애플리케이션들의 통신을 위한 플러그인을 또한 구현할 수 있다. 또한, 블록체인 규칙들 엔진 명세는, 규칙 실행을 수행하기 위해 클라이언트 애플리케이션과 블록체인 규칙들 엔진 사이의 연결을 생성하는 함수들을 제공하는 것에 의해, 생성된 규칙들을 실행하는 능력을 클라이언트 애플리케이션에게 제공할 수 있다. 블록체인 규칙들 엔진 명세는, 클라이언트 애플리케이션 코드에 직접 이용 가능한 함수들을 정의하는 함수 인터페이스들의 세트 및 블록체인 규칙들 엔진 명세에 대한 호출들을 규칙들 엔진 API에 대한 호출들로 변환하는 것에 의해 함수 인터페이스들을 구현하는 함수 구현 코드를 또한 포함할 수 있다.

[0012] 본원에서 설명되는 실시예들은 본 개시 전체에 걸쳐 설명되는 방법들을 구현하기 위한 명령어들로 코딩되는 컴퓨터 판독 가능 매체들 및 컴퓨터 시스템들을 또한 포괄한다. 예를 들면, 본원에서 설명되는 시스템들 및 방법들은 본원에서 설명되는 바와 같이 블록체인 플랫폼에 액세스하고 데이터를 저장하는 기능성을 제공하도록 클라우드의 컴퓨팅 플랫폼 상에서 구현될 수 있다.

도면의 간단한 설명

[0013] 본 개시는 첨부의 도면들의 도면들에서 제한이 아닌 예로서 예시되는데, 첨부의 도면들에서 유사한 참조 부호들은 유사한 요소들을 나타내며, 첨부의 도면들에서:

도 1은 예시적인 예에서 블록체인 규칙들 엔진의 개요를 제공하는 일반적인 블록 다이어그램이다.

도 2는 블록체인 규칙들 엔진의 샘플 구성의 구현의 상세한 컴포넌트 다이어그램이다.

도 3은 샘플 실현에서 블록체인 규칙들 엔진의 규칙 데이터 시스템 객체들의 개략도를 제공한다.

도 4는 사용자의 애플리케이션과의 블록체인 규칙들 엔진의 패브릭(Fabric)-Node.js SDK의 상호 작용 및 그들 둘 모두가 호스팅되는 실행 환경을 포함하는 사용자 애플리케이션 노드를 예시하는 다이어그램이다.

도 5는 본원에서 개시되는 블록체인 규칙들 엔진의 하나 이상의 실시예들을 구현하는 데 적절한 특수 목적 컴퓨터로 프로그래밍될 수 있는 통상적인 범용 컴퓨터의 블록 다이어그램이다.

발명을 실시하기 위한 구체적인 내용

[0014] 도 1 내지 도 5와 관련한 다음의 설명은 기술 분야의 숙련된 자들이 구체적인 실시예들을 실시하는 것을 가능하게 하기 위해 그들을 충분히 예시한다. 다른 실시예들은 구조적, 논리적, 프로세스, 및 다른 변경 사항들을 통합할 수 있다. 일부 실시예들의 부분들 또는 피처들은 다른 실시예들의 부분들 또는 피처들에 포함되거나 또는 대체될 수 있다. 청구항들에서 기술되는 실시예들은 그들 청구항들의 모든 이용 가능한 등가물들을 포괄한다.

[0015] **전문 용어**

[0016] **애플리케이션:** 애플리케이션은 블록체인 규칙들 엔진과 첨부되어 자신의 기능성들을 활용할 수 있는 컴포넌트(서비스, 또는 도구)를 나타낸다. 애플리케이션들은 테넌트들에 의해 생성되고 관리된다. 애플리케이션들은, 테넌트들과 마찬가지로, 그들 고유의 ID 및 비밀(secret)을 또한 갖는다. 규칙들 및 함수들에 대해 임의의 동작들을 행하기 위해, 테넌트는 (애플리케이션 비밀이 아니라) 애플리케이션 ID를 또한 제공할 필요가 있다. 이것은, 규칙들 및 함수들이 테넌트에게 직접 속하는 것이 아니라, 테넌트에 의해 직접 소유되는 애플리케이션에 속하기 때문이다. 규칙 및 함수 CRUD(create, read, update, and delete; 생성, 판독, 업데이트, 및 삭제) 동작들에서 애플리케이션 비밀이 사용되지 않는 이유는, 그들 동작들이, 애플리케이션 그 자체에 의해서가 아니라, 테넌트에 의해 개시되기 때문이다. 그러나, 규칙 실행은 애플리케이션에 의해 직접 개시되며 애플리케이션 비밀을 필요로 한다.

[0017] **블록체인:** 블록들로 칭해지는 지속적으로 성장하는 레코드들의 리스트는 암호화(cryptography)를 사용하여 링크되고 보안된다. 각각의 블록은 통상적으로 이전 블록의 암호 해시, 타임스탬프, 및 거래 데이터를 포함한다. 블록체인은 거래 데이터의 수정을 본질적으로 방지하도록 설계된다. 분산 원장(distributed ledger)으로서의 사용을 위해, 블록체인은 통상적으로 새로운 블록들을 검증하기 위한 프로토콜을 집합적으로 준수하는 피어 투 피어 네트워크에 의해 관리된다. 일단 기록되면, 임의의 주어진 블록의 데이터는 모든 후속하는 블록들의 수정 없이는 역행적으로 변경될 수 없는데, 이것은 네트워크 대다수의 공모를 필요로 한다.

[0018] **함수:** 함수는 규칙에서 존재하는 JSON 로직 신택스에서의 명령어들의 세트이다. 규칙 생성 동안, 테넌트는 규

칙 본문(Rule body)의 이미 생성된 함수(전용 엔드포인트를 사용하여 생성됨)를 참조할 수 있거나 또는 함수를 규칙 본문에 직접 기입할 수 있다. 함수의 목적은 그들의 타입들에 관해 학습하는 것에 의해 더 잘 이해될 수 있다. 함수들은 다음과 같이 설명되는 두 가지 타입들을 갖는다:

[0019] 전처리기 함수: 규칙의 전처리기 함수는 거래 객체가 규칙으로 공급되기 이전의 거래 객체에 대한 전처리기이다. 그것은 거래 객체가 규칙 실행을 위해 요구되는 데이터를 바로 가지지는 않지만 그 데이터에 대한 링크를 포함하는 시나리오들에서 사용될 수 있다. 그러한 경우, 전처리기 함수는 데이터에 대한 링크를 실제 데이터로 변환하기 위해 활용될 수 있다. 그 이후, 거래 객체는 규칙으로 공급될 수 있다.

[0020] 후처리기 함수: 규칙은 하나의 통과시 후처리기 함수 및 하나의 실패시 후처리기 함수를 가질 수 있다. 그들은 규칙 실행 이후 애플리케이션으로 반환되는 명령어들을 포함한다. 거래 객체가 규칙 실행을 위해 블록체인 규칙들 엔진으로 전송되는 경우, "거래 타입"에 대해 거래 객체와 동일한 값을 갖는 모든 규칙들이 그 거래 객체에 대해 실행된다. 규칙이 실패하면, 그것의 실패시 후처리기가 응답으로서 푸시되고, 그것이 통과하면, 통과시 후처리기가 푸시된다. 일단 모든 규칙들이 실행되면, 완전한 응답 객체(실패시 및 통과시 후처리기들의 어레이 응답을 포함함)가 그들의 사용을 위해 애플리케이션으로 반환된다.

[0021] **JSON:** 자바스크립트 오브젝트 표기법, 경량의 텍스트 기반 사람-판독 가능한 데이터 교환 포맷.

[0022] **Node.js®:** 브라우저 외부에서 자바스크립트(JavaScript) 코드를 실행하는 무료 오픈 소스 서버 환경. Node.js®는 Chrome(크롬) 브라우저의 자바스크립트 런타임에 기초하여 구축되는 비동기식 프로그래밍 플랫폼이다. Node.js®에 관한 더 많은 정보는 "About Node.js®"에서 발견될 수 있다. *Node.js*, Node.js Foundation, Node.js™ 및 JSON은 블록체인 규칙들 엔진의 블록체인 양태들에서 수반되는 기술들 및 표준들이다. Node.js는 브라우저 외부에서 자바스크립트 코드를 실행하는 무료 오픈 소스 서버 환경이다. 그것은 크롬 브라우저의 자바스크립트 런타임에 기초하여 구축되는 비동기식 프로그래밍 플랫폼이다.

[0023] **플랫폼 관리자(Admin):** 블록체인 규칙들 엔진에서, 테넌트들은 애플리케이션들, 규칙들, 및 함수들을 생성하기 위해, 그리고 또한 그들을 관리하기 위해 먼저 등록되어야 한다. 이것을 위해 특별히 설계되는 플랫폼 관리자의 역할은 블록체인 규칙들 엔진에서 테넌트들을 생성하고 관리하는 특권만을 갖는다.

[0024] **규칙들:** 규칙은 소정의 조건들, 일부 함수들 및 거래 타입을 포함하는 객체이다. 테넌트는 (소프트웨어 개발 키트(Software Development Kit)(SDK)를 통해) 블록체인 규칙들 엔진에 의해 제공되는 엔드포인트를 사용하는 것에 의해 그들을 생성할 수 있다. 규칙은 거래 객체에 적용될 수 있는데, 여기서 "적용한다"는 규칙에서 존재하는 조건들에 대해 거래 객체를 체크하는 것을 의미한다.

[0025] **스마트 계약:** 블록체인 또는 분산 원장 상에서 발생하는 범용 계산을 제공하는 컴퓨터 프로토콜. 스마트 계약 거래들은 간단할 수 있거나 또는 정교한 로직을 구현할 수 있다. 결과적으로 발생하는 거래들은 통상적으로 원장 참가자들에게 투명하고, 추적 가능하며 비가역적이다.

[0026] **테넌트:** 테넌트는 블록체인 규칙들 엔진을 사용하는 임의의 클라이언트(조직)를 나타낸다. 테넌트들은 애플리케이션들, 함수들 및 규칙들에 대해 기본 CRUD 동작들을 실행하기 위해 사용되는 ID 및 비밀을 갖는다. 테넌트는 블록체인 규칙들 엔진의 플랫폼 관리자에 의해 등록된다.

[0027] **거래:** 어떤 규칙들을 적용할지 결정하기 위해 사용되는 "거래 타입" 키를 갖는 자바스크립트 객체. 거래 객체는 규칙이 적용되고 결과를 생성하는 데이터를 포함한다. 거래 객체의 데이터에 대해 규칙이 실행되기 이전에, 그것은 전처리기 함수(규칙에서 존재함)에 의해 먼저 수정될 수 있다. 거래 객체들은, 계층 구조를 거래 레벨에서 유지되게 하려는 개념을 가지고 블록체인 규칙들 엔진이 만들어졌기 때문에 그렇게 명명된다. 거래 객체들은, 그들이 "거래 타입" 키를 최고 레벨 속성으로서 포함하는 한, 임의의 데이터를 가질 수 있다.

[0028] **거래 타입:** 거래 타입은 거래를 위해 함께 실행되도록 의도되는 다수의 규칙들의 그룹화를 나타낸다. 거래 타입 분류 피처는 데이터베이스의 규칙들을 관리하는 것을 훨씬 더 쉽고 직관적이게 만든다.

[0029] **개요**

[0030] 컴퓨터 애플리케이션 설계자들은, 특히 대규모 기업 환경에서, 현존하는 분산 원장 기술(distributed ledger technology)(DLT) 플랫폼들을 그들의 솔루션들에 점점 더 많이 통합하고 있다. 이들 플랫폼들은 엄격한 데이터 복제, 규칙 시행, 불변성 및 감사 가능성(auditability)을 제공하지만, 프로그래밍하고 운용하기 위해서는 전문적인 스킬들을 종종 필요로 한다. 근접한 DLT 노드의 통신의 추상화(데이터 및 그 데이터에 대한 규칙들 둘 모두의 기입을 포함함)를 카테고리 내에서 광범위한 잠재적 DLT 애플리케이션들을 지원할 데이터 및 규칙 프리미

티브들의 세트와 결합하는 것에 의해 선택된 플랫폼에서 전문적인 스킬들이 없어도 개발자들에 의한 클라이언트 애플리케이션들의 생성 및 동작을 가능하게 하는 서브시스템들이 사용될 수 있다. 프리미티브들은 통상적으로, 애플리케이션 개발자들이 그들을 특정한 사용 사례들에 적응시켜, DLT 플랫폼 스킬들을 필요로 하지 않으면서, 그리고 DLT 플랫폼이 제공하는 엄격한 규칙 시행을 상실하지 않으면서, 속성들 및 추가적인 데이터 규칙들을 사용하여 그들을 확장시킬 수 있도록, 구성된다. 또한, 클라이언트 애플리케이션 개발자들이 이들 프리미티브들을 설계하고 이들 프리미티브들에 대해 작용하는 인터페이스(들)는 DLT 플랫폼의 기술적 이해에 대한 어떠한 필요성도 완전히 추상화해 버려야 한다.

[0031] 구체적인 구현들은 단일의 클라이언트 애플리케이션의 전용 서브시스템들, 많은 클라이언트 애플리케이션들과의 사용을 위한 재사용 가능한 컴포넌트들, 또는 애플리케이션 인에이블먼트(application enablement)를 많은 클라이언트 애플리케이션들에 제공하는 서비스들일 수 있다. 다른 구현들은 하나의 또는 많은 인접한 DLT 노드들과의 통신을 중재할 수 있으며, 그들 노드들은 단일의 또는 다수의 DLT 네트워크들로부터 인출될 수 있다.

[0032] 샘플 서브시스템은 광범위한 카테고리 또는 가상 워크플로우 및 정보 공유 애플리케이션들에 대한 하이퍼레저 패브릭(Hyperledger Fabric) DLT 플랫폼의 간접적인 사용뿐만 아니라, 그러한 클라이언트 애플리케이션들의 여러 운용 상의 실제 실현들을 가능하게 할 수 있다. 그러한 서브시스템들을 구현하기 위해서는, 세 가지 핵심 요건들이 있다:

[0033] 1. 분산 원장 상에서 표현될 확장 가능한 일반 데이터 및 액션 프리미티브들의 세트;

[0034] 2. 클라이언트 애플리케이션에 의해 제공될 수 있는 바와 같은 그들 규칙들의 허용 가능한 확장들 외에 서브시스템의 고유의 데이터 규칙들 양자를 시행하는, 분산 원장에 임베딩되는, 그들 프리미티브들에 대해 동작하는 규칙들의 세트; 및

[0035] 3. 동작하기 위해 DLT 플랫폼의 어떠한 지식도 요구하지 않는 프로그래밍 인터페이스.

[0036] 결과적으로 나타나는 서브시스템은 일부 애플리케이션 카테고리의 지식 및 일부 DLT 플랫폼의 스킬들을 갖춘 일반 사용자가 그 카테고리에서 클라이언트 애플리케이션들의 생성을 지원하는 그들 자신의 서브시스템을 구성하는 것을 가능하게 하는 라이브러리 및 도구들의 모음을 제공한다. 결과적으로 나타나는 코드는 API 네트워크와 블록체인 프로토콜 사이의 미들웨어로서 구현되고, 그에 의해, 사용자는 블록체인의 동작의 광범위한 지식이 없어도 블록체인에 일관되게 인터페이싱할 수 있다. 그러한 애플리케이션들에서, 상위 계층으로부터의 사용 사례 특정 비즈니스 규칙들이 블록체인 상에서 기입되고 시행 가능하다. 블록체인 성능들 및 제어들을 방해하지 않으면서 워크플로우를 업데이트하기 위해 수시로 새로운 비즈니스 규칙들이 추가될 필요가 있다.

[0037] 규칙들 엔진은 그러한 서브시스템에서 새로운 규칙을 포함시키기 위한 또는 규칙들을 편집 또는 비활성화하기 위한 용이한 방식을 제공한다. 규칙들 엔진의 가치는 애플리케이션 코드로부터의 규칙들의 고유한 분리로부터 유래한다. 분리 외에도, 규칙들 엔진들은 어떠한 관련된 전문가 시스템들에도 영향을 주지 않으면서 규칙들이 쉽게 구성되는 것을 가능하게 한다.

[0038] 규칙들 엔진들이 따르는 원리들 중 일부는 다음과 같다:

[0039] 1. 규칙들 엔진들은 고도로 '플러그가능하고 재사용 가능'해야 하며, 이는 규칙들 엔진이 임의의 외부 애플리케이션과 함께 사용되도록 통합될 수 있고, 따라서 그것을 다수의 애플리케이션들에서 사용 가능하게 만들 수 있다는 것을 의미한다. 비즈니스 로직은 다양한 프로세스들 및 클라이언트 애플리케이션들에 의해 공동으로 사용될 수 있다.

[0040] 2. 규칙들 엔진들은 '확장 가능'해야 하며, 이는 임의의 타입의 복잡도를 갖는 임의의 수의 규칙들이 규칙들 엔진에서 구성될 수 있다는 것을 의미한다.

[0041] 3. 규칙들 엔진들은 빠르고, 쉽고, 정확한 규칙 변경들을 촉진하는 데 고도로 효율적이다. 규칙들 엔진들을 고도로 비용 효율적으로 만들기 위해서는, 심지어 비 IT 전문가들도 짧은 시간 내에 규칙들을 구성(추가, 편집, 삭제, 비활성화)할 수 있어야 한다.

[0042] 4. 규칙들 엔진들은 '진실의 소스'로서 작용해야 하며, 이는 애플리케이션 프로세스 또는 시스템에 대한 모든 비즈니스 규칙들 및 로직이 하나의 중앙 스토리지에서 유지되어야 한다는 것을 의미한다.

[0043] 하기에서 설명되는 블록체인 규칙들 엔진은 이들 원리 모두를 달성한다.

[0044] 블록체인 채택을 둘러싸는 현재의 에너지는 인터넷의 초기 채택을 연상시킨다. 애플리케이션들 및 시스템들이

성숙해짐에 따라 그들에게 부담을 주었던 로직 관리와 관련한 동일한 실수들을 되풀이하거나 또는 방지하는 잠재력이 존재한다. 초기 웹사이트들 및 애플리케이션들과 유사하게, 블록체인의 암호 화폐 사용들은 상대적으로 간단한 데이터 구조들 및 그들을 관리하기 위한 하드코딩된 비즈니스 로직을 특징으로 한다. 그러나, "스마트 계약들"의 후속하는 발명은 임의의 복잡도의 규칙들 및 데이터에 대한 문을 개방하였다.

[0045] 공급망 물류와 같은 기업 사용 사례들은 블록체인에 관한 정보 및 논리를 전통적인 솔루션들에 필적하는 볼륨들에 넣지만, 표준화된 데이터 관리 도구들 중 어느 것도 갖지 않는다. 모든 프로그래머가 애플리케이션 코드 베이스 내에서 자기 자신의 로직 세트를 설계하는 상황에서, 그 결과는 다시 한번, 그것을 기입하는 프로그램들(스마트 계약들)에 밀접하게 커플링되는 로직이다. 본원에서 설명되는 블록체인 규칙들 엔진과 같은 관계형 계층은 이러한 커플링을 감소시키는 데 필요로 한다.

[0046] 기술 분야의 숙련된 자들은 블록체인의 모든 피처가 기업 애플리케이션에서 사용될 필요는 없다는 걸 인식할 것이다. 일반적인 상황들은 암호화 변조 방지 및 데이터 복제와 같은 피처들을 달성하는 편리한 방식으로서 블록체인 플랫폼들의 사용을 포함한다. 또한, 기업들은, 완화된 합의 규칙들 또는 중앙 집중화된 프론트 엔드를 사용하여, 그들의 완전한 제어 하에서 블록체인 네트워크를 구현할 수 있다. 관리자들 및 내부 감시자들이 그들 마음대로 블록체인 데이터를 직접 업데이트하는 것을 허용하는, 지원 직원에 대한 엄격한 스마트 계약 시행과 함께 블록체인 네트워크들을 셋업하는 높은 레벨들의 신뢰를 갖는 컨소시엄들이 사용될 수 있다.

[0047] 스마트 계약에 저장되는 거래 데이터 및 결정 데이터는 애플리케이션들에서 더 큰 가치를 갖는다. 블록체인에 저장되는 규칙 결과들 및 함수들은 합의에 의해 관리되지 않고도 다른 함수들 또는 로직에서 후속하여 사용될 수 있고, 그에 의해, 동작들을 빠르고, 효율적이며 신뢰 가능하게 만들 수 있다. 유일한 요건은 블록체인 상에 저장되는 규칙 결과들 및 함수들을 사용하기 위한 액세스이다. 블록체인 상에 저장되는 규칙들 및 함수들은, 원래의 스마트 계약 프로그래머들에 의해 구상되었던 것을 넘어서는 목적들을 위해, 블록체인 내부의 또는 외부의 애플리케이션, 규칙, 함수, 또는 시스템에 의해 효율적으로 판독 및 프로세싱될 수 있다는 것이 인식될 것이다.

[0048] **블록체인 구동식 규칙들 엔진의 샘플 실시예**

[0049] 본원에서 설명되는 블록체인 규칙들 엔진은, 규칙들, 자신의 거래들 및 결과들을 블록체인 상에 저장하며, 상기에서 언급되는 이점들을 제공하는 규칙들 엔진의 실제 구현을 묘사하는 플러그가능한 서비스이다. 특정한 비즈니스 실현을 위해 개발되는 바와 같은 블록체인 규칙들 엔진은 블록체인뿐만 아니라 비 블록체인 애플리케이션들에 의해 활용될 수 있는 플러그가능한 서비스이다.

[0050] 노드 패키지 관리자(Node Package Manager)(NPM) 패키지로서 개발되는 SDK를 사용하여, 샘플 구성들에서의 블록체인 규칙들 엔진은 임의의 Node.js 애플리케이션에 의해 사용될 수 있고, 따라서, 그것을 거의 모든 애플리케이션들과 통합될 수 있는 고도로 구성 가능한 규칙들 엔진으로 만들 수 있다. 본원에서 설명되는 블록체인 규칙들 엔진은 다양한 플랫폼들 및 언어들에 걸친 구현을 위해 설계되었다. 본 출원은 하이퍼레저 패브릭 및 Node.js 프로그래밍 언어에 기초하여 하나의 그러한 구현을 설명한다. 기술 분야의 숙련된 자들에 의해 인식될 바와 같이, 하이퍼레저 패브릭은 산업 사용 사례들의 광범위한 세트에 대해 모듈성(modularity) 및 다양성을 제공하는 기업 등급의 허가형 분산 원장 플랫폼이다.

[0051] 블록체인 규칙들 엔진은 모든 구현들을 안내하도록 의도되는 명세들, 인터페이스들, 및 핵심 컴포넌트들의 세트와 관련하여 설명될 것이다. 도 1은 예시적인 예에서 블록체인 규칙들 엔진의 개요를 제공하는 일반적인 블록 다이어그램이다.

[0052] 블록체인 규칙들 엔진은 다양한 사용자 프로그래밍 언어들 및 블록체인 플랫폼들에 대해 구현될 수 있다. 시스템의 부분들은 모든 구현들에 공통이며, 부분들은 맞춤화되지만 공통 명세들의 세트를 따른다. 도 1은 이들 컴포넌트들 및 명세들을 보여주는 적응된 컴포넌트 다이어그램이다. 공통 명세들은 추상적인 요건들을 나타내기 위해 <<명세>> 스테레오타입을 갖는 컴포넌트들로서 표현된다. 명세들은 모든 블록체인 규칙들 엔진 구현들 중에서 공통 방식으로 구현될 것으로 예상되는 양태들이다. 도시되는 인터페이스들은 또한 모든 구현들에 대해 공통인 명세들이다.

[0053] 도 1에서 예시되는 바와 같이, 블록체인 규칙들 엔진은 블록체인 규칙들 엔진을 활용하도록 의도되는 클라이언트 애플리케이션(100)과 상호 작용한다. 클라이언트 애플리케이션(100)은 임의의 써드파티 애플리케이션, 서비스 또는 도구일 수 있다. 언어 특정 SDK(110)는 블록체인 규칙들 엔진 서비스들에 연결하기 위한 인터페이스(115)를 클라이언트 애플리케이션(100)에 제공한다. 언어 특정 SDK(110)에 의해 노출되는 주요 기능성은 규칙

실행 기능성이다. 블록체인 규칙들 엔진 API 계층(120)은, 언어 특정 SDK(110)로부터 커맨드들을 수신하여 프로세싱하고 내부 컴포넌트들과 통신하여 소망되는 결과를 생성하는 규칙들 엔진 API(125)를 제공한다. 블록체인 플랫폼 특정 서비스 계층(130)은 블록체인 시스템들(이 예에서는 하이퍼레저 패브릭)과 함께 작동하도록 설계되며 규칙들 엔진 서비스 계층 인터페이스(135)를 통해 규칙들 엔진 API 계층(120)으로부터 입력을 받아 비즈니스 로직을 실행한다. 규칙 객체들에서 존재하는 조건들을 파싱하고 평가하기 위해, 블록체인 플랫폼 특정 서비스 계층(130)은 JSON 조건 파서(140)로 칭해지는 모듈을 사용할 수 있다.

[0054] 시스템에서의 데이터 동작들은 블록체인 플랫폼 특정 서비스 계층(130) 및 블록체인 플랫폼(150)에 의해 함께 관리된다. 작업의 정확한 구분은 블록체인 플랫폼(150)의 성능들에 의존한다. 필수 기능성은 다음을 포함하는 두 가지 하위 명세들에 의해 추가로 상세하게 주어진다: 지원되어야만 하는 데이터 동작들 및 표준화된 데이터 표현들(JSON 스키마들)을 정의하는 블록체인 상의 규칙 데이터(155), 및 스마트 계약들에서 어떤 규칙 데이터 피처들이 액세스 가능해야 하는지를 명시하는 스마트 계약 통합(160).

[0055] **블록체인 규칙들 엔진 언어 특정 SDK들(110)**

[0056] 블록체인 규칙들 엔진 언어 특정 SDK들(110) 각각은 블록체인 애플리케이션 개발자들에게 인기 있는 프로그래밍 언어(Node.js)로 개발되는 임의의 블록체인 애플리케이션에 의해 블록체인 규칙들 엔진(규칙 실행 기능성)의 모든 피처들에 액세스하기 위한 인터페이스를 제공한다. 블록체인 규칙들 엔진 언어 특정 SDK들(110)은, 블록체인 규칙들 엔진과의 인터페이싱 연결을 확립하고, 그에 의해, 인터페이싱된 클라이언트 애플리케이션(들)(100)으로부터의 규칙 실행 요청을 개시하기 위해, Node.js 애플리케이션에 의해 임포트되는(imported) 노드 패키지 관리자(NPM) 패키지이다.

[0057] **블록체인 규칙들 엔진 API 계층(120) 및 UI 콘솔**

[0058] 블록체인 규칙들 엔진의 핵심 컴포넌트들 중 하나는 블록체인 규칙들 엔진의 모든 실현들에서 사용되는 규칙들 엔진 API 계층(120)이다. 규칙들 엔진 API 계층(120)은 상호 작용적이고 직관적인 사용자 인터페이스를 제공하여, 블록체인 규칙들 엔진과 인터페이싱하도록 다중 테넌트들 및 다중 애플리케이션들을 지원한다. 규칙들 엔진 API 계층(120)은 또한 사용자 인터페이스 콘솔(도시되지 않음)을 통해 구축되는 규칙들의 컴파일(compilation) 및 요약을 제공한다. 또한, 규칙들 엔진 API 계층(120)은 언어 특정 SDK(110)로부터 유래하는 블록체인 규칙들 엔진 API 계층(120)을 통해 또는 사용자 인터페이스 콘솔을 통해 블록체인 규칙들 엔진의 규칙 신택스들 및 로직 커맨드들을 받는다. 규칙들 엔진 API 계층(120)은 그들 신택스들 및 로직 커맨드들을, 블록체인 플랫폼(150) 상에서 구현되는 블록체인 규칙들 엔진 서비스 계층 인터페이스(135) 상으로 전달될 수 있는 포맷으로 변환한다.

[0059] **블록체인 규칙들 엔진 다중테넌시**

[0060] 블록체인 규칙들 엔진은, 다수의 애플리케이션들에 의해 사용될 동일한 블록체인 규칙들 엔진 인스턴스를 제공한다는 점에서, 다중 테넌트 지원을 제공한다. 도구의 이 명세는, 기업 또는 생태계에서 하나보다 더 많은 애플리케이션이 가장 복잡하고 강력한 규칙들을 구축하는 것을 허용하고, 그에 의해, 기업 또는 생태계는 모든 인터페이싱 애플리케이션들 내에서 비즈니스 로직 상호 작용을 가능하게 한다. 블록체인 규칙들 엔진의 다중 테넌시의 원격 액세스 및 가상화 피처들은 블록체인 규칙들 엔진을 사용하는 애플리케이션들에게 더 많은 제어 및 더 많은 유연성을 제공한다. 또한, 블록체인 규칙들 엔진의 다중 테넌트 피처 때문에, 가상화 및 원격 액세스 피처들이 기업 내에서 완벽하게 사용될 수 있다.

[0061] 블록체인 규칙들 엔진 서비스 계층 명세(130)

[0062] 블록체인 규칙들 엔진의 서비스 계층 명세(130)는, 블록체인 규칙들 엔진의 모든 실현에서 지원될 기능성들 외에, 인터페이스를 설명한다. 전체 인터페이스를 다루는 것과는 별개로, 이 명세는 이 블록체인 규칙들 엔진의 서비스 계층(130)에서 또는 블록체인 플랫폼(150) 그 자체 상에서 실현될 수 있는 피처들 및 기능성들의 서브세트를 또한 다룬다. 이들 피처들은 다음을 포함한다:

[0063] - 함수 생성

[0064] - 함수 업데이팅

[0065] - 함수 삭제

[0066] - 규칙 생성

[0067] - 규칙 업데이팅

[0068] - 규칙 삭제

[0069] - 규칙 활성화 및 비활성화

[0070] - 규칙 가져오기 및 내보내기

[0071] - 규칙 롤백

[0072] 블록체인 규칙들 엔진의 서비스 계층(130)은 Node.js로 구현되며 타겟팅되는 블록체인 플랫폼(150) 또는 SDK 언어에 관계없이 재사용된다. 추가적인 피처들은 블록체인 플랫폼(150) 그 자체를 타겟팅할 수 있다. 이들은 하기의 블록체인 플랫폼 명세와 관련하여 설명된다.

[0073] **블록체인 규칙들 엔진 함수 저장 및 활용**

[0074] 블록체인 규칙들 엔진은 블록체인 내부에서 함수들을 저장하는 피처를 추가로 제공한다. 이들 함수들은 다음의 두 가지 타입들을 갖는다: 전처리기 함수들 및 후처리기 함수들. 이들 함수들은 특수 JSON 포맷을 사용하여 기입되는데, 특수 JSON 포맷은 그 다음, 실행을 위해 블록체인 규칙들 엔진에 의해 파싱된다.

[0075] 이들 함수들은 규칙들과 링크되어 그들의 거동을 변경할 수 있다. 임의의 링크된 전처리기 함수들은 규칙의 (클라이언트로부터의) 제공된 입력을 수정하기 위해 사용될 수 있다.

[0076] 후처리기 함수들은 다음의 두 가지 타입들을 갖는다: 통과시 후처리기 함수들 및 실패시 후처리기 함수들. 그들의 이름이 암시하는 바와 같이, 그들은 규칙이 실행되고 그것의 결과가 생성된 이후에(통과 또는 실패) 그들의 역할을 행한다. 결과에 따라, 대응하는 후처리기 함수가 호출자에게 반환되는데, 호출자는 이를 자신의 끝에서 실행할 수 있다. 하기의 도 2는 이들 개념들을 추가로 설명한다.

[0077] **패브릭/NODEJS에서의 블록체인 규칙들 엔진의 샘플 구성**

[0078] 도 2는 블록체인 규칙들 엔진(200)의 샘플 구성의 구현의 상세한 컴포넌트 다이어그램이다. Node.js는 언어 특정 SDK(110)에 의해 사용되는 SDK 프로그래밍 언어이고, 하이퍼레저 패브릭은 사용되는 블록체인 플랫폼(150)이다. 블록체인 규칙들 엔진(200)의 도 2의 모델은 패브릭/Node.js 구현이 일부 추가적인 기능성들과 함께 도 1에서 약술하는 명세들을 어떻게 실현하는지에 초점을 맞춘다.

[0079] 블록체인 규칙들 엔진 Node.js 소프트웨어 개발자 키트(SDK) 컴포넌트(210)는 사용자 언어 특정 기능성을 실현한다. 규칙들 엔진 API 계층(120), 블록체인 플랫폼 특정 서비스 계층(130), 및 인터페이스들(115, 125, 및 135)은, 도 1과 동일하게, 모든 구현들에서 공통이다. 그들 아래에서, 블록체인 규칙들 엔진 Node.js 패브릭 서비스 계층(220)은 모든 플랫폼 동작들을 위한 외관을 실현한다. 블록체인 플랫폼 명세(150)에 대한 자신의 종속성을 통해 블록체인 규칙들 엔진 Node.js 패브릭 서비스 계층(220)을 직접 지원하는 것은 패브릭 CouchDB 구현(230)인데, 이것은 소정의 인덱싱 성능들을 위해 직접 액세스된다. 패브릭 실현에서, 스키마 인덱스 수정 및 사용자 특권들은 블록체인 규칙들 엔진 Node.js 패브릭 서비스 계층(220)에서 부분적으로 구현되는데, 이것은 패브릭 CouchDB 구현(230)을 직접 호출한다.

[0080] 패브릭 피어(240)는 블록체인 네트워크에 대한 원장들 및 스마트 계약들의 인스턴스들을 호스팅한다. 원장들은 스마트 계약들에 의해 생성되는 모든 거래들을 불변적으로 기록한다(이것은 하이퍼레저 패브릭에서 체인 코드 또는 스마트 계약에서 포함되며, 이것은 지원되는 프로그래밍 언어로 기입되는 원장에 액세스하는 코드의 조각이다). 스마트 계약들 및 원장들은 각각 네트워크 내의 공유된 프로세스들 및 공유된 정보를 캡슐화하기 위해 사용된다. 패브릭 피어(240)을 통해, 애플리케이션들은 체인 코드들을 실행하여 원장을 질의하거나 또는 업데이트할 수 있다.

[0081] **블록체인 규칙들 엔진 블록체인 플랫폼 명세(150)**

[0082] 블록체인 규칙들 엔진 블록체인 플랫폼 명세(150)는, 규칙 관련 기능성들이 블록체인 플랫폼 상에서 스마트 계약들에 의해 사용 가능하기 위해 그 블록체인 플랫폼 상에 직접 구현되어야만 하는 기능성을 설명한다. 블록체인 플랫폼들 그들 자체들과 같이, 이들 구현들은 접근법이 크게 다를 것이다. 이들 명세들은, 조건 데이터 및 스키마들에 대한 공통의 JSON 기반의 데이터 포맷을 포함하여, 여전히 공통일 것을 설명한다. 명세들은 다음의 두 개의 카테고리들로 분할된다: 블록체인 상의 규칙 데이터 및 스마트 계약 통합.

[0083] 블록체인 명세(155) 상의 규칙 데이터

[0084] 블록체인 규칙들 엔진(200)의 주요 자산들은 그것의 규칙들이다. 블록체인 규칙들 엔진(200)은 광고된 기능성들을 달성하기 위해 데이터에서 소정의 수의 추상화를 활용한다. 이들 추상화들은 테넌트, 애플리케이션, 규칙 그룹, 함수 및 규칙을 포함하는 객체들로서 정의된다. 이들 개념들이 상이한 블록체인 플랫폼들에 대해 구현될 방식에서 상당한 차이들이 존재하지만, 그러나 다음의 사항은 모든 플랫폼 상에서 참일 것이다:

[0085] - 다섯 가지 타입들의 기능성들의 각각의 기능성의 각각의 인스턴스는 각각의 타입의 기능성에 대해 공통인 표준 블록체인 규칙들 엔진 포맷의 독립형 JSON 문서에 의해 표현될 것이다.

[0086] - 이들 문서들은 다른 블록체인 거래들과 동일한 방법 및 합의를 사용하여 블록체인에 기입된다.

[0087] - 이 요소들 중 하나에 대한 상태의 각각의 수정 또는 변경은 JSON 문서의 완전하고 업데이트된 버전을 블록체인에 첨부하는 것에 의해 표현된다.

[0088] 풍부한 세부 사항들을 갖는 다섯 가지 타입들의 객체들에 대한 JSON 스키마 정의들은 블록체인 규칙들 엔진(200)의 설계에서 공식적으로 명시된다.

[0089] 스마트 계약 통합 명세(160)

[0090] 블록체인 규칙들 엔진(200)의 핵심 양태는 스마트 계약들로부터 직접 대부분의 규칙 기능성(주로, 상기에서 언급된 객체들에 대한 CRUD 동작들)에 액세스하는 자신의 능력이다. SDK와 같이, 공식적인 명세는 지원되어야만 하는 함수들을 열거하며, 블록체인 플랫폼 구현들에 걸쳐 일관성을 극대화하기 위해 선호되는 형태, 명명법(naming), 및 의미론(semantics)을 제공한다. 다시 말하면, 스마트 계약들로부터 블록체인 규칙들 엔진(200)에 액세스하는 것은 상이한 블록체인 플랫폼 구현들에 걸쳐 가능한 한 일관성이 있을 것이다.

[0091] **블록체인 규칙들 엔진 Node.js SDK(210)**

[0092] 블록체인 규칙들 엔진 Node.js 소프트웨어 개발자 키트(SDK)(210)는, 블록체인 규칙들 엔진에 의해 노출되는 API들을 활용하여 애플리케이션들에서 동적 규칙 생성을 가능하게 하는 신택스, 인터페이스들 및 요건들을 약술한다 - 의미론 및 신택스는 비 블록체인 프로그래머들에게 이미 친숙하고 모든 블록체인 플랫폼들에 걸쳐 동일하다. 블록체인 규칙들 엔진 Node.js SDK(210)는 블록체인 규칙들 엔진 API들과 인터페이싱되는 애플리케이션들에서 규칙들의 동적 수정 및 삭제를 추가로 가능하게 한다 - 다시, 의미론 및 신택스는 비 블록체인 프로그래머들에게 이미 친숙하고 모든 블록체인 플랫폼들에 걸쳐 동일하다.

[0093] 블록체인 규칙들 엔진 Node.js SDK(210)는 블록체인 규칙들 엔진(200)이 규칙들 데이터를 블록체인 상에서 저장하는 방법을 약술한다. 규칙들, 그것의 실행 결과들(성공/에러), 및 결과 거래들(후처리기 함수들)은 모두 블록체인 상에서 저장된다. 규칙들에 대해 행해지는 임의의 수정(그것의 버저닝(versioning))이 또한 블록체인 상에 배치된다. 블록체인 규칙들 엔진 Node.js SDK(210)는 블록체인 규칙들 엔진(200)이 전처리기 함수들 - 규칙 실행 이전에 실행되는 명령어들의 세트 - 을 블록체인 스토리지 디바이스 상에 저장하는 방법을 추가로 명시한다. 블록체인 규칙들 엔진 Node.js SDK(210)는 다수의 테넌트들이 블록체인 상에서 규칙들을 생성하고 관리하는 것을 가능하게 하는 것에 의해 블록체인 상에서 다중 테넌시를 추가로 구현한다. 테넌트들은 그들의 자격증명 데이터, 규칙 함수들을 블록체인 상에서 또한 저장한다. 다중 테넌시 외에도, 블록체인 규칙들 엔진 Node.js SDK(210)는 규칙 실행 피처들을 활용하기 위해 블록체인 규칙들 엔진에 의해 지원되는 애플리케이션 레벨 계층 구조를 제공한다. 애플리케이션들은 생성된 규칙들 및 함수들을 사용할 수 있고, 자격증명 및 메타데이터를 블록체인 상에서 또한 저장할 수 있다. 규칙들을 블록체인 상에서 저장하는 것 외에도, 테넌트들, 애플리케이션들, 함수들 및 규칙들과 관련되는 모든 동작들은 블록체인 상에서 안전하게 로그된다.

[0094] 블록체인 규칙들 엔진 Node.js SDK(210)는 거래 타입들에 기초하여 규칙들에 대한 카테고리들을 생성하는 능력을 추가로 명시한다. 블록체인 규칙들 엔진 Node.js SDK(210)는 API 호출들을 통해 애플리케이션들에 대한 거래 타입들의 생성을 추가로 지원하고, 그에 의해, 일단 거래 타입들이 등록되면 애플리케이션의 사용자들은 규칙들을 생성한다. 또한, 블록체인 규칙들 엔진 Node.js SDK(210)는 블록체인 규칙들 엔진(200)과의 블록체인 애플리케이션들의 통신을 위한 플러그인을 구현하여 체인간 코드 통신을 확립한다. 추가적으로, 이 플러그인은 비 블록체인 애플리케이션들에 대한 연결을 또한 확립할 수 있다.

[0095] **NODE.JS SDK를 사용한 패브릭 상에서의 블록체인 규칙 엔진**

[0096] 블록체인 규칙들 엔진(200)에 대한 상기의 명세들에 기초하여, 블록체인 규칙들 엔진(200)은 샘플 구성에서 Node.js 애플리케이션들에 대해 SDK를 그리고 그것의 블록체인 플랫폼에 대해 하이퍼레저 패브릭을 사용하여 구현될 수 있다. 블록체인 플랫폼 상에서 규칙들 데이터 성능들을 스마트 계약들이 이용 가능하게 만드는 것은

스마트 계약들을 사용하는 전체 블록체인 규칙들 엔진(200)의 구현을 필요로 하였다. 블록체인 규칙들 엔진(200)은 테넌트들과 애플리케이션들 사이의 논리적 데이터 분리 및 블록체인 플랫폼과의 긴밀한 통합을 추가로 구현한다.

[0097] 샘플 구성에서, 블록체인 규칙들 엔진(200)은 Node.js, 하이퍼레저 패브릭 및 Apache Couch(아파치 카우치) DB를 사용하여 구현된다. Node.js는 웹 서버 측 개발을 위한 현재 전세계에서 가장 인기 있는 프레임워크이며, 한편 하이퍼레저 패브릭은 다수의 블록체인 규칙들 엔진의 요건들을 부분적으로 충족하는 아키텍처를 갖추고 있다. Apache CouchDB®는 월드 상태 데이터베이스(world state database)에 대한 옵션으로서 하이퍼레저 패브릭에서 통합된다. CouchDB®는 체인 코드 데이터 값들이 블록체인 규칙들 엔진의 규칙 데이터처럼 JSON으로서 모델링될 때 풍부한 질의들을 지원한다.

[0098] **블록체인 규칙들 엔진 패브릭 구현**

[0099] 도 3은 실현(300)에서 블록체인 규칙들 엔진의 규칙 데이터 시스템 객체들의 개략도를 제공한다. 패브릭 상에서의 블록체인 규칙들 엔진의 규칙 데이터 실현은 다음의 총 세 개의 역할들을 수반한다: 플랫폼 관리자(310), 테넌트(320), 및 애플리케이션(330). 플랫폼 관리자(310)는 제품 배치의 시점에서 등록된다. 이 역할은 테넌트들(320)을 시스템(300)에 등록하는 것을 담당한다. 테넌트들(320)은 ID 및 비밀을 가지는데, 그들은 이들을 사용하여 규칙들(340) 및 함수들(350)을 생성/판독/업데이트/삭제할 수 있다. 테넌트들(320)은 애플리케이션들(330)을 생성하는 것을 또한 담당한다. 애플리케이션들(330)은, 제품의 API 계층을 사용하여 규칙 실행 요청들을 개시하기 위해 SDK에 의해 필요로 되는 그들 자신의 ID 및 비밀을 갖는다. 거래 타입(360)은 거래를 위해 함께 실행되도록 의도되는 다수의 규칙들(340)을 그룹화할 수 있다.

[0100] 도 3은 블록체인 규칙들 엔진(200)의 다양한 데이터베이스 엔티티들 사이의 관계를 설명한다. 연결 링크들의 양쪽 단부들에 있는 숫자들은 엔티티들 사이의 관계를 그들의 숫자들의 관점에서 나타낸다. 예를 들면, 도 3에서, 플랫폼 관리자(310)는 "1"로서 마킹되고 테넌트(320)는 "0**1"로서 마킹되는데, 이것은 하나의 테넌트(320)가 단지 하나의 플랫폼 관리자(310)에게만 관련될 수 있지만, 그러나 하나의 플랫폼 관리자(310)는, 0부터 시작하여, 임의의 개수의 테넌트들(320)에 관련될 수 있다는 것을 의미한다.

[0101] 블록체인 규칙들 엔진 규칙 데이터 액세스 제어

[0102] 블록체인 규칙들 엔진(200)이 다중 테넌트 시스템이기 때문에, 그것은 다수의 레벨들의 액세스 제어를 통합한다. 규칙에 대한 임의의 동작(판독 또는 기입)을 수행하기 위해, 호출자는 테넌트의 자격증명뿐만 아니라 애플리케이션의 자격증명을 제공해야 한다. 블록체인 규칙들 엔진(200)의 본문들의 계층 구조는 다음과 같다:

[0103] 플랫폼 관리자 > 테넌트 > 애플리케이션 > 함수들 및 규칙들.

[0104] 또한, 테넌트들(320)은 플랫폼 관리자(310)에 의해서만 생성될 수 있고, 애플리케이션들(330)은 테넌트들(320)에 의해서만 생성될 수 있다.

[0105] 블록체인 규칙들 엔진 스마트 계약 통합

[0106] 하이퍼레저 패브릭 상에서의 블록체인 규칙들 엔진의 구현은 블록체인 플랫폼 상에서의 스마트 계약들(160)의 설치를 허용한다. 이러한 방식으로, 비즈니스 로직은 블록체인 규칙들 엔진 Node.js 패브릭 서비스 계층(220) 및 블록체인 플랫폼(150) 그 자체 사이에서 분할된다.

[0107] 블록체인 규칙들 엔진 패브릭 서비스 계층

[0108] 블록체인 규칙들 엔진 Node.js 패브릭 서비스 계층(220)은 API 계층/SDK로부터 입력들을 받고 블록체인 규칙들 엔진 체인 코드, 및 CouchDB 월드 상태 데이터베이스(230)를 호출하는 것에 의해 그들을 액션으로 변환한다.

[0109] 블록체인 규칙들 엔진 Node.js SDK

[0110] 블록체인 규칙들 엔진 Node.js SDK(210)는 클라이언트 애플리케이션(100)에게 생성된 규칙들을 실행하는 능력을 제공한다. 구체적으로, 블록체인 규칙들 엔진 Node.js SDK(210)는 블록체인 규칙들 엔진(200)과의 연결을 생성하고 규칙 실행을 수행하는 함수들을 제공한다.

[0111] **블록체인 규칙들 엔진 패브릭-Node.js SDK**

[0112] 도 4는 사용자의 애플리케이션(420)과의 블록체인 규칙들 엔진의 패브릭-Node.js SDK(430)의 상호 작용 및 그들

둘 모두가 호스팅되는 실행 환경을 포함하는 사용자 애플리케이션 노드(400)를 예시한다. 사용자 애플리케이션 노드(400)는, 도 4의 모델의 최상위 레벨의 세부 사항에서 표시되는 바와 같이, Node.js에 대한 지원 및 블록체인 규칙들 엔진 API(125)의 웹 소켓을 통해 블록체인 규칙들 엔진 웹 서비스(API 계층)(410)의 실행 중인 인스턴스와 통신하도록 구성될 능력에 의해 제한되는 광범위한 잠재적 구현들을 포괄한다.

[0113] 사용자 애플리케이션 노드(400)를 상세하게 살펴보면, Node.js의 사용자 애플리케이션(420)은 패브릭/Node.js 블록체인 규칙들 엔진 SDK(430)에 대한 종속성을 갖는 컴포넌트로서 모델링된다. 더욱 상세하게 살펴보면, 패브릭/Node.js 블록체인 규칙들 엔진 SDK(430)는 두 개의 부분들로 모델링된다. Node.js 래퍼(Wrapper) 함수 인터페이스들(440) 세트는 사용자의 애플리케이션 코드가 직접 이용 가능한 함수들을 정의한다. Node.js 래퍼 함수 인터페이스들(440)은 일반적으로 도 1에 소개되는 언어 특정 SDK <<명세>>(110)를 따른다. Node.js 래퍼 함수 구현 코드(450)는, 패브릭/Node.js 블록체인 규칙들 엔진 SDK(430)에 대한 호출들을 블록체인 규칙들 엔진 API(125)에 대한 호출들로 변환하는 인터페이스를 구현한다.

[0114] 블록체인 규칙들 엔진 명세들에 의해 요구되는 함수들 외에도, 패브릭/Node.js 실현은 규칙 실행 능력을 포함한다. 상기에서 언급되는 블록체인 규칙들 엔진(200)의 피처들 외에도, 도 4의 패브릭/Node.js 버전은 테넌트-애플리케이션 계층 구조 및 클라이언트 애플리케이션들에게 규칙 실행과 같은 피처들을 제공하는 프로그래머 친화적 SDK를 사용하여 블록체인 상에서 규칙 데이터를 달성한다.

[0115] 본원의 설명은 Hyperledger® Fabric과 관련하여 제공되지만, 본원의 설명에서 설명되는 기술들은 AION, ArcBlock, EOS, NEO, Hyperledger® Sawtooth, NxT, QTUM, Quorum, Smilo, Tezos, TRON, Wanchain, 또는 Zilliqa와 같은 다른 블록체인 네트워크들과도 또한 사용될 수 있다는 것이 기술 분야의 숙련된 자들에 의해 인식될 것이다. 본원에서 사용되는 전문 용어는 Hyperledger® Fabric 블록체인 네트워크와 관련된다. 대응하는 용어들은 다른 블록체인 네트워크들, 예컨대 언급된 것들의 맥락에서 사용될 것이다.

[0116] **컴퓨터 실시예**

[0117] 도 5는 본원에서 개시되는 블록체인 규칙들 엔진(200)의 하나 이상의 실시예들을 구현하는 데 적절한 특수 목적 컴퓨터로 프로그래밍될 수 있는 통상적인 범용 컴퓨터(500)의 블록 다이어그램이다. 상기에서 설명되는 블록체인 규칙들 엔진(200)은 임의의 범용 프로세싱 컴포넌트, 예컨대 자신에게 부과되는 필요한 작업 부하를 핸들링하기에 충분한 프로세싱 파워, 메모리 리소스들, 및 통신들 스루풋 성능을 갖춘 컴퓨터 상에서 구현될 수 있다. 컴퓨터(500)는 보조 스토리지(504), 리드 온리 메모리(read only memory)(ROM)(506), 랜덤 액세스 메모리(random access memory)(RAM)(508), 입력/출력(input/output)(I/O) 디바이스들(510), 및 네트워크 연결성 디바이스들(512)을 포함하는 메모리 디바이스들과 통신하는 프로세서(502)(이것은 중앙 프로세서 유닛 또는 CPU로서 지칭될 수 있음)를 포함한다. 프로세서(502)는 하나 이상의 CPU 칩들로서 구현될 수 있거나 또는 하나 이상의 주문형 집적 회로(application specific integrated circuit)(ASIC)들의 일부일 수 있다.

[0118] 보조 스토리지(504)는 통상적으로 하나 이상의 디스크 드라이브들 또는 테이프 드라이브들로 구성되며 데이터의 불휘발성 저장을 위해 사용되고 RAM(508)이 모든 작업 데이터를 유지할만큼 충분히 크지 않은 경우 오버플로우 데이터 스토리지 디바이스로서 사용된다. 보조 스토리지(504)는 그러한 프로그램들이 실행을 위해 선택되는 경우 RAM(508)으로 로딩되는 프로그램들을 저장하기 위해 사용될 수 있다. ROM(506)은 프로그램 실행 동안 판독되는 명령어들 및 어쩌면 데이터를 저장하기 위해 사용된다. ROM(506)은 보조 스토리지(504)의 더 큰 메모리 용량과 비교하여 통상적으로 작은 메모리 용량을 갖는 불휘발성 메모리 디바이스이다. RAM(508)은 휘발성 데이터를 저장하기 위해 그리고 어쩌면 명령어들을 저장하기 위해 사용된다. ROM(506) 및 RAM(508) 양자에 대한 액세스는, 통상적으로, 보조 스토리지(504)에 대한 것보다 더 빠르다.

[0119] 본원에서 설명되는 디바이스들은 메모리에 커플링되는 하나 이상의 프로세서들 및 컴퓨터 판독 가능 명령어들을 저장하는 컴퓨터 판독 가능 비일시적 매체들을 포함하도록 구성될 수 있고, 컴퓨터 판독 가능 명령어들을 실행할 때, 도 1 내지 도 4를 참조하여 상기에서 설명되는 방법 단계들 및 동작들을 수행하도록 컴퓨터(500)를 구성할 수 있다. 컴퓨터 판독 가능 비일시적 매체들은 자기 저장 매체들, 광학 저장 매체들, 플래시 매체들, 솔리드 스테이트 저장 매체들을 포함하는, 모든 타입들의 컴퓨터 판독 가능 매체들을 포함한다.

[0120] 또한, 본 개시의 단계들 중 임의의 하나 또는 모두와 관련하여 상기에서 설명되는 바와 같은 동작들 및 프로세싱을 용이하게 하는 하나 이상의 컴퓨터 실행 가능 명령어들을 포함하는 소프트웨어는 본 개시와 부합하는 소비자 및/또는 생산자 도메인들 내의 하나 이상의 서버들 및/또는 하나 이상의 라우터들 및/또는 하나 이상의 디바이스들에 설치되어 함께 판매될 수 있다는 것이 추가로 이해되어야 한다. 대안적으로, 소프트웨어는 본 개시와

부합하는 소비자 및/또는 생산자 도메인들 내의 하나 이상의 서버들 및/또는 하나 이상의 라우터들 및/또는 하나 이상의 디바이스들로 획득되어 로딩될 수 있는데, 예를 들면, 소프트웨어 작성자에 의해 소유되는 서버로부터의 또는 소프트웨어 작성자에 의해 소유되지는 않지만 그러나 사용되는 서버로부터의 것을 포함한 배포 시스템 또는 물리적 매체를 통해 소프트웨어를 획득하는 것을 포함한다. 예를 들면, 소프트웨어는 인터넷을 통한 배포를 위해 서버 상에 저장될 수 있다.

[0121] 또한, 본 개시는, 그 적용이, 설명에서 기술되는 또는 도면들에서 예시되는 컴포넌트들의 구성 및 배열의 세부사항들로 제한되지 않는다는 것이 기술 분야에서 숙련된 자에 의해 이해될 것이다. 본원의 실시예들은 다른 실시예들에 대응할 수 있으며, 다양한 방식들로 실시되거나 또는 실행될 수 있다. 또한, 본원에서 사용되는 문체 및 전문 용어는 설명의 목적을 위한 것이며 제한하는 것으로 간주되어서는 안된다는 것이 이해될 것이다. 본원에서 "포함하는(including)", "포함하는(comprising)", 또는 "갖는(having)" 및 이들의 변형어들의 사용은 그 뒤에 나열되는 아이템들과 그들의 등가물들뿐만 아니라 추가적인 아이템들을 포괄하도록 의도된다. 달리 제한되지 않는 한, 본원에서 용어들 "연결되는", "커플링되는" 및 "장착되는" 및 이들의 변형어들은 광범위하게 사용되며 직접적 및 간접적 연결들, 커플링들, 및 장착들을 포괄한다. 또한, 용어들 "연결되는" 및 "커플링되는" 및 이들의 변형어들은 물리적 또는 기계적 연결들 또는 커플링들로 제한되지는 않는다. 게다가, 위, 아래, 하부, 및 상부와 같은 용어들은 상대적이며, 예시를 돕기 위해 활용되지만, 그러나 제한하는 것은 아니다.

[0122] 예시된 실시예들에 따라 활용되는 예시적인 디바이스들, 시스템들 및 방법들의 컴포넌트들은, 적어도 부분적으로, 디지털 전자 회로부(circuitry)에서, 아날로그 전자 회로부에서, 또는 컴퓨터 하드웨어, 펌웨어, 소프트웨어에서, 또는 그들의 조합들에서 구현될 수 있다. 이들 컴포넌트들은, 프로그래머블 프로세서, 컴퓨터, 또는 다수의 컴퓨터들과 같은 데이터 프로세싱 장치에 의한 실행을 위해, 또는 이들의 동작을 제어하기 위해, 예를 들면, 정보 캐리어에서, 또는 머신 판독 가능 스토리지 디바이스에서 유형적으로 구체화되는 컴퓨터 프로그램, 프로그램 코드 또는 컴퓨터 명령어들과 같은 컴퓨터 프로그램 제품으로서 구현될 수 있다.

[0123] 컴퓨터 프로그램은 컴파일된 또는 해석된 언어들을 포함한 임의의 형태의 프로그래밍 언어로 작성될 수 있으며, 독립 실행형 프로그램으로서 또는 컴퓨팅 환경에서의 사용에 적절한 모듈, 컴포넌트, 서브루틴, 또는 다른 유닛으로서 배치되는 것을 포함하여, 임의의 형태로 배치될 수 있다. 컴퓨터 프로그램은 하나의 사이트에 있는 또는 다수의 사이트들에 걸쳐 분산되며 통신 네트워크에 의해 상호 접속되는 다수의 컴퓨터들 상에서 또는 하나의 컴퓨터 상에서 실행되도록 배치될 수 있다. 또한, 본원에서 설명되는 기술들을 달성하기 위한 함수형 프로그램들, 코드들, 및 코드 세그먼트들은 기술 분야의 숙련된 프로그래머들에 의해 본 개시의 범위 내에 있는 것으로 쉽게 해석될 수 있다. 예시적인 실시예들과 관련되는 방법 단계들은 (예를 들면, 입력 데이터에 대해 동작하고 및/또는 출력을 생성하는 것에 의해) 함수들을 수행하기 위한 컴퓨터 프로그램, 코드 또는 명령어들을 실행하는 하나 이상의 프로그래머블 프로세서들에 의해 수행될 수 있다. 방법 단계들이, 예를 들면, 특수 목적 로직 회로부, 예를 들면, FPGA(field programmable gate array) 또는 ASIC(application-specific integrated circuit)에 의해 또한 수행될 수 있고, 장치가 이들로서 구현될 수 있다.

[0124] 본원에서 개시되는 실시예들과 관련하여 설명되는 다양한 예시적인 논리적 블록들, 모듈들, 및 회로들은 범용 프로세서, 디지털 신호 프로세서(digital signal processor)(DSP), ASIC, FPGA 또는 다른 프로그래머블 로직 디바이스, 이산 게이트 또는 트랜지스터 로직, 이산 하드웨어 컴포넌트들, 또는 본원에서 설명되는 함수들을 수행하도록 설계되는 이들의 임의의 조합을 사용하여 구현되거나 또는 수행될 수 있다. 범용 프로세서는 마이크로프로세서일 수 있지만, 그러나 대안예에서, 프로세서는 임의의 종래의 프로세서, 컨트롤러, 마이크로컨트롤러, 또는 상태 머신일 수 있다. 프로세서는 컴퓨팅 디바이스들의 조합, 예를 들면, DSP와 마이크로프로세서, 복수의 마이크로프로세서들, DSP 코어와 연계한 하나 이상의 마이크로프로세서들, 또는 임의의 다른 그러한 구성의 조합으로서 또한 구현될 수 있다.

[0125] 컴퓨터 프로그램의 실행에 적절한 프로세서들은, 예로서, 범용 및 특수 목적 마이크로프로세서들 둘 모두, 및 임의의 종류의 디지털 컴퓨터의 임의의 하나 이상의 프로세서들을 포함한다. 일반적으로, 프로세서는 리드 온리 메모리 또는 랜덤 액세스 메모리 또는 둘 모두로부터 명령어들 및 데이터를 수신할 것이다. 컴퓨터의 필수 요소들은 명령어들을 실행하기 위한 프로세서 및 명령어들 및 데이터를 저장하기 위한 하나 이상의 메모리 디바이스들이다. 일반적으로, 컴퓨터는 또한 데이터를 저장하기 위한 하나 이상의 대용량 스토리지 디바이스들, 예를 들면, 자기, 광자기 디스크들, 또는 광학 디스크들을 포함할 것이거나, 또는 이들로부터 데이터를 수신하도록 또는 이들로 데이터를 전송하도록, 또는 둘 모두를 하도록 동작 가능하게 커플링될 것이다. 컴퓨터 프로그램 명령어들 및 데이터를 구체화하기에 적절한 정보 캐리어들은, 예로서, 반도체 메모리 디바이스들, 예를 들면, 전기적으로 프로그래밍 가능한 리드 온리 메모리 또는 ROM(electrically programmable read-only memory

or ROM)(EPROM), 전기적으로 소거 가능한 프로그래머블 ROM(electrically erasable programmable ROM)(EEPROM), 플래시 메모리 디바이스들, 및 데이터 저장 디스크들(예를 들면, 자기 디스크들, 내장 하드 디스크들, 또는 착탈식 디스크들, 광자기 디스크들, CD-ROM 및 DVD-ROM 디스크들)을 포함하여, 모든 형태들의 불휘발성 메모리를 포함한다. 프로세서 및 메모리는 특수 목적 로직 회로부에 의해 보완될 수 있거나 또는 특수 목적 로직 회로부에 통합될 수 있다.

[0126] 기술 분야의 숙련된 자들은 정보 및 신호들이 여러 가지 상이한 기술들 및 기법들 중 임의의 것을 사용하여 표현될 수 있다는 것을 이해한다. 예를 들면, 상기의 설명 전체에 걸쳐 참조될 수 있는 데이터, 명령어들, 커맨드들, 정보, 신호들, 비트들, 심볼들, 및 칩들은 전압들, 전류들, 전자기파들, 자기장들 또는 자기 입자들, 광학장들 또는 광학 입자들, 또는 이들의 임의의 조합에 의해 표현될 수 있다.

[0127] 기술 분야의 숙련된 자들은 본원에서 개시되는 실시예들과 관련하여 설명되는 다양한 예시적인 논리적 블록들, 모듈들, 회로들, 및 알고리즘 단계들이 전자 하드웨어, 컴퓨터 소프트웨어, 또는 이들 양자의 조합들로서 구현될 수 있다는 것을 추가로 인식한다. 하드웨어 및 소프트웨어의 이러한 상호 교환성을 명확하게 예시하기 위해, 다양한 예시적인 컴포넌트들, 블록들, 모듈들, 회로들, 및 단계들은 상기에서 일반적으로 그들의 기능성의 관점에서 설명되었다. 그러한 기능성이 하드웨어로서 구현되는지 또는 소프트웨어로서 구현되는지의 여부는 전체 시스템에 부과되는 특정한 애플리케이션 및 설계 제약들에 의존한다. 숙련된 기술자들은 각각의 특정한 애플리케이션에 대한 설명된 기능성을 다양한 방식들로 구현할 수 있지만, 그러나 그러한 구현 결정들은 본 개시의 범위로부터의 일탈을 야기하는 것으로 해석되어서는 안된다. 소프트웨어 모듈은 랜덤 액세스 메모리(RAM), 플래시 메모리, ROM, EPROM, EEPROM, 레지스터들, 하드 디스크, 착탈식 디스크, CD-ROM, 또는 기술 분야에서 공지되어 있는 임의의 다른 형태의 저장 매체에서 존재할 수 있다. 프로세서가 저장 매체로부터 정보를 판독하고, 저장 매체에 정보를 기입할 수 있도록, 예시적인 저장 매체는 프로세서에 커플링된다. 대안예에서, 저장 매체는 프로세서와 일체형일 수 있다. 다시 말하면, 프로세서 및 저장 매체는 집적 회로에서 존재할 수 있거나 또는 개별 컴포넌트들로서 구현될 수 있다.

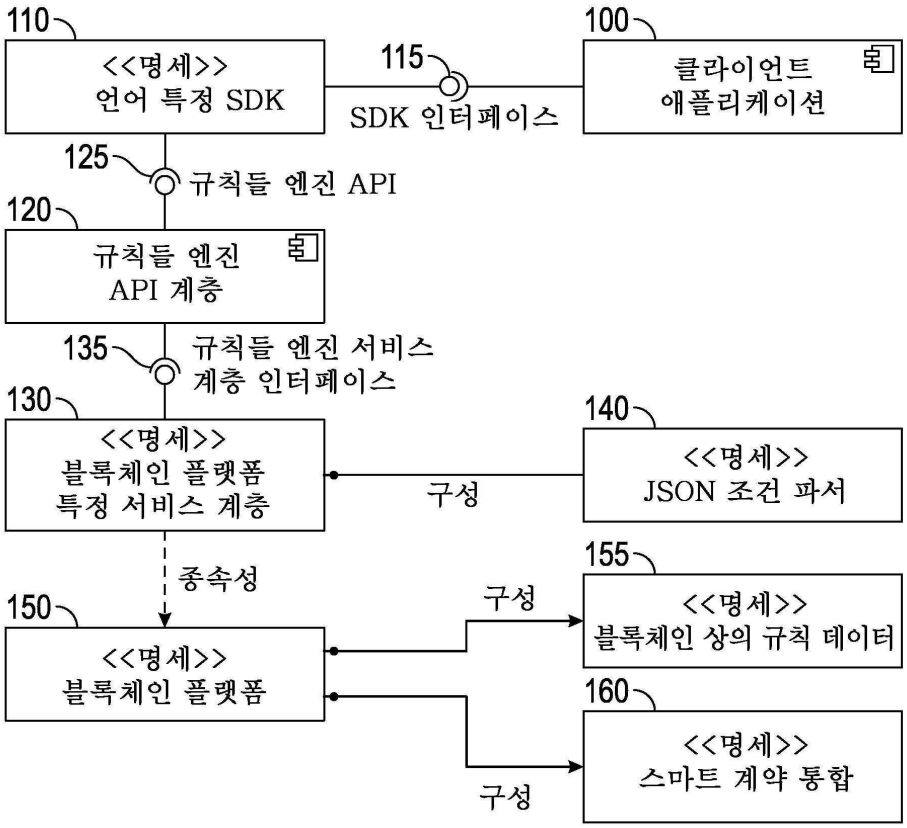
[0128] 본원에서 사용될 때, "머신 판독 가능 매체"는 명령어들 및 데이터를 일시적으로 또는 영구적으로 저장할 수 있는 디바이스를 의미하며, 랜덤 액세스 메모리(RAM), 리드 온리 메모리(ROM), 버퍼 메모리, 플래시 메모리, 광학 매체들, 자기 매체들, 캐시 메모리, 다른 타입들의 스토리지(예를 들면, 소거 가능한 프로그래머블 리드 온리 메모리(EEPROM)), 및/또는 이들의 임의의 적절한 조합을 포함할 수 있지만, 그러나 이들로 제한되지는 않는다. 용어 "머신 판독 가능 매체"는 프로세서 명령어들을 저장할 수 있는 단일의 매체 또는 다수의 매체들(예를 들면, 중앙 집중식 또는 분산형 데이터베이스, 또는 관련된 캐시들 및 서버들)을 포함하는 것으로 간주되어야 한다. 용어 "머신 판독 가능 매체"는, 하나 이상의 프로세서들에 의한 실행을 위한 명령어들을 저장할 수 있는 임의의 매체, 또는 다수의 매체들의 조합을 포함하는 것으로 또한 간주되어야 하며, 그 결과, 명령어들은, 하나 이상의 프로세서들에 의해 실행될 때, 하나 이상의 프로세서들로 하여금 본원에서 설명되는 방법론들 중 임의의 하나 이상을 수행하게 한다. 따라서, "머신 판독 가능 매체"는 단일의 스토리지 장치 또는 디바이스뿐만 아니라, 다수의 스토리지 장치 또는 디바이스들을 포함하는 "클라우드 기반의" 스토리지 시스템들 또는 스토리지 네트워크들을 지칭한다. 용어 "머신 판독 가능 매체"는 본원에서 사용될 때 신호들 그 자체를 배제한다.

[0129] 상기에서 제시된 설명 및 도면들은 단지 예로서 의도되며, 첨부된 청구항들에서 기술되는 바를 제외하면 예시적인 실시예들을 어떤 식으로든 제한하도록 의도되는 것은 아니다. 상기에서 설명된 다양한 예시적인 실시예들의 다양한 요소들의 다양한 기술적 양태들은 수많은 다른 방식들로 결합될 수 있되, 그들 모두는 본 개시의 범위 내에 있는 것으로 간주된다는 것을 유의한다.

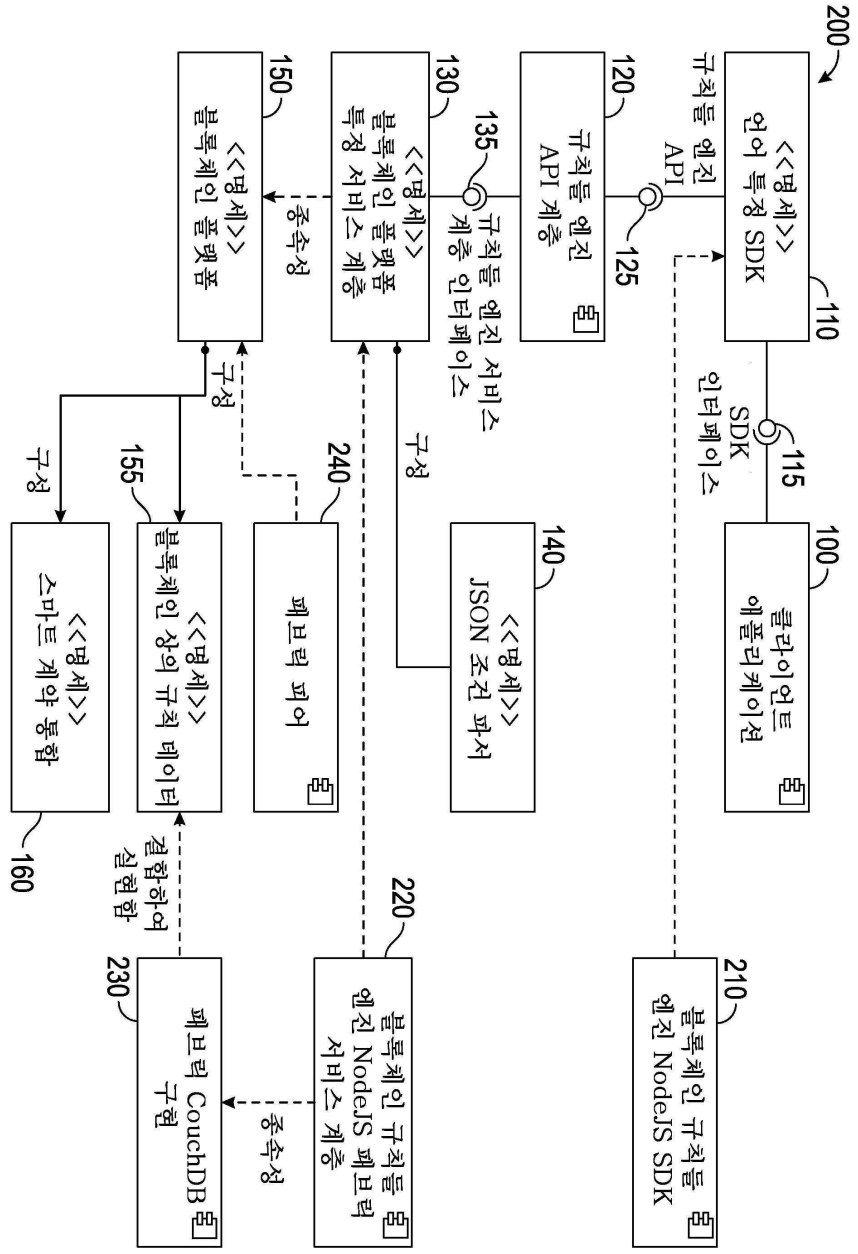
[0130] 따라서, 예시적인 실시예들이 예시적 목적들을 위해 개시되었지만, 기술 분야의 숙련된 자들은 다양한 수정예들, 추가예들, 및 대체예들이 가능하다는 것을 인식할 것이다. 따라서, 본 개시는 상기에서 설명된 실시예들로 제한되는 것이 아니라, 첨부된 청구항들의 전체 범위의 등가물들과 함께, 첨부된 청구항들의 범위 내에서 수정될 수 있다.

도면

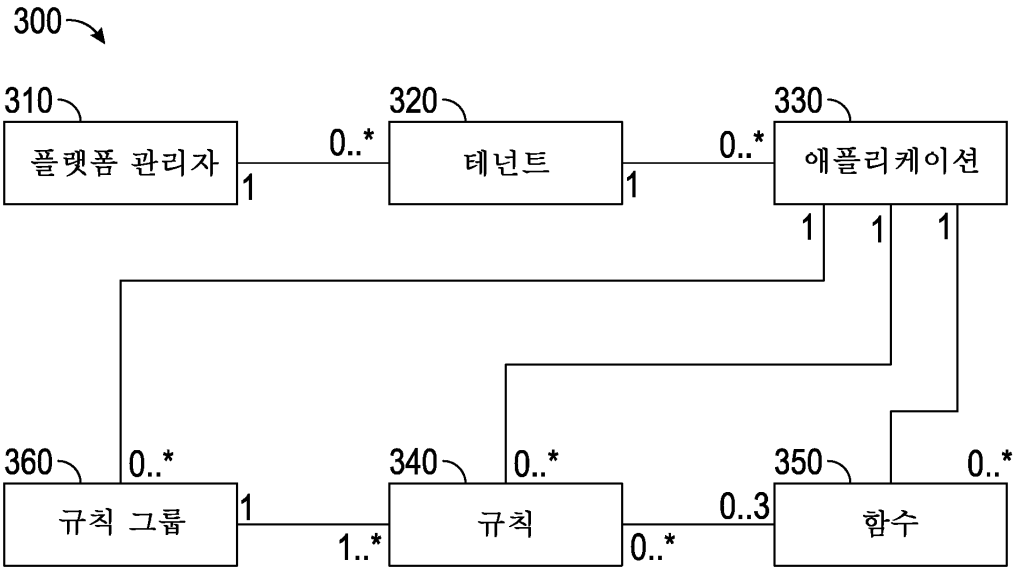
도면1


110
<<명세>>
언어 특정 SDK
115
SDK 인터페이스
100
클라이언트
애플리케이션
125
규칙들 엔진 API
120
규칙들 엔진
API 계층
135
규칙들 엔진 서비스
계층 인터페이스
130
<<명세>>
블록체인 플랫폼
특정 서비스 계층
구성
140
<<명세>>
JSON 조건 파서
종속성
150
<<명세>>
블록체인 플랫폼
구성
155
<<명세>>
블록체인 상의 규칙 데이터
구성
160
<<명세>>
스마트 계약 통합

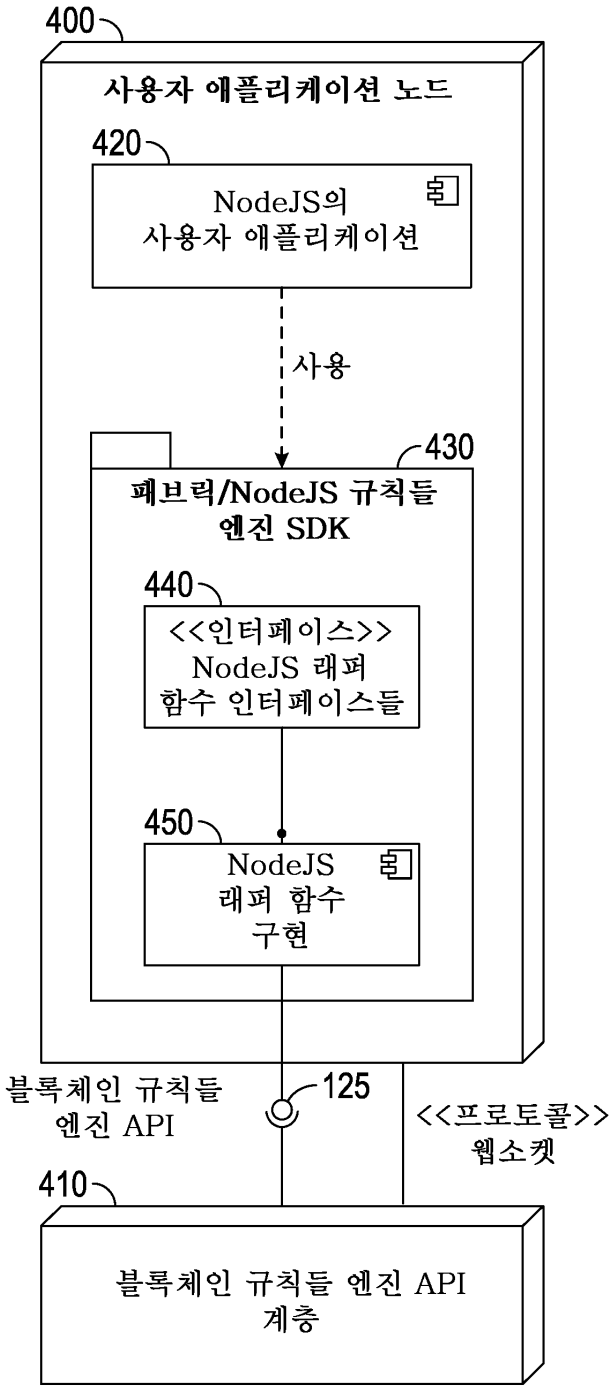
도면2


200
<<명세>>
언어 특정 SDK
110
규칙들 엔진
API
125
120
규칙들 엔진
API 계층
135
규칙들 엔진 서비스
계층 인터페이스
130
<<명세>>
블록체인 플랫폼
특정 서비스 계층
150
<<명세>>
블록체인 플랫폼
종속성
SDK
인터페이스
115
100
클라이언트
애플리케이션
210
블록체인 규칙들
엔진 NodeJS SDK
140
<<명세>>
JSON 조건 파서
구성
240
페브릭 피어
155
<<명세>>
블록체인 상의 규칙 데이터
구성
<<명세>>
스마트 계약 통합
160
결합하여
실현함
220
블록체인 규칙들
엔진 NodeJS 페브릭
서비스 계층
종속성
230
페브릭 CouchDB
구현

도면3



도면4


400
사용자 애플리케이션 노드
420
NodeJS의
사용자 애플리케이션
사용
430
페브릭/NodeJS 규칙들
엔진 SDK
440
<<인터페이스>>
NodeJS 래퍼
함수 인터페이스들
450
NodeJS
래퍼 함수
구현
블록체인 규칙들
엔진 API
125
<<프로토콜>>
웹소켓
410
블록체인 규칙들 엔진 API
계층

도면5

