详解agent-js

从原理到应用和扩展

michael@AstroX.Network

目录

主要内容

快速了解ICP

快速了解ICP

它是啥，为啥我们要在上面做开发

- 全称Internet Computer Protocal，https://smartcontracts.org/
- 它有两种代币，ICP（用于价值交换）和Cycles（用于合约消耗，类似Gas，合约部署人支付，成本超级低）
- Canister，轻量级WASM容器，运行智能合约，还可以放置前端资源，即全栈链上开发，很多人认为IC就像区块链版本的亚马逊
- IC采用了多项密码学创新，包括NIDKG，BLS阈值签名，ECDSA阈值签名，finalty 2秒内，通过message完成与wasm合约的交互。
- 除了金融属性的DApp之外，生态还构建了如社交网络，加密网盘，加密笔记等具有IC特色的项目
- IC的整个开发范式与其他区块链有很大的不同，也许概念需要切换，但是上手体验相当的好。

前言和自我介绍

前言和自我介绍

为什么要了解agent-js，以及本人

- agent-js是IC上的前端依赖库，每一个DApp都会用到它，agent-js包含了许多的底层的编码和基础功能，是IC客户端体系的一个聚合，了解它，你会更加了解如何与IC的区块链交互，了解与Canister的交互，了解通信的编码和安全机制。Agent-JS在IC生态中的地位，相当于Web3.js或者ether.js在以太坊生态中的地位，简单说无论是Metamask，还是DApp用到的合约处理，都需要用到它。

- 关于本人，AstroX Network联合创始人，为多条公链定制和设计客户端SDK，深度研究Web3.js，ether.js，polkadot.js以及移动端和跨平台的实现。在IC上，是agent_dart的作者和主要维护者，理解并实现所有agent-js包含的功能，并做出了新的扩展包，为IC移动和跨平台生态打下基础。

- 前方预警：本讲座内容含有不少的计算机通信和底层知识，未必适合所有人的观看和收听。同时，本次讲座并非告诉您如何做一个漂亮的前端网站，也并不包含构建DApp的实战教程，更不包含任何的投资建议。

agent-js的功能概览和作用

agent-js的功能概览和作用

它诞生的目标，packages的组成和应用场景

- agent-js来自于IC的SDK，与dfx的定位稍有不同，agent-js面向DApp开发者，提供了一系列使用的工具集，包括了通信编码，椭圆曲线，Candid解析，Identity系统和http基础库。agent-js可同时面向浏览器和node-js进行开发。

- agent-js，顾名思义，可以理解为通信代理人，它将javascript的请求根据一定的编码规则进行了协议层的封装，然后发送至IC的协议层，然后根据响应，将结果进行解码返回至客户端。

- agent-js广泛应用于IC DApp和开放服务或工具的开发，在身份场景下，有Internet Identity，在钱包的场景下，有Plug或者Stoic Wallet等等。

agent-js的功能概览和作用

它诞生的目标，packages的组成和应用场景

- 三大板块：1）X509和CBOR的实现（面向IC通信协议）2）Identity体系和椭圆曲线的应用。3）Candid解析和Http库使用。
- 分包看，https://github.com/dfinity/agent-js/tree/main/packages：
  - agent：基础包，X509和CBOR，Http库，Identity抽象
  - auth-client：用于与Internet Identity通信的客户端SDK，在DApp集成。
  - authentication：被auth-client集成，无需特别关心。
  - candid：Candid的编码和解析库，用于合约调用和结果解析。
  - identity-ledgerhq：用于硬件钱包Ledger的身份创建和通信包，可无需特别关心。
  - identity：identity体系的实现，包含Delegation，Ed25519和Secp256k1的应用，以及webauthn的扩展类
  - principal：principal基础包，principal和publickey互转工具，常用

X509和CBOR

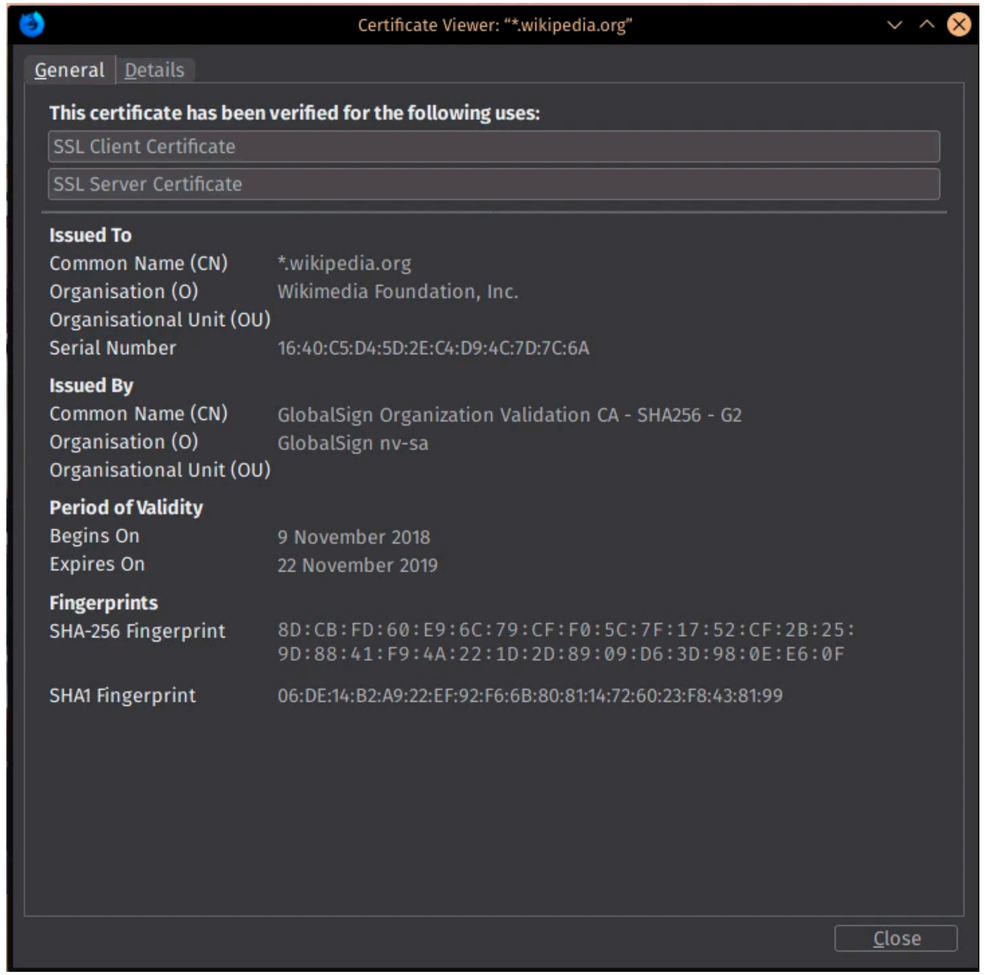
X509和CBOR

什么是X509

- https://en.wikipedia.org/wiki/X.509
- X.509是一个Public Key Certificates的格式标准，TLS/SSL使用它，TLS/SSL是HTTPS的基础所以HTTPS也使用它。而所谓Public Key Certificates又被称为 **Digital Certificate** 或 **Identity Certificate**
- 一个X.509 Certificate包含一个Public Key和一个身份信息，它要么是被CA签发的要么是自签发的。

X509和CBOR

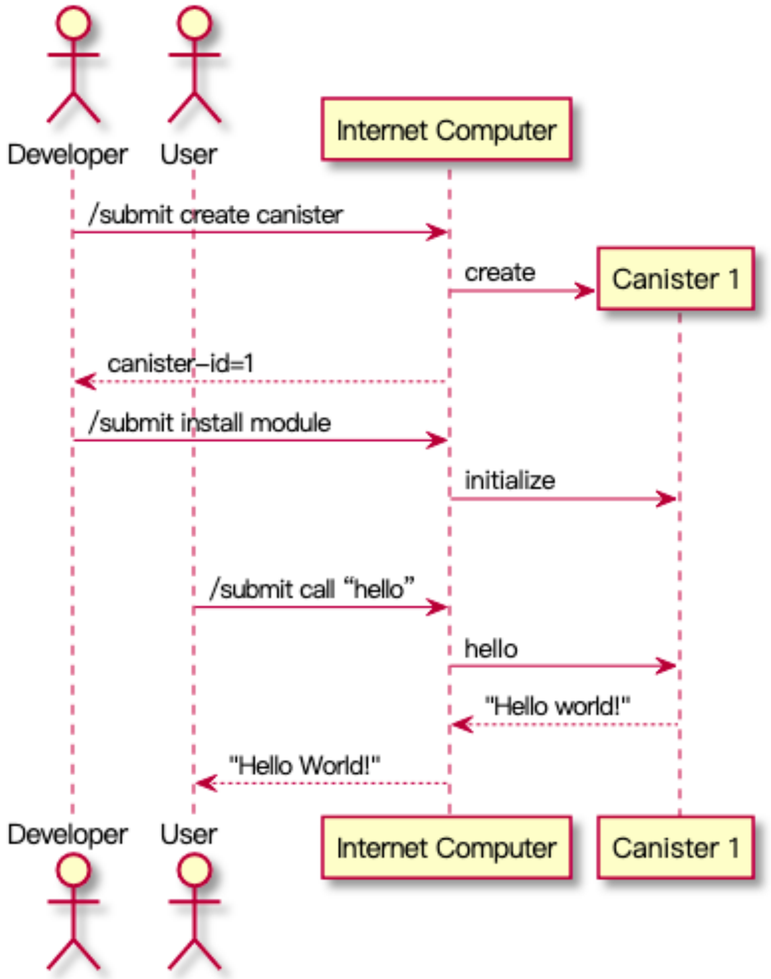
什么是X509



- Chain key 生成X509证书
- Https 通信过程使用该证书进行认证和加密
- 提供前端浏览器显示加载资源
- 前端与Canister之间的Post call之间使用该证书进行加密

X509和CBOR

通信过程



- 用户/Canister，对于系统而言，都是Principal ID
- 外部通信：Query Call 和Update Call
- 内部通信：Sender 和 Receiver发送消息
- WebAssembly和系统API方法开放给外部，通过HTTPS接口调用

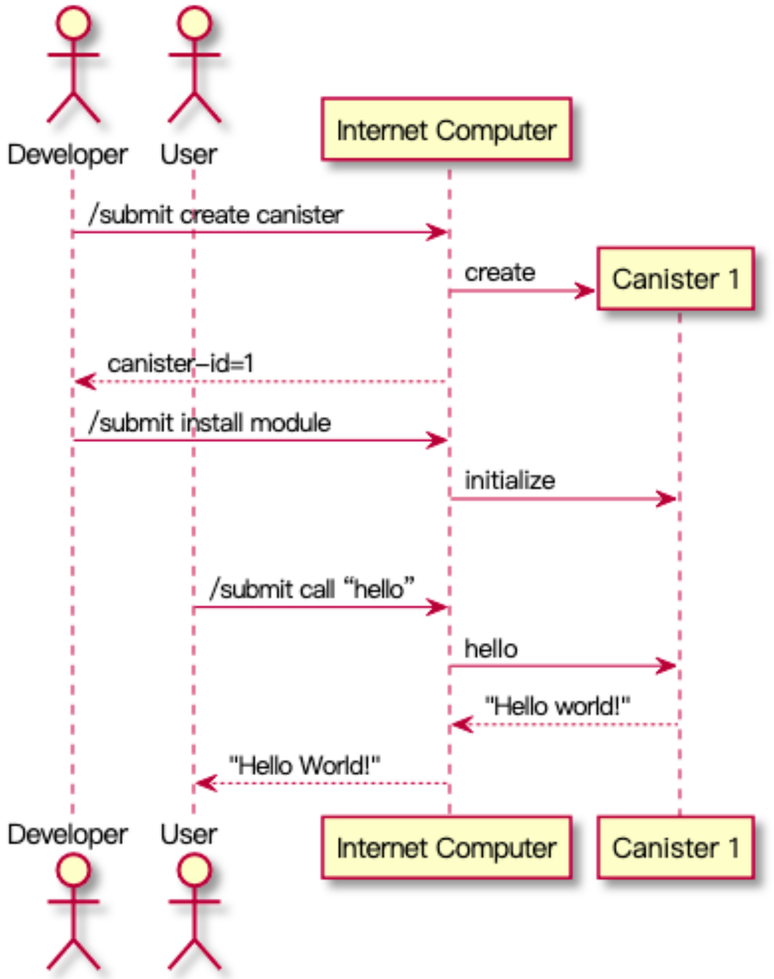
X509和CBOR

什么是CBOR

- https://en.wikipedia.org/wiki/CBOR
- 简明二进制对象展现（CBOR，Concise Binary Object Representation）是一种提供良好压缩性，扩展性强，不需要进行版本协商的二进制数据交换形式。RFC 7049定义了详细的CBOR格式与说明。
- 简单的比喻：二进制版本的JSON。
- 8种主类型（Major Type），如无符号整数，负整数，Byte String，字符串，Array，Map，Tag（扩展），特殊类型
- 编码（encode）可以节省大量空间，解码（decode）速度很快，跨平台。比Protobuf轻量，可读性较好。不需要提前写proto，内存读写。

X509和CBOR

为什么应用CBOR



- 所有的HTTPS Call都会经过CBOR处理。
- 并且所有的Public Key都是DER-wrapped的COSE key，其中DER-Encoded-ED25519是 RFC 8410，DER-Encoded-ECDSA（对于Secp256k1）是RFC 5480。
- Internet Identity的核心原理，Canister签名与Webauthn遵循同一套编码规则(Self-Describe CBOR)，让浏览器可以解析并使用正确的签名

读代码时间

Identity体系和椭圆曲线

Identity体系和椭圆曲线

IC的Identity定义

- Identity在IC中，是一个认证单元，类型有Anonymous Identity，Sign Identity，Delegation Identity。每种Identity都可以获取Principal ID，以及在Http请求时进行可能的拦截处理。
- Anonymous Identity包含一个默认的Principal ID，不对请求进行签名
- Sign Identity的Principal ID由其公钥DER化计算得出，需要对HTTP请求进行签名（CBOR化）
- Delegation Identity是Chain Key技术衍生的特性，具备CBOR化的能力，由一个Sign Identity签名，它还可以持有多个公钥组合成的Delegation Chain，是实现Delegation的重要方式。

Identity体系和椭圆曲线

IC的Identity定义

- 目前IC支持ED25519和ECDSA的签名校验，Identity也就包含ED2559KeyIdentity和SECP256k1KeyIdentity，不同曲线下的Identity，Principal ID不同，签名的结果也不一样。需要特别注意。

- 为什么要做Identity？因为通信对象都需要Principal ID，而发送消息需要在X509下保持CBOR化的签名，最终在IC中得到校验。每一次你在DSCVR发帖子，都是一次Update Call，因为你持有Identity，帖子因你而签名。

- 而为什么要做Delegation签名而不使用原本的Identity签名？因为：1）访问不同的DApp和认证的时候需要保持隐私和匿名化。2）Delegation兼具了安全性和时效性，方便了用户访问DApp完成与Canister的交互。

Identity体系和椭圆曲线

IC的Identity定义

- 为什么不直接做钱包就好了？有钱包应用返回Principal和签名，如PlugWallet。钱包更关注的是资产操作和交易，而Identity更关注的是认证（当然也可以发送交易），尤其是实现WebAuthentication之后，用户就无需手动托管私钥了。

读代码时间

网络请求库和Candid解析

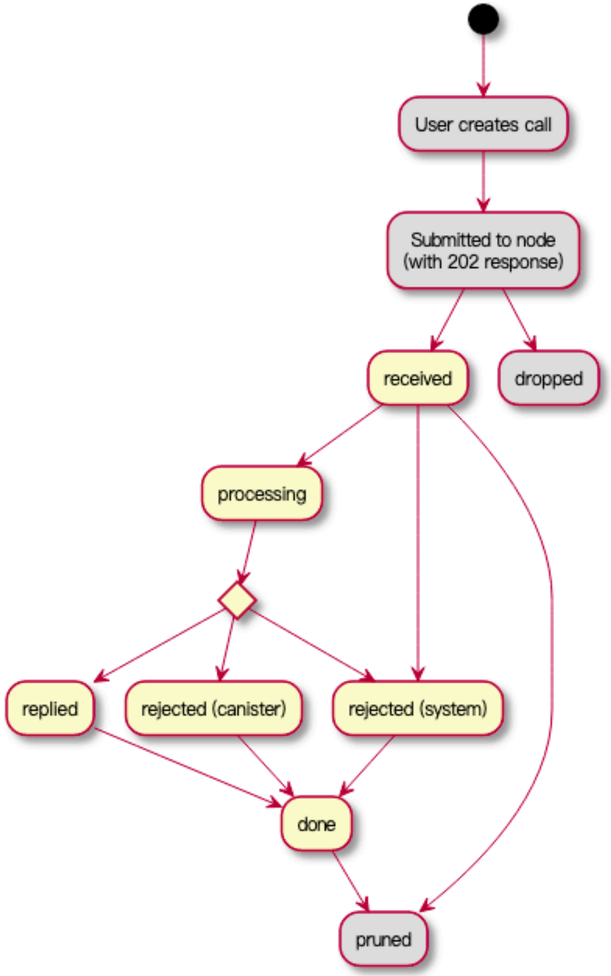
网络请求库和Candid解析

http 包

- 对Fetch进行封装，兼容nodejs和浏览器环境
- 生成httpAgent，作为身份入参入口。
- 区分update call和query，在发送请求之前调用相关的签名和编码
- 同时调用webassembly版本的bls完成verify

网络请求库和Candid解析

http 包



- https://smartcontracts.org/docs/interface-spec/index.html#http-interface
- /api/v2/canister/{canister_id}
  - /read_state: 读取Canister 在 IC状态
  - /call：调用Canister update方法
  - /query：调用Canister query方法
- /status：拉取IC当前状态

网络请求库和Candid解析

Candid 包

- https://smartcontracts.org/docs/candid-guide/candid-concepts.html
- Candid是WASM接口描述文件，包含了类型Type和对外暴露的方法签名(function signature)
- Candid文件可以通过didc包生成不同语言版本的接口类和接口文件，在dfx中，如果用motoko开发，这个过程会自动完成，如果用rust开发，这个过程 (可能)需要手动完成。
- Candid需要在JS侧完成编解码的解析（CBOR），并且生成最后的ActorClass，当开发者调用类的成员方法的时候，即可完成canister方法的调用。这个过程类比ether.js或者web3.js对于solidity的编解码和方法生成。

读代码时间

agent-js缺失的内容

agent-js缺失的内容

需要一些补完

- ICP钱包管理，参考stoic wallet和plug wallet，现在还有nns-js可以用
- Cycles管理，目前缺失
- Rosetta-client，交易所的朋友需要了解（如果你用JS来请求）
- NFT类，衍生Token类，域名解析类，通用登录/身份类，社区还有很长的路要走，很多标准需要共识。

有关AstroX Network

AstroX

项目和招聘

- AstroX是IC上的基础设施团队，现有身份类产品ME，和移动端技术栈agent_dart和链上小程序容器Tachikoma，国内第一个DFINITY Grant winner
- 我们正在招募，Rust开发（Canister和节点方向），Flutter和移动开发（App和SDK方向），Web开发（DApp方向），产品和社区运营（面向全球）。
- 欢迎开发者加入我们的DevDAO，共同建立生态
- ME测试版： beta.astrox.me
- https://github.com/AstroxNetwork
- https://discord.gg/D5Z57Z2gxh

一起BUIDL吧，谢谢大家！