Uniswap v2 核心

整理人：嚴佳華

原文：Uniswap v2 Core
作者：
Hayden Adams - hayden@uniswap.org
Noah Zinsmeister - noah@uniswap.org
Dan Robinson - dan@paradigm.xyz

摘要

這篇技術白皮書解釋了 **Uniswap v2 核心合約**背後的設計決策。

內容覆蓋了合約的新特性，包括任意 **ERC20 代幣**的交易對，強化的**價格預言機制（price oracle）**，允許交易者先接收資產並使用，隨後在轉賬中支付的**快速互換（flash swaps）**功能，在未來可以開啟的**協議手續費（protocol fee）**為了減少受攻擊面，重新設計了合約的架構。

本白皮書描述了 **Uniswap v2 核心合約**的運行機制，包括儲存流動性提供者資金的交易對合約，和用於**實例化交易對合約的工廠合約。**

1 介紹

Uniswap v1 是一個以太坊區塊鏈上的智能合約系統，基於常數乘積公式實現了自動流動性協議。每個 Uniswap v1 交易對在資金池裡儲存兩種資產，並位這兩種資產提供流動性，維持這兩種資產的乘積不減少。交易者支付 30bp（譯者註：0.3%）的交易手續費給流動性提供者。合約是不能升級的。

Uniswap v2 基於同樣的公式實現，添加了幾個特性。最重要的一個是支持 ERC20 代幣/ERC20 代幣交易對，而不像 v1 只支持 ERC20 代幣/ETH 交易對。除此以外，它也強化了價格預言，在每個區塊開始處累積兩個資產的相對價格。這允許其他以太坊合約計算兩個資產在任意時間區間上的 TWAP 價格。最後，它啟用了快速互換（flash swaps）功能，用戶可以自由地接收資產並把他們用在鏈上的其他地方，只要在轉賬的最後支付（或返還）即可。

雖然合約通常來說不可升級，但是有一個私鑰能夠更新一個變量，打開鏈上 5bp 的手續費，這個變量一開始是關閉的，可以在未來打開，打開之後流動性提供者在每筆交易上賺取 25bp，而不是 30bp。

Uniswap v2 修復了一些 v1 中的小問題，也在實現中重新設計了架構，減少了 Uniswap 的受攻擊面，通過把邏輯最小化放在核心合約（core contract）中，這個核心合約維持了流動性提供者的資金，從而使系統變得更容易升級。

這篇論文描述了**核心合約（core contract）**的運行機制，以及用來實例化核心合約的**工廠合約（factory contract）**。實際上使用 Uniswap v2 需要通過一個**"路由"合約（router contract）**，計算交易/存款數額並轉移資金到**交易對合約（pair contract）**。

譯者註：
合約代碼：https://github.com/Uniswap/uniswap-v2-core
Uniswap v1 白皮書：
https://hackmd.io/@HaydenAdams/HJ9jLsfTz#%F0%9F%A6%84-Uniswap-Whitepaper

2 新特性

2.1 ERC20 交易對

Uniswap v1 使用 ETH 作為過渡貨幣（bridge currency），每個交易對都包含 ETH。這使得路由更簡單（ABC 和 XYZ 之間的每筆交易都通過 ETH/ABC 交易對和 ETH/XYZ 交易對進行），並且減少了流動性的分散。

但是，這個規則給流動性提供者增加了可觀的成本。所有流動資金提供者都有 ETH 的敞口，並且由於其他資產相對於 ETH 的價格變化而遭受**無常損失（impermanent loss）**。當兩個資產 ABC 和 XYZ 相關時

（例如，如果它們都是錨定美元的穩定幣），則在 Uniswap 交易對 ABC/XYZ 上的流動性提供者遭受的無常損失會比 ABC/ETH 或 XYZ/ETH 交易對少。

使用 ETH 作為強制性過渡貨幣也會給交易者帶來成本。交易者必須支付的費用是直接購買 ABC/XYZ 交易對的費用的兩倍，因此他們遭受兩次滑點。

Uniswap v2 允許流動性提供商為任意兩個 ERC20 創建對合約。

任意 ERC20 之間的交易對數量的激增可能會使尋找交易特定貨幣對的最佳路徑變得更加困難，但是路由可以在更高的層面上處理（在鏈下處理或通過鏈上路由器或聚合器）。

2.2 價格預言(Price oracle)

Uniswap 提供的 t 時刻的邊際價格（marginal price，不包括手續費）可以用資產 a 的儲備除以資產 b 的儲備來計算。

$$p_t = \frac{r_t^a}{r_t^b}$$

如果這個價格偏離（超出手續費足夠的數額），套利者會和 Uniswap 進行交易使價格回歸正常，所以 Uniswap 提供的價格趨向於追踪資產在相關市場的價格。這意味著它可以被用作一個近似的價格預言。

但是用 Uniswap v1 作為鏈上價格預言是不安全的，因為它非常容易被操縱。假設一個其他的合約用當前的 ETH-DAI 的價格交割某個衍生品，操縱價格的攻擊者會從 ETH-DAI 交易對買入 ETH，在衍生品上觸發結算（引起失真的結算價），然後再把 ETH 賣給交易對，用真實價

格反向交易。這甚至可以用一個原子的轉賬來完成，或者通過一個控制區塊內交易（transaction）順序的礦工。

Uniswap v2 改進了預言的功能，通過測算和記錄每個區塊第一筆交易之前的價格（也就是前一個區塊最後的價格）。這個價格比一個區塊內的價格更難被操縱。如果攻擊者提交了一筆交易（transaction）嘗試在區塊末尾處操縱價格，其他的套利者可以提交另一個交易（transaction）立即進行反向交易。某個礦工（或有足夠 gas 填滿整個區塊的攻擊者）可以操縱區塊末尾處的價格，除非他們可以挖出下一個區塊，否則他們他們沒有特殊的的套利優勢。

具體來說，Uniswap v2 追踪每個和合約交互的區塊開始處的價格的累加和，來累加價格。每個價格用距離上一個更新價格的區塊的時間進行加權，根據區塊時間戳。這意思是累加器的值在任意時間（更新後）的值等於合約歷史上每秒的現貨價格的和。

$$a_t = \sum_{i=1}^{t} p_i$$

要計算從時間 t1 到 t2 的時間加權平均價（譯者註：TWAP 價格），一個外部調用者可以檢查 t1 和 t2 時間的累加器的值，將後值減去前值，再除以期間經過的秒數。（注意，合約本身並不記錄歷史累加值，調用者必須在時間段開始處調用合約來讀取和儲存這個值。）

$$p_{t_1,t_2} = \frac{\sum_{i=t_1}^{t2} p_i}{t_2 - t_1} = \frac{\sum_{i=1}^{t_2} p_i - \sum_{i=1}^{t_1} p_i}{t_2 - t_1} = \frac{a_{t_2} - a_{t_1}}{t_2 - t_1}$$

預言的用戶可以選擇這個區間的起始和結束時間。選擇更長的區間可以讓攻擊者操縱價格的成本更高，雖然這會導致價格變化滞後。

一個複雜之處：我們應該用資產 B 來計算資產 A 的價格還是用資產 A 來計算資產 B 的價格？雖然用 B 計算的 A 的現貨價格總是用 A 計算的 B 的現貨價格的倒數，但是在某一時間段內用 B 計算的 A 的均價不等於用 A 計算的 B 的均價的倒數。舉個例子，如果 USD/ETH 價格在區塊 1 中是 100，在區塊 2 中是 300，USD/ETH 的均價是 200 USD/ETH，但是 ETH/USD 的均價是 1/150 ETH/USD（譯者註：他們的均價不是倒數關係）。因為合約無法知道用戶想要用哪個資產作為賬戶單位，所以 Uniswap v2 會同時追踪兩個價格。

另一個複雜之處：有沒有可能某個人發送資產給交易對合約，用來改變它的餘額和邊際價格，但又不和它交互，因此不會觸發價格更新。如果合約簡單的檢查它自己的餘額然後更新預言，攻擊者可以在某一個區塊中第一次調用合約之前向合約發送資產，從而操縱價格。如果上一次交易的區塊是 X 秒以前，合約會在累加之前錯誤地把新價格乘以 X，即使沒有用戶用那個價格進行過交易。為了防止這個問題，核心合約在每次交互後緩存它的資金儲備，用緩存的資金儲備更新價格預言而不用當前資金儲備。除了保護價格預言被操縱，這個改動也啟用 3.2 節中描述的合約重新架構。

2.2.1 精度

Solidity 沒有一等的非整型數的數據結構的支持，Uniswap v2 用簡單的二進制定點數格式編碼和控制價格。具體來說，某一時間的價格存儲為 UQ112.112 格式，意思是在小數點的任意一邊都有 112 位精度，無符號。這些數字的範圍是 $[0, 2^{112} - 1]$，精度是 1/2^112。

選擇 UQ112.112 格式是由於實用的原因，因為這些數可以被存在 uint224 中，在 256 位中剩餘的 32 位空餘。儲備資金各自存在 uint112 中，剩餘 32 位存儲空間。這些空閒空間被用於之前描述的累加過程。具體來說，儲備資金和時間戳存儲在至少有一個交易的最近的區塊中，mod 232（譯者註：取餘數）之後可以存進 32 位空間。另外，雖然任意時間的價格（UQ112.112 數字）確保可以儲存進 224 位中，但某段時間的累加值確保能存下。存儲 A/B 和 B/A 累加價格空間尾部附加的 32 位用來存連續累加溢出的位。這樣設計意味著價格預言只在每一個區塊的第一次交易中增加了 3 次 SSTORE 操作（目前花費 15000gas）。

主要的缺點是 32 位不足以儲存時間戳並確保不溢出。事實上 32 位 Unix 時間戳的溢出日期是 2106 年 7 月 2 日。為了確保系統在這個日

期後以及每　　秒的間隔能夠繼續運行，預言簡單要求每個間隔至少檢查一次價格（大約 136 年一次）。這是由於累加的核心函數（mod 取餘運算）是溢出安全的，意思是預言用溢出算法計算差值，跨越溢出區間的交易可以被合理計算。

2.3 快速互換(Flash Swaps)

Uniswap v1 中，用戶用 XYZ 買 ABC 需要發送 XYZ 到合約，然後才能收到 ABC。如果用戶需要 ABC 為了獲取他們支付的 XYZ，這種方式不方便的。舉個例子，用戶可能在其他合約中用 ABC 買 XYZ，為了對沖 Uniswap 上的價格，或者他們可能在 Maker 或 Compound 上賣出抵押品平倉然後返還給 Uniswap。

Uniswap v2 添加了一個新的特性，允許用戶在支付前接收和使用資產，只要他們在同一個原子的轉賬中完成支付。swap 函數調用一個可選的用戶指定的回調合約，在這之間轉出用戶請求的代幣並且強制確保不變。一旦回調完成，合約檢查新余額並且確保滿足不變（在經過支付手續費調整後）。如果合約沒有足夠的資金，它會回滾整個交易。

用戶也可以用同樣的代幣返還給 Uniswap 資金池而不完成互換。這高效地讓任何人從 Uniswap 資金池中快速借取任何資產（Uniswap 收取同樣的千分之三的交易手續費）。

2.4 協議手續費(Protocol fee)

Uniswap v2 包括 0.05%協議手續費，可以打開或關閉，如果打開，手續費會被發送給工廠合約中指定的 feeTo 地址。

初始時，feeTo 沒有被設定，不收手續費。一個預先指定的地址 feeToSetter 可以在 Uniswap v2 工廠合約上調用 setFeeTo 函數，設置

feeTo 地址。feeToSetter 也可以自己調用 setFeeToSetter 修改 feeToSetter 地址。

如果 feeTo 地址被設置，協議會收取 5pb 的手續費，從流動性提供者的 30bp 手續費中抽取 1/6。交易者將在所有交易上支付 0.3%手續費，83.3%的手續費給流動性提供者，16.6 手續費給 feeTo 地址。

總共收集的手續費可以用自從上次手續費收集以來（譯者註：k 是常數乘積，可以看 v1 白皮書）的增長來計算（也就是）。以下公式給出了 t1 和 t2 之間的累加手續費佔 t2 時間資金池中流動性的百分比：

$$f_{1,2} = 1 - \frac{\sqrt{k_1}}{\sqrt{k_2}} \tag{4}$$

如果fee在時間t1前啟用，feeTo地址應該獲得1/6的t1到t2時間段內的累加手續費。因此，我們要鑄造新的流動性代幣給feeTo地址 $\phi \cdot f_{1,2}$ ，其中 $\phi = \frac{1}{6}$ 。

我們要選擇一個sm滿足以下關係，其中s1是t1時刻的流通份額（outstanding shares）總量：

$$\frac{s_m}{s_m + s_1} = \phi \cdot f_{1,2} \tag{5}$$

經過變換，將替換為後，解得

$$s_m = \frac{\sqrt{k_2} - \sqrt{k_1}}{(\frac{1}{\phi} - 1) \cdot \sqrt{k_2} + \sqrt{k_1}} \cdot s_1 \tag{6}$$

設，得到以下公式

$$s_m = \frac{\sqrt{k_2} - \sqrt{k_1}}{5 \cdot \sqrt{k_2} + \sqrt{k_1}} \cdot s_1 \tag{7}$$

假設初始存款人存了100DAI和1ETH在交易對中，收到10份額。一段時間後（如果沒有其他存款人參與）他們把錢轉出，這時交易對有96DAI和1.5ETH，用上面得公式可以得出：

$$s_m = \frac{\sqrt{1.5 \cdot 96} - \sqrt{1 \cdot 100}}{5 \cdot \sqrt{1.5 \cdot 96} + \sqrt{1 \cdot 100}} \cdot 10 \approx 0.0286 \tag{8}$$

2.5 資金池份額的元交易(Meta transactions for pool shares)

Uniswap v2 交易對鑄造得資金池份額原生支持元轉賬。這意思是用戶可以用簽名授權一個他們的資金池份額的轉賬，而不用從他們的地址進行鏈上轉賬。任何人都可以調用 permit 函數來以用戶的名義發送簽名，支付 gas 手續費並在同一個轉賬中執行其他操作。

3 其他修改

3.1 Solidity

Uniswap v1 是用 Vyper 實現的，一種類似於 Python 的智能合約預言。Uniswap v2 是用更廣泛使用的 Solidity 實現的，因為它在開發的時候需要一些 Vyper 中不可用的功能（比如翻譯非標準 ERC20 代幣的返回值，通過內聯彙編訪問新的字節碼 chanid）。

3.2 合約重新架構

Uniswap v2 在設計時優先考慮的一個方面是最小化受攻擊表面積和核心交易對合約的複雜度，交易對合約儲存了流動性提供者的資產。這個合約中的任何 bug 都可能是災難性的，因為數百萬美元的流動性可能會被竊取或凍結。

在評估核心合約的安全性時，最重要的問題時它是否保護流動性提供者以防他們的資產被竊取或凍結。除了基本的允許資產在資金池中互換的功能，任何支持或保護交易者的特性，都可以在路由合約中處理。

事實上，甚至一部分的互換功能也可以被抽出放到路由合約中。之前提到過，Uniswap v2 儲存有每種資產最近一次記錄的餘額（為了防止預言機制被操縱）。新的架構利用了這一點來進一步簡化 Uniswap v1 合約。

Uniswap v2 中，賣方在調用互換函數之前發送資產到核心合約。然後合約通過比較上次記錄的餘額和最新余額來計算它收到了多少資產。這意味著核心合約對交易者轉賬的方式是不可知的。除了 transferFrom 以外，也可能是元交易（meta transaction），或未來任何其他的授權 ERC20 代幣轉賬的機制。

3.2.1 手續費調整

Uniswap v1通過轉入合約的代幣數量，在保持常數乘積不變之前收取交易手續費。合約強制確保了以下公式：

$$(x_1 - 0.003 \cdot x_{in}) \cdot y_1 >= x_0 \cdot y_0 \qquad (9)$$

使用flash swaps時，Uniswap v2引入了xin和yin可以同時為非零的可能性（當用戶想要返還同樣的資產，而不是互換時）。為了處理這種情況，同時正確地收取手續費，合約強制確保：

$$(x_1 - 0.003 \cdot x_{in}) \cdot (y_1 - 0.003 \cdot y_{in}) >= x_0 \cdot y_0 \qquad (10)$$

為了簡化鏈上計算，兩邊同時乘以1000000，得到：

$$(1000 \cdot x_1 - 3 \cdot x_{in}) \cdot (1000 \cdot y_1 - 3 \cdot y_{in}) >= 1000000 \cdot x_0 \cdot y_0 \qquad (11)$$

3.2.2 sync()和 skim()函數

為了防止特殊實現用來修改交易對合約餘額的代幣，並且更優雅地處理總發行量超過的代幣，Uniswap v2 有兩個救援函數：sync()和 skim()。

sync()作用是在代幣異步地減少交易對的餘額時的恢復機制。這種情況下，交易會收到次優的匯率，如果沒有流動性提供者作出反應，交易對會卡住。sync()的作用是設置合約的儲備金為合約當前的餘額，提供一種稍微優雅一點的恢復機制。

skim()作用是在發送到代幣的數量溢出了 uint112 大小的儲備金存儲空間時的恢復機制，否則交易會失敗。skim()函數允許用戶將提出交易對當前餘額和的差值大於 0 時，將差值提出到調用者的地址。

3.3 處理非標準和非常規代幣

ERC20 標準要求 transfer()函數和 transferFrom()函數返回布爾值表示調用的成功或失敗。一些代幣對這兩個函數的實現沒有返回值，比如泰達幣（USDT）和幣安幣（BNB）。Uniswap v1 將這種不標準的函數返回值轉換成 false，表示轉賬不成功，並且回滾交易，導致轉賬失敗。

Uniswap v2 用不同的方式處理非標準的代幣實現。具體來說，如果一次 transfer()調用沒有返回值，Uniswap v2 把它轉換為成功而非失敗。這個改動不應該影響任何遵從 ERC20 協議的代幣（因為那些代幣中 transfer()總是有返回值）。

Uniswap v1 此外假設調用 transfer()和 transferFrom()不能觸發 Uniswap 交易對合約的重入調用。某些 ERC20 代幣違反了這個假設，包括支持 ERC777 協議的"hooks"的代幣。為了完全支持這些代幣，Uniswap v2 包含了一個“鎖”，直接防止重入所有公共的修改狀態的函數。這也保護防止了在快速互換（flash swaps）中從用戶定義的回調函數重入，如 2.3 節所描述的那樣。

3.4 初始化流動性代幣供給

當新的流動性提供者向現有的 Uniswap 交易對中存代幣時，計算鑄造的流動性代幣（譯者註：流動性代幣需要看 Uniswap v1 白皮書）數量基於現有的代幣數量：

$$s_{minted} = \frac{x_{deposited}}{x_{starting}} \cdot s_{starting} \tag{12}$$

如果是第一個存款人呢？在 $x_{starting}$ 為0的情況下，這個公式不能用。

Uniswap v1設初始份額供給等於存入地ETH數量（以Wei計）。這有一定的合理價值，因為如果初始流動性是在合理價格存入的，那麼1流動性份額（和ETH一樣是18位小數精度代幣）大約值2ETH。

但是這意味著流動性資金池份額的價值依賴於初始存入的比例，這完全可能是任意值，尤其是因為沒有任何比例可以反應真實價格的保證。另外，Uniswap v2支持任意交易對，有許多交易對根本不包含ETH。

相反，Uniswap v2初始鑄造份額等於存入代幣數量的幾何平均值：

$$s_{minted} = \sqrt{x_{deposited} \cdot y_{deposited}} \tag{13}$$

$$s_{minted} = \frac{x_{deposited}}{x_{starting}} \cdot s_{starting} \tag{12}$$

如果是第一個存款人呢？在 $x_{starting}$ 為0的情況下，這個公式不能用。

Uniswap v1設初始份額供給等於存入地ETH數量（以Wei計）。這有一定的合理價值，因為如果初始流動性是在合理價格存入的，那麼1流動性份額（和ETH一樣是18位小數精度代幣）大約值2ETH。

但是這意味著流動性資金池份額的價值依賴於初始存入的比例，這完全可能是任意值，尤其是因為沒有任何比例可以反應真實價格的保證。另外，Uniswap v2支持任意交易對，有許多交易對根本不包含ETH。

相反，Uniswap v2初始鑄造份額等於存入代幣數量的幾何平均值：

$$s_{minted} = \sqrt{x_{deposited} \cdot y_{deposited}} \tag{13}$$

$$s_{minted} = \frac{x_{deposited}}{x_{starting}} \cdot s_{starting} \qquad (12)$$

如果是第一個存款人呢？在 $x_{starting}$ 為0的情況下，這個公式不能用。

Uniswap v1設初始份額供給等於存入地ETH數量（以Wei計）。這有一定的合理價值，因為如果初始流動性是在合理價格存入的，那麼1流動性份額（和ETH一樣是18位小數精度代幣）大約值2ETH。

但是這意味著流動性資金池份額的價值依賴於初始存入的比例，這完全可能是任意值，尤其是因為沒有任何比例可以反應真實價格的保證。另外，Uniswap v2支持任意交易對，有許多交易對根本不包含ETH。

相反，Uniswap v2初始鑄造份額等於存入代幣數量的幾何平均值：

$$s_{minted} = \sqrt{x_{deposited} \cdot y_{deposited}} \qquad (13)$$

這個公式確保了流動性資金池份額的價值在任意時間和在本質上和初始存入的比例無關。舉個例子，假設當前 1 ABC 的價格是 100XYZ。如果初始存款是 2 ABC 和 200 XYZ（比例 1:100），存款人會獲得份額。這些份額現在應該任然值 2 ABC 和 200 XYZ，加上累加手續費。

如果初始存款是 2 ABC 和 800 XYZ（1：400 比例），存款人會收到資金池份額。

以上公式確保了流動性資金池不會少於資金池中儲備金額的幾何平均值。但是，流動性資金池份額的價值隨時間增長是可能的，通過累加交易手續費或者向流動性資金池“捐款”。理論上，這會導致一種極端情況，最小的流動性份額數量（1e-18 份額）過於值錢，以至於無法位小流動性提供者提供任何流動性。

為了減輕這種情況，Uniswap v2 銷毀第一次鑄造的 1e-15 資金池份額，發送到全零地址而不是鑄造者。這對任何代幣交易對都是微不足道的。但是這顯著地提高了上述攻擊地成本。為了提高流動性資金池份額價值到 100 美元，攻擊者需要捐獻 100000 美元到資金池總，這會被作為流動性永久鎖定。

3.5 包裝 ETH

以太坊原生資產 ETH 的轉賬接口和 ERC20 交互用的標準接口不同。結果，以太坊上許多其他的協議不支持 ETH，而是使用了一種標準的“包裝的 ETH”代幣，WETH。

Uniswap v1 是一個例外。因為每個 Uniswap v1 交易對包含了 ETH 作為其中一種資產，所以它可以直接處理 ETH，而且更高效地使用 gas。

因為 Uniswap v2 支持任意 ERC20 交易對，它現在不再支持無包裝的 ETH。如果添加這個特性需要兩倍的核心代碼，並且產生 ETH 和 WETH 流動性分散的風險。原生 ETH 需要包裝後才能在 Uniswap v2 上交易。

3.6 確定交易對地址

和 Uniswap v1 一樣，Uniswap v2 交易對也是通過單一的工廠合約進行實例化的。在 Uniswap v1 中，交易對合約用 CREATE 運算碼進行創建，這意味著合約地址依賴於交易對創建的順序。Uniswap v2 使用以太坊新的 CREATE2 運算碼來創建確定地址的交易對合約，這意味著可以在鏈下計算某個交易對的地址，不用查看以太坊區塊鏈的狀態。

3.7 最大代幣餘額

為了高效地實現預言機制，Uniswap v2 只支持最高儲備餘額 。這個數字足以支持總發行量超過一千萬億的 18 位小數精度的代幣。

如果儲備餘額超過了，任何 swap 函數調用都會失敗（由於_update() 函數中的檢查邏輯）。要從這個狀態中恢復，任何用戶都可以調用 skim()函數從流動性池中刪除超出的資產。

引用

- [1] Hayden Adams. 2018. url: hackmd.io/@477aQ9OrQTCb.
- [2] Guillermo Angeris et al. An analysis of Uniswap markets. 2019. arXiv: 1911.03380[q-fin.TR].
- [3] samczsun. Taking undercollateralized loans for fun and for profit. Sept. 2019. url: samczsun.com/taking-und.
- [4] Fabian Vogelsteller and Vitalik Buterin. Nov. 2015. url: eips.ethereum.org/EIPS/.
- [5] Jordi Baylina Jacques Dafflon and Thomas Shababi. EIP 777: ERC777 Token Standard.Nov. 2017. url: eips.ethereum.org/EIPS/.
- [6] Radar. WTF is WETH? url: weth.io/.
- [7] Uniswap.info. Wrapped Ether (WETH). url: uniswap.info/token/0xc0
- [8] Vitalik Buterin. EIP 1014