SC+业务快速落地：四种首笔凭证发行与融资技术方案

摘要

本报告旨在为 Unloq 的 **SC+凭证**业务提供一个可立即执行的、包含多种选项的快速落地技术方案。我们的目标是：**在最短时间内，利用 XRPL 的原生功能，跑通第一笔 SC+凭证的发行、转让和融资流程。**

本报告将摒弃宏大的战略叙事，聚焦于具体的技术实现步骤，并提供四种并行的、端到端的全流程操作指南，每种方案都对应不同的战略侧重：

1. **方案一：MPT 方案** — 功能完备，是 Ripple 的 RWA 战略方向。
2. **方案二：IOU 方案** — 快速灵活，利用成熟生态和 Memo 字段实现。
3. **方案三：NFT 方案** — 模型匹配度高，自带交易市场，完美契合"一发票一凭证"。
4. **方案四：Credentials 方案** — 极致合规，作为非代币的"声明"进行资产存证。

报告将详细列出每种方案所需的 XRPL 交易类型、JSON 字段、参数配置和 API 调用示例，旨在成为技术团队可以直接上手的开发手册，确保 SC+业务在 XRPL 上的迅速启动和验证。

---

1. 准备工作：定义参与方与账户

无论选择哪种方案，我们都需要为流程中的所有参与方在 XRPL 上创建或指定账户。所有账户都需要激活（至少有 10 XRP 的储备金）。

参与方	角色	XRPL 账户地址（示例）
Unloq	SC+平台运营方、资产发行方、KYC 凭证签发方	rUnloqIssuer...
供应商 A	应收账款的原始债权人	rSupplierA...

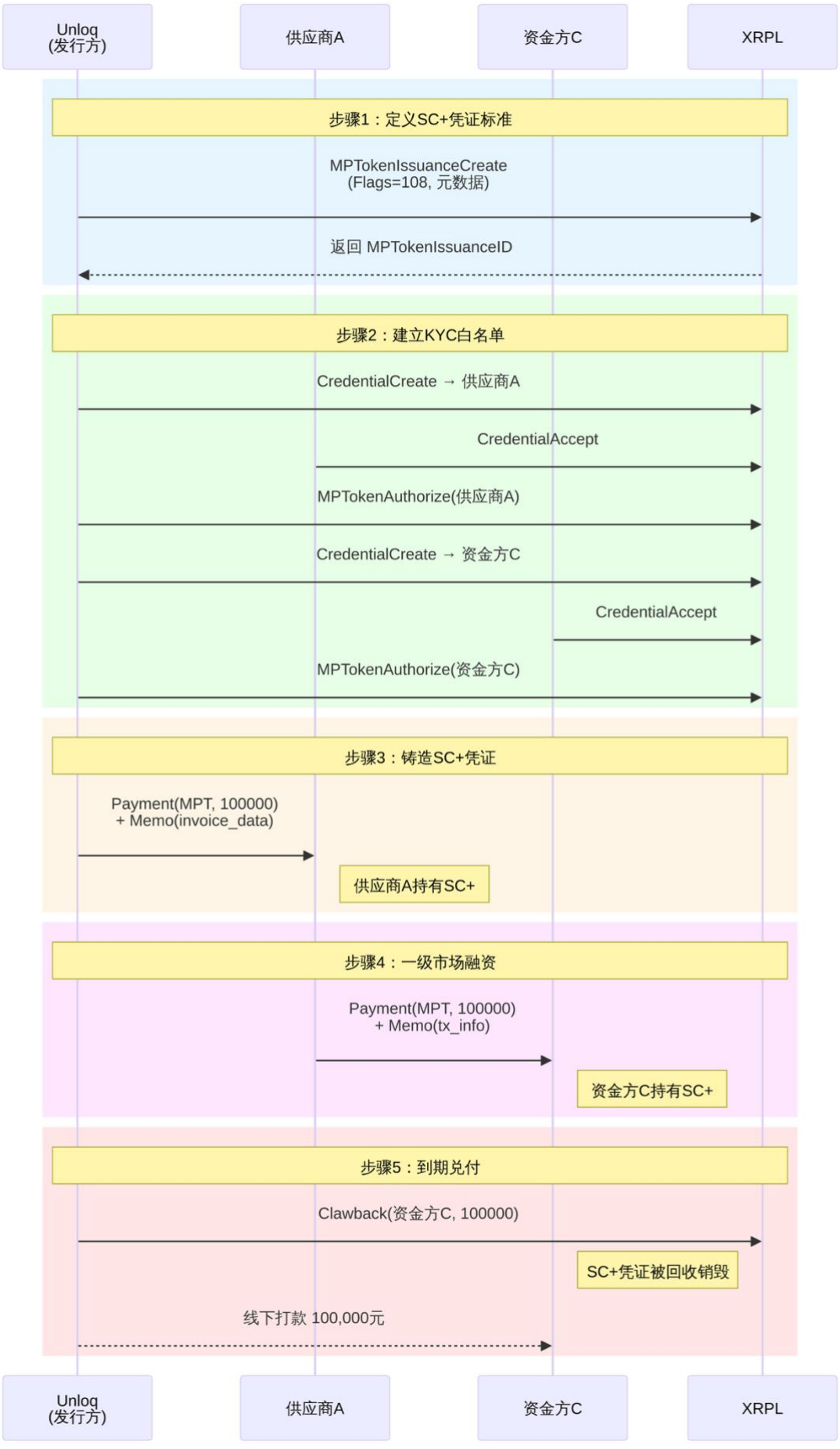
参与方	角色	XRPL 账户地址（示例）
核心企业 B	应收账款的债务人	rCoreEnterpriseB... (本方案中仅作信息参考)
资金方 C	一级市场投资者	rInvestorC...
资金方 D	二级市场投资者	rInvestorD...

业务场景：供应商 A 有一笔对核心企业 B 的 **100,000 元**应收账款（发票号 INV-2026-001，到期日 2026-05-24）。供应商 A 希望将该应收账款通过 Unloq 平台上链，并融资给资金方 C。

---

2. 方案一： MPT (Multi-Purpose Token) 方案

此方案功能最完备，是 XRPL 专为 RWA 设计的"官方"解决方案，提供原生元数据、精细化权限控制和 Clawback 回收能力。


Unlog
(发行方)
供应商A
资金方C
XRPL
步骤1：定义SC+凭证标准
MPTokenIssuanceCreate
(Flags=108, 元数据)
返回 MPTokenIssuanceID
步骤2：建立KYC白名单
CredentialCreate → 供应商A
CredentialAccept
MPTokenAuthorize(供应商A)
CredentialCreate → 资金方C
CredentialAccept
MPTokenAuthorize(资金方C)
步骤3：铸造SC+凭证
Payment(MPT, 100000)
+ Memo(invoice_data)
供应商A持有SC+
步骤4：一级市场融资
Payment(MPT, 100000)
+ Memo(tx_info)
资金方C持有SC+
步骤5：到期兑付
Clawback(资金方C, 100000)
SC+凭证被回收销毁
线下打款 100,000元
Unlog
(发行方)
供应商A
资金方C
XRPL

2.1 步骤 1：定义 SC+凭证标准 (MPTokenIssuanceCreate)

通过一次性的 MPTokenIssuanceCreate 交易，定义 SC+凭证的"模具"。

交易发送方： rUnloqIssuer...

```
{
  "TransactionType": "MPTokenIssuanceCreate",
  "Account": "rUnloqIssuer...",
  "AssetScale": 2,
  "MaximumAmount": "100000000000",
  "TransferFee": 0,
  "Flags": 108,
  "MPTokenMetadata":
"7B2274223A225343504C5553222C226E223A22E4BE9BE5BA94E993BEE98791E89E8DE587ADE8AF81222C226163223A22727761222C226173223A22707269766174655F637265646974222C22696E223A22556E6C6F71227D"
}
```

字段详解：

Flags: 108 是一个关键的位组合，代表四项能力：tfRequireAuth (4) 表示必须授权才能持有，这是构建白名单体系的基石；tfCanTransfer (32) 表示凭证可以转让给其他授权用户；tfCanClawback (64) 表示 Unloq 作为发行方可以回收凭证，用于到期兑付或处理异常情况；tfCanEscrow (8) 表示凭证可以被托管，是实现 DvP 原子结算的前提。

MPTokenMetadata 是资产的"身份证"，内容是 JSON 的十六进制编码。解码后的明文如下：

```
{
  "t": "SCPLUS",
  "n": "供应链金融凭证",
  "ac": "rwa",
  "as": "private_credit",
  "in": "Unloq"
}
```

交易成功后，会创建一个 MPTokenIssuance 账本对象，并生成一个唯一的 MPTokenIssuanceID。**这个 ID 将用于后续所有与该批次 SC+相关的操作。**

2.2 步骤 2：建立 KYC 白名单 (CredentialCreate & MPTokenAuthorize)

由于设置了 tfRequireAuth，任何想持有 SC+的用户都必须得到 Unloq 的授权。

1. 签发 KYC 凭证（可选但推荐）

```
{
  "TransactionType": "CredentialCreate",
  "Account": "rUnloqIssuer...",
  "Subject": "rSupplierA...",
  "CredentialType": "4B59435F415050524F564544"
}
```

> *CredentialType 解码明文：KYC_APPROVED*

此交易需要 rSupplierA...发送 CredentialAccept 交易来接受，凭证才正式生效。对所有参与方（C, D 等）重复此操作。

2. 授权持有 SC+凭证

```
{
  "TransactionType": "MPTokenAuthorize",
  "Account": "rUnloqIssuer...",
  "MPTokenIssuanceID": "(步骤 1 中获取的 ID)",
  "Holder": "rSupplierA..."
}
```

对所有需要持有该 SC+的账户（rInvestorC..., rInvestorD...等）重复此操作。

2.3 步骤 3：铸造并发行 SC+ (Payment)

1. 准备链下数据与哈希

根据隐私模型，敏感信息存在链下，链上只存哈希或非敏感摘要。链下完整 JSON 数据如下：


```
{
  "invoiceId": "INV-2026-001",
  "debtor": "核心企业 B（深圳 XX 科技有限公司）",
  "debtorId": "91440300XXXXXXXX",
  "creditor": "供应商 A（东莞 XX 电子有限公司）",
  "amount": 100000.00,
  "currency": "CNY",
  "issueDate": "2026-02-23",
  "dueDate": "2026-05-24",
  "description": "电子元器件采购合同 PO-2026-0158",
  "contractHash": "a1b2c3d4e5f6..."
}
```


计算该 JSON 字符串的 SHA-256 哈希（示例）：e3b0c44298fc1c149afbf4c8996fb924...

2. 铸造 SC+

铸造就是发行方（Unloq）向供应商 A 的账户发送一笔 Payment 交易。


```
{
  "TransactionType": "Payment",
  "Account": "rUnloqIssuer...",
  "Destination": "rSupplierA...",
  "Amount": {
    "mpt_issuance_id": "(步骤 1 中获取的 ID)",
    "value": "100000"
  },
  "Memos": [
    {
      "Memo": {
        "MemoType": "696E766F6963655F64617461",
        "MemoData":
"7B226964223A22494E562D323032362D303031222C22616D6F756E74223A3130303030302C226375722
23A22434E59222C22647565223A22323032362D30352D3234222C2264657363223A22E794B5E5AD90E58
583E599A8E4BBB6E98787E8B4ADE59088E5908C222C2268617368223A22653623063343432393866633
163313439616662663463383939366662393234 2E2E2E227D"
      }
```

```
  }
 ]
}
```

> *MemoType 解码明文：invoice_data*

> *MemoData 解码明文：*

```
{
 "id": "INV-2026-001",
 "amount": 100000,
 "cur": "CNY",
 "due": "2026-05-24",
 "desc": "电子元器件采购合同",
 "hash": "e3b0c44298fc1c149afbf4c8996fb924..."
}
```

交易成功后，rSupplierA...账户将持有面额为 100,000 的 SC+凭证。

2.4 步骤 4：一级市场融资 (Payment)

场景：资金方 C 希望以 98,000 元的价格从供应商 A 处购买这笔 SC+凭证。在确认收到资金方 C 的 98,000 元线下付款后，供应商 A 向 C 转让 SC+凭证。

```
{
 "TransactionType": "Payment",
 "Account": "rSupplierA...",
 "Destination": "rInvestorC...",
 "Amount": {
  "mpt_issuance_id": "(步骤 1 中获取的 ID)",
  "value": "100000"
 },
 "Memos": [
  {
   "Memo": {
    "MemoType": "74785F696E666F",
    "MemoData":
"7B2274797065223A227072696D6172795F73616C65222C22696E76223A22494E562D323032362D30303
```

```
1222C227072696365223A39383030302C22637572223A22434E59222C2262757965725F6B7963223A224
35F4B59435F303032227D"
    }
  }
 ]
```

```
}
```

> *MemoType 解码明文：tx_info*

> *MemoData 解码明文：*

```
{
  "type": "primary_sale",
  "inv": "INV-2026-001",
  "price": 98000,
  "cur": "CNY",
  "buyer_kyc": "C_KYC_002"
}
```

交易成功后，rInvestorC...成为该 SC+凭证的持有人。

> *注意：为消除交易对手风险，此步骤在成熟阶段应使用 Token Escrow 实现原子化的 DvP 结算。*

2.5 步骤 5：到期兑付 (Clawback)

场景：90 天后，核心企业 B 向 Unloq 支付了 100,000 元。Unloq 需要收回凭证并销毁，同时将资金结算给最终持有人。

```
{
  "TransactionType": "Clawback",
  "Account": "rUnloqIssuer...",
  "Amount": {
    "mpt_issuance_id": "(步骤 1 中获取的 ID)",
    "value": "100000"
  },
  "Holder": "rInvestorC..."
```

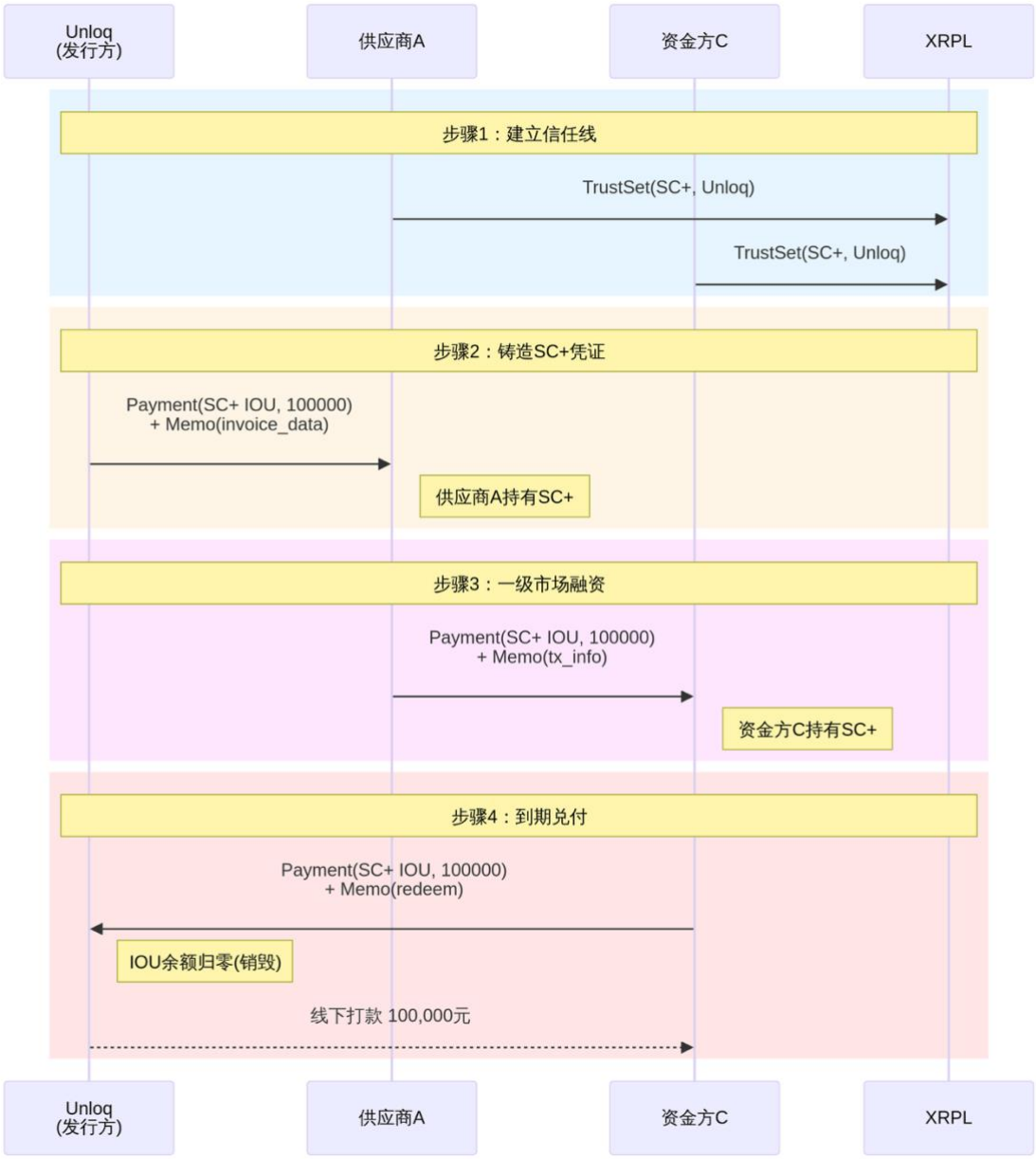
```
}
```

交易成功后，SC+凭证从 rInvestorC...账户中消失，回到了发行方 rUnloqIssuer...手中（可视为已销毁）。Unloq 随后完成对 C 的线下打款 100,000 元。整个生命周期闭环。

---

3. 方案二：IOU (Trustline Token) 方案

此方案利用成熟的 IOU 生态，通过 Memo 字段实现 RWA 数据存证，上市速度最快。IOU 与 MPT 在核心机制上区别不大——都有 amount 字段，都通过 Payment 铸造，都可以附加 Memo。



3.1 步骤 1：建立信任线 (TrustSet)

所有希望持有 SC+的账户都需要向 Unloq 的发行账户建立信任线。这是 IOU 的前置条件，类似于 MPT 中的 MPTokenAuthorize。

交易发送方：rSupplierA...（及其他参与方）

```
{
  "TransactionType": "TrustSet",
  "Account": "rSupplierA...",
  "LimitAmount": {
    "currency": "534350000000000000000000000000000000000000",
```

```
  "issuer": "rUnloqIssuer...",
  "value": "1000000000"
 }
}
```

> *currency 说明：XRPL 中 3 字符以上的货币代码需要使用 160 位十六进制表示。上述值代表"SCP"。如果使用 3 字符代码"SC+"，可直接写"currency": "SC+"。*

对所有参与方（rInvestorC..., rInvestorD...等）重复此操作。

3.2 步骤 2：铸造并发行 SC+ (Payment)

Unloq 向供应商 A 的账户发送一笔 Payment 交易。Memo 字段承载了所有的业务存证数据。

```
{
 "TransactionType": "Payment",
 "Account": "rUnloqIssuer...",
 "Destination": "rSupplierA...",
 "Amount": {
  "currency": "SC+",
  "issuer": "rUnloqIssuer...",
  "value": "100000"
 },
 "Memos": [
  {
   "Memo": {
    "MemoType": "696E766F6963655F64617461",
    "MemoData":
"7B226964223A22494E562D323032362D303031222C22616D6F756E74223A3130303030302C226375722
23A22434E59222C22647565223A22323032362D30352D3234222C2264657363223A22E794B5E5AD90E58
583E599A8E4BBB6E98787E8B4ADE59088E5908C222C2268617368223A2265336230633434323938666633
163313439616662663463383939366662393234 2E2E2E227D"
   }
  }
 ]
}
```

> *MemoType 解码明文：invoice_data*

> *MemoData 解码明文：*


```
{
  "id": "INV-2026-001",
  "amount": 100000,
  "cur": "CNY",
  "due": "2026-05-24",
  "desc": "电子元器件采购合同",
  "hash": "e3b0c44298fc1c149afbf4c8996fb924..."
}
```


3.3 步骤 3：一级市场融资 (Payment)

供应商 A 向资金方 C 发送 Payment 交易转让 IOU。


```
{
  "TransactionType": "Payment",
  "Account": "rSupplierA...",
  "Destination": "rInvestorC...",
  "Amount": {
    "currency": "SC+",
    "issuer": "rUnloqIssuer...",
    "value": "100000"
  },
  "Memos": [
    {
      "Memo": {
        "MemoType": "74785F696E666F",
        "MemoData":
"7B2274797065223A227072696D6172795F73616C65222C22696E76223A22494E562D323032362D303031222C227072696365223A39383030302C22637572223A22434E59222C2262757965725F6B7963223A22435F4B59435F303032227D"
      }
    }
  ]
}
```


> *MemoType 解码明文：tx_info*

> *MemoData 解码明文：*


```
{
  "type": "primary_sale",
  "inv": "INV-2026-001",
  "price": 98000,
  "cur": "CNY",
  "buyer_kyc": "C_KYC_002"
}
```


3.4 步骤 4：到期兑付

IOU 方案的兑付方式与 MPT 不同。最终持有人将 SC+ (IOU) 发送回 Unloq 的发行账户，IOU 余额自动归零（因为发行方收到自己发行的 IOU 等于销毁）。Unloq 随后完成线下打款。


```
{
  "TransactionType": "Payment",
  "Account": "rInvestorC...",
  "Destination": "rUnloqIssuer...",
  "Amount": {
    "currency": "SC+",
    "issuer": "rUnloqIssuer...",
    "value": "100000"
  },
  "Memos": [
    {
      "Memo": {
        "MemoType": "74785F696E666F",
        "MemoData":
"7B2274797065223A2272656465656D222C22696E76223A22494E562D323032362D303031227D"
      }
    }
  ]
}
```


> *MemoType 解码明文：tx_info*

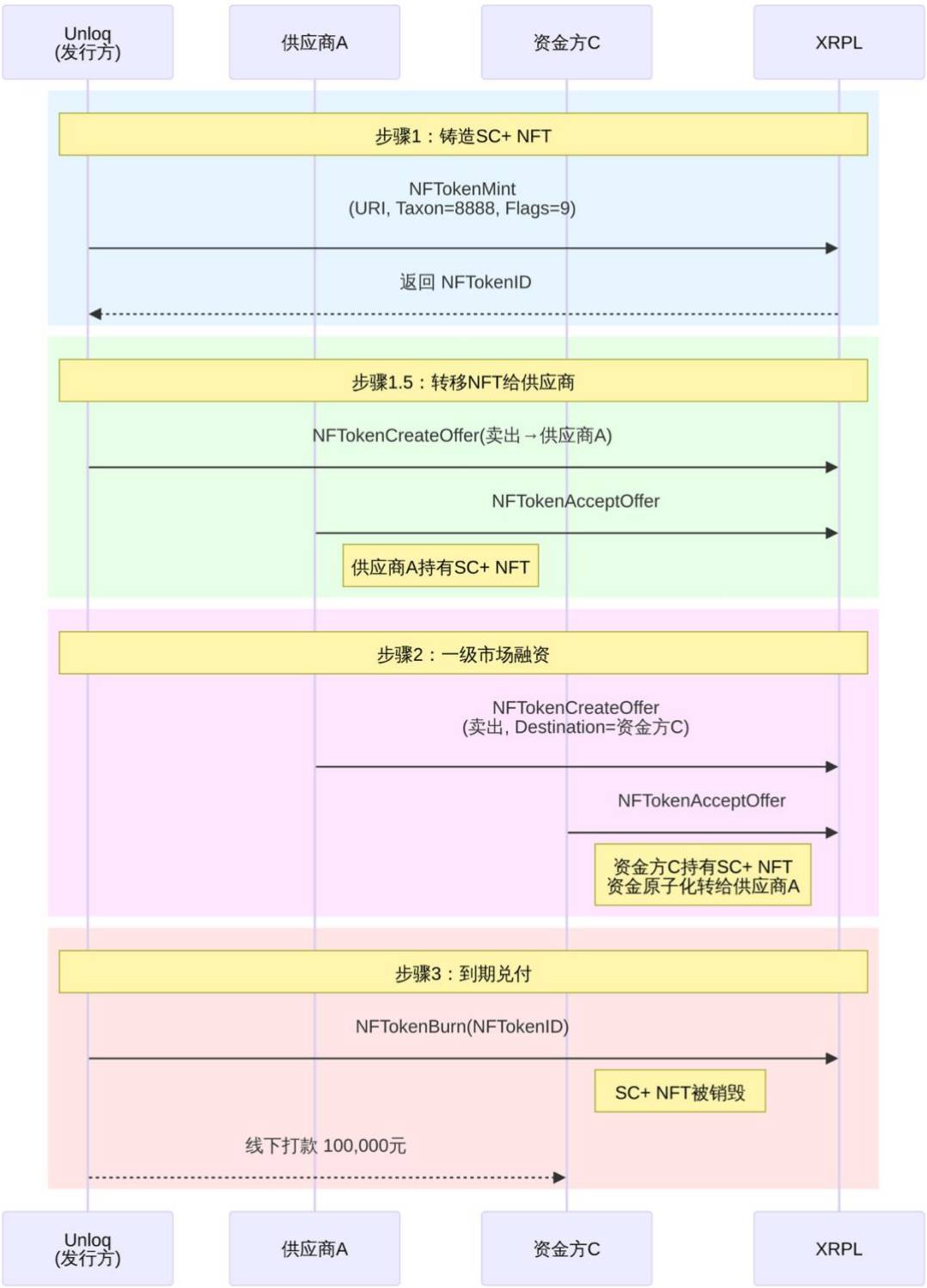
> *MemoData 解码明文：*

```
{
  "type": "redeem",
  "inv": "INV-2026-001"
}
```

---

4. 方案三： NFT (Non-Fungible Token) 方案

此方案与 SC+业务模型匹配度最高，每个 NFT 代表一张唯一的应收账款凭证，并自带交易市场。



4.1 步骤 1：铸造 SC+凭证 (NFTokenMint)

Unloq 为每一笔应收账款铸造一个唯一的 NFT。

交易发送方：rUnloqIssuer...

```
{
  "TransactionType": "NFTokenMint",
  "Account": "rUnloqIssuer...",
  "NFTokenTaxon": 8888,
  "Flags": 9,
  "TransferFee": 500,
  "URI":
"68747470733A2F2F6170692E756E6C6F712E696F2F73632D706C75732F494E562D323032362D3030313F746F6B656E3D65794A68624763694F694A49557A49314E694A392E2E2E"
}
```

字段详解：

- NFTokenTaxon: 8888 (示例)，代表"SC+凭证"这个集合，用于对 NFT 进行分类。
- Flags: 9: tfBurnable (1) + tfTransferable (8)，表示发行方可销毁，且该 NFT 可转让。
- TransferFee: 500: 代表 5.00%的二级市场转让费，Unloq 可从中获利。
- URI: 指向链下详细数据的 API 链接，解码明文为：https://api.unloq.io/sc-plus/INV-2026-001?token=eyJhbGciOiJIUzI1NiJ9...

交易成功后，rUnloqIssuer...账户将持有一个新的 NFT，拥有唯一的 NFTokenID。

4.2 步骤 2：一级市场融资 (NFTokenCreateOffer + NFTokenAcceptOffer)

NFT 的转让不使用 Payment，而是通过 Offer 机制。这里 Unloq 可以作为 Broker（经纪人）撮合交易。

第一步：供应商 A 创建卖出要约

首先，Unloq 需要将 NFT 转给供应商 A。然后，供应商 A 创建一个卖出要约，指定只有资金方 C 可以接受。

```
{
  "TransactionType": "NFTokenCreateOffer",
  "Account": "rSupplierA...",
  "NFTokenID": "(步骤 1 中获取的 NFT ID)",
  "Amount": "98000000000",
  "Flags": 1,
```

```
  "Destination": "rInvestorC..."
}
```

- Amount: 98,000 XRP (示例，也可以是 IOU 稳定币)。
- Flags: 1: tfSellNFToken，表示这是一个卖出要约。
- Destination: 限定只有 rInvestorC...可以接受此要约。

第二步：资金方 C 接受要约

```
{
  "TransactionType": "NFTokenAcceptOffer",
  "Account": "rInvestorC...",
  "NFTokenSellOffer": "(上一步创建的 Offer 索引)"
}
```

交易成功后，NFT 从供应商 A 转移到资金方 C，同时 98,000 XRP 从 C 转移到 A，原子化完成。

4.3 步骤 3：到期兑付 (NFTokenBurn)

核心企业付款后，Unloq 行使 tfBurnable 赋予的权力，直接销毁该 NFT。

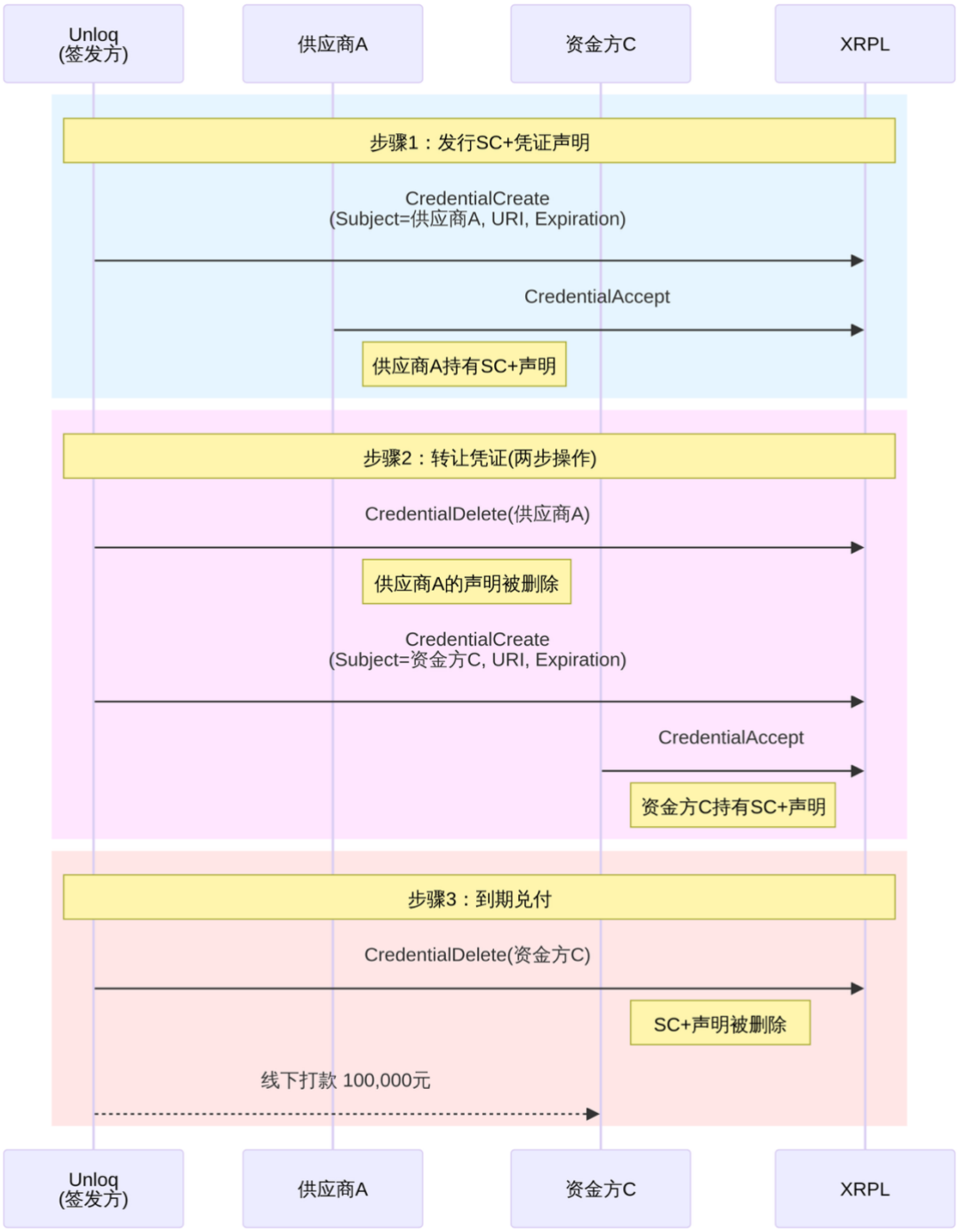
```
{
  "TransactionType": "NFTokenBurn",
  "Account": "rUnloqIssuer...",
  "NFTokenID": "(步骤 1 中获取的 NFT ID)"
}
```

交易成功后，该 NFT 从链上永久消失。Unloq 随后完成对最终持有人 C 的线下打款。整个生命周期闭环。

---

5. 方案四：Credentials 方案

此方案完全不使用代币，在合规压力最大时提供了一条"存证"路径。Credentials 本质上是一个账户对另一个账户的"声明"，不涉及 Token 概念。



5.1 步骤 1：发行 SC+凭证声明 (CredentialCreate)

Unloq 为供应商 A 签发一个代表 SC+的"声明"。

```
{
  "TransactionType": "CredentialCreate",
  "Account": "rUnloqIssuer...",
  "Subject": "rSupplierA...",
  "CredentialType": "5343504C55535F494E562D323032362D303031",
  "URI":
"68747470733A2F2F6170692E756E6C6F712E696F2F73632D706C75732F494E562D323032362D3030313F
746F6B656E3D65794A68624763694F694A49557A49314E694A392E2E2E",
  "Expiration": 1779753600
}
```


> *CredentialType 解码明文：SCPLUS_INV-2026-001*

> *URI 解码明文：https://api.unloq.io/sc-plus/INV-2026-001?token=eyJhbGciOiJIUzI1NiJ9...*

> *Expiration 说明：值为 Ripple Epoch 时间戳，对应 2026-05-24（应收账款到期日）。*

供应商 A 需发送 CredentialAccept 交易接受此声明，使其正式生效。

5.2 步骤 2："转让"凭证（两步操作）

当供应商 A 希望将 SC+凭证转让给资金方 C 时，由 Unloq 执行以下两步操作：

第一步：删除旧凭证


```
{
  "TransactionType": "CredentialDelete",
  "Account": "rUnloqIssuer...",
  "Subject": "rSupplierA...",
  "Issuer": "rUnloqIssuer...",
  "CredentialType": "5343504C55535F494E562D323032362D303031"
}
```


第二步：创建新凭证

```
{
  "TransactionType": "CredentialCreate",
  "Account": "rUnloqIssuer...",
  "Subject": "rInvestorC...",
  "CredentialType": "5343504C55535F494E562D323032362D303031",
  "URI":
"68747470733A2F2F6170692E756E6C6F712E696F2F73632D706C75732F494E562D323032362D3030313F
746F6B656E3D65794A68624763694F694A49557A49314E694A392E2E2E",
  "Expiration": 1779753600
}
```


资金方 C 同样需要接受此新凭证。

5.3 步骤 3：到期兑付 (CredentialDelete)

核心企业付款后，Unloq 直接删除最终持有人 C 持有的凭证。


```
{
  "TransactionType": "CredentialDelete",
  "Account": "rUnloqIssuer...",
  "Subject": "rInvestorC...",
  "Issuer": "rUnloqIssuer...",
  "CredentialType": "5343504C55535F494E562D323032362D303031"
}
```
