Duniter Blockchain Protocol

Les bases du protocole

Documents

La communication entre noeuds ou entre les clients et les noeuds s’effectue par envoi de documents.

Différents types :

- Identity
- Revocation
- Certification
- Membership
- Transaction
- Block

Identity

Un document Identity est envoyé pour déclarer une identité, c'est à dire lier une clé publique **PUBLIC_KEY** à une chaîne de caractères **USER_ID** représentant l'identité (un pseudo). C'est cette identité à laquelle feront référence des certifications dans le processus d'acceptation des membres.

```
Version: 10
Type: Identity
Currency: CURRENCY_NAME
Issuer: PUBLIC_KEY
UniqueID: USER_ID
Timestamp: BLOCK_UID
SIGNATURE
```

Identity

Ex: identité “lolcat” liée à la clé “HgTTJL…”

```
Version: 10
Type: Identity
Currency: beta_brousouf
Issuer: HgTTJLAQ5sqfknMq7yLPZbehtuLSsKj9CxWN7k8QvYJd
UniqueID: lolcat
Timestamp: 32-DB30D958EE5CB75186972286ED3F4686B8A1C2CD
J3G9oM5AKYZNLAB5Wx499w61NuUoS57JVccTShUbGpCMjCqj9yXXqNq7dyZpDWA6BxipsiaMZhujMeBfCznzyci
```

Revocation

Un document Revocation est envoyé quand le détenteur d'une identité souhaite la supprimer, pour une raison ou pour une autre : identité faite par erreur, ou dont la sécurité est compromise, etc.

```
Version: 10
Type: Revocation
Currency: CURRENCY_NAME
Issuer: PUBLIC_KEY
IdtyUniqueID: USER_ID
IdtyTimestamp: BLOCK_UID
IdtySignature: IDTY_SIGNATURE
REVOCATION_SIGNATURE
```

Revocation

Ex: pour notre identité “lolcat”, la révocation sera celle-ci . A noter, une révocation contient l’identité qu’elle révoque, un noeud recevant cette révocation peut ainsi vérifier la validité de l’identité avant de la révoquer.

```
Version: 10
Type: Identity
Currency: beta_brousouf
Issuer: HgTTJLAQ5sqfknMq7yLPZbehtuLSsKj9CxWN7k8QvYJd
UniqueID: lolcat
Timestamp: 32-DB30D958EE5CB75186972286ED3F4686B8A1C2CD
J3G9oM5AKYZNLAB5Wx499w61NuUoS57JVccTShUbGpCMjCqj9yXXqNq7dyZpDWA6BxipsiaMZhujMeBfCznzyci
```

```
Version: 10
Type: Revocation
Currency: beta_brousouf
Issuer: HgTTJLAQ5sqfknMq7yLPZbehtuLSsKj9CxWN7k8QvYJd
IdtyUniqueID: lolcat
IdtyTimestamp: 32-DB30D958EE5CB75186972286ED3F4686B8A1C2CD
IdtySignature: J3G9oM5AKYZNLAB5Wx499w61NuUoS57JVccTShUbGpCMjCqj9yXXqNq7dyZpDWA6BxipsiaMZhujMeBfCzn
SoKwoa8PFfCDJWZ6dNCv7XstezHcc2BbKiJgVDXv82R5zYR83nis9dShLgWJ5w48noVUHimdngzYQneNYSMV3rk
```

Certification

Pour avoir le droit d'utiliser le réseau, une identité doit être certifiée par plusieurs identités existantes dans le cadre d'une "toile de confiance" (Web of Trust).

Ces certifications sont appliquées par des identitées à des identitées, et se déclare au travers d'un document. Là encore l'identité certifiée est contenue dans le document.

```
Version: 10
Type: Certification
Currency: CURRENCY_NAME
Issuer: PUBLIC_KEY
IdtyIssuer: IDTY_ISSUER
IdtyUniqueID: USER_ID
IdtyTimestamp: BLOCK_UID
IdtySignature: IDTY_SIGNATURE
CertTimestamp: BLOCK_UID
CERTIFIER_SIGNATURE
```

Certification

Ex: L'identité "lolcat" certifiée par une identité dont la clé publique est "DNann…". On remarque que l'identité à certifiée est incluse dans la certification, mais pas l'identité certifiante : en effet, pour avoir une valeur une certification ne peut être prodiguée que par une identité déjà connue.

```
Version: 10
Type: Identity
Currency: beta_brousouf
Issuer: HgTTJLAQ5sqfknMq7yLPZbehtuLSsKj9CxWN7k8QvYJd
UniqueID: lolcat
Timestamp: 32-DB30D958EE5CB75186972286ED3F4686B8A1C2CD
J3G9oM5AKYZNLAB5Wx499w61NuUoS57JVccTShUbGpCMjCqj9yXXqNq7dyZpDWA6BxipsiaMZhujMeBfCznzyci
```

```
Version: 10
Type: Certification
Currency: beta_brousouf
Issuer: DNann1Lh55eZMEDXeYt59bzHbA3NJR46DeQYCS2qQdLV
IdtyIssuer: HgTTJLAQ5sqfknMq7yLPZbehtuLSsKj9CxWN7k8QvYJd
IdtyUniqueID: lolcat
IdtyTimestamp: 32-DB30D958EE5CB75186972286ED3F4686B8A1C2CD
IdtySignature: J3G9oM5AKYZNLAB5Wx499w61NuUoS57JVccTShUbGpCMjCqj9yXXqNq7dyZpDWA6BxipsiaMZhujMeBfCznzyci
CertTimestamp: 36-1076F10A7397715D2BEE82579861999EA1F274AC
SoKwoa8PFfCDJWZ6dNCv7XstezHcc2BbKiJgVDXv82R5zYR83nis9dShLgWJ5w48noVUHimdngzYQneNYSMV3rk
```

Membership

Pour intégrer un le WoT, un membre qui possède une identité doit déclarer sa volonté de l'intégrer, ce au travers de l'émission d'un document Membership.

```
Version: VERSION
Type: Membership
Currency: CURRENCY_NAME
Issuer: ISSUER
Block: M_BLOCK_UID
Membership: MEMBERSHIP_TYPE
UserID: USER_ID
CertTS: BLOCK_UID
SIGNATURE
```

IN ou OUT (pour se désinscrire).

Transaction

Les transactions sont au coeur de la notion de propriété de la monnaie dans une blockchain. C'est leur agrégation dans la blockchain qui certifie l'origine du montant de chaque identité.

Dans Duniter, la capacité à posséder de l'argent n'est pas liée au fait de faire partie de la communauté. Il suffit de posséder la clé privée permettant de dépenser un montant dans une transaction.

Chaque transaction réfère à de précédentes transaction pour prouver qu'elle possède les fonds qu'elle dit transférer, et donne des conditions pour que cet argent soit dépensé.

Transaction

Les premiers champs sont habituels dans les documents duniter.

- Locktime : tps avant d'être inclus dans la blockchain
- Issuers: clés publiques d'identités voulant dépenser
- Inputs: Tx précédentes, sources de l'argent
- Unlocks: "clés" données aux transactions précédentes pour débloquer leurs fonds
- Outputs: sorties d'argent, contenant des conditions pour être débloquées par les Unlocks de futures tx
- SIGNATURES: signatures certifiant les identités du champ Issuers

```
Version: VERSION
Type: Transaction
Currency: CURRENCY_NAME
Blockstamp: BLOCK_UID
Locktime: INTEGER
Issuers:
PUBLIC_KEY
...
Inputs:
INPUT
...
Unlocks:
UNLOCK
...
Outputs:
AMOUNT:BASE:CONDITIONS
...
Comment: COMMENT
SIGNATURES
...
```

Transaction

Ex: Ici, l'identité liée à la clé "**HsLSh..**" veut distribuer 30 beta_brousouf de la tx "**8361C9…**" Si on va voir cette transaction, il est à parier qu'il y ait une entrée du champ Output = 30:0:SIG(HsLSh…) rendant cette tx valide. Elle les envoie ensuite à "**BYfW…**" et s'en garde 5 (envoie à elle même) en l'écrivant dans le champ Output sous la forme 25:0:SIG(BYfW...)

```
Version: 10
Type: Transaction
Currency: beta_brousouf
Blockstamp: 204-00003E2B8A35370BA5A7064598F628A62D4E9EC1936BE8651CE9A85F2E06981B
Locktime: 0
Issuers:
HsLShAtzXTVxeUtQd7yi5Z5Zh4zNvbu8sTEZ53nfKcqY
Inputs:
30:0:T:8361C993631BED4733972ED7538E41CCC33660F554E3C51963E2A0AC4D6453D3:3
Unlocks:
0:SIG(0)
Outputs:
25:0:SIG(BYfWYFrsyjpvpFysgu19rGK3VHBkz4MqmQbNyEuVU64g)
5:0:SIG(HsLShAtzXTVxeUtQd7yi5Z5Zh4zNvbu8sTEZ53nfKcqY)
Comment: First transaction

42yQm4hGTJYWkPg39hQAUgP6S6EQ4vTfXdJuxKEHL1ih6YHiDL2hcwrFgBHjXLRgxRhj2VNVqqc6b4JayKqTE14r
```

Transaction

Pour débloquer des sources d'argent, on utilise des fonction de locking/unlocking.

Dans le champ Unlocks, ces fonctions sont séparées par des espaces : chacune s'applique à une fonction de lock dans le champ Output auquel l'index fait référence.

Dans le champ Outputs, on les sépare par des opérateurs logiques pour combiner les conditions à remplir.

```
Version: 10
Type: Transaction
Currency: beta_brousouf
Blockstamp: 204-00003E2B8A35370BA5A7064598F628A62D4E9EC1936BE8651CE9A85F2E06981B
Locktime: 0
Issuers:
HsLShAtzXTVxeUtQd7yi5Z5Zh4zNvbu8sTEZ53nfKcqY
CYYjHsNyg3HMRMpTHqCJAN9McjH5BwFLmDKGV3PmCuKp
9WYHTavL1pmhunFCzUwiiq4pXwvgGG5ysjZnjz9H8yB
Inputs:
40:2:T:6991C993631BED4733972ED7538E41CCC33660F554E3C51963E2A0AC4D6453D3:2
70:2:T:3A09A20E9014110FD224889F13357BAB4EC78A72F95CA03394D8CCA2936A7435:8
20:2:D:HsLShAtzXTVxeUtQd7yi5Z5Zh4zNvbu8sTEZ53nfKcqY:46
70:2:T:A0D9B4CDC113ECE1145C5525873821398890AE842F4B318BD076095A23E70956:3
20:2:T:67F2045B5318777CC52CD38B424F3E40DDA823FA0364625F124BABE0030E7B5B:5
15:2:D:9WYHTavL1pmhunFCzUwiiq4pXwvgGG5ysjZnjz9H8yB:46
Unlocks:
0:SIG(0)
1:XHX(7665798292)
2:SIG(0)
3:SIG(0) SIG(2)
4:SIG(0) SIG(1) SIG(2)
5:SIG(2)
Outputs:
120:2:SIG(BYfWYFrsyjpvpFysgu19rGK3VHBkz4MqmQbNyEuVU64g)
146:2:SIG(DSz4rgncXCytsUMW2JU2yhLquZECD2XpEkpP9gG5HyAx)
49:2:(SIG(6DyGr5LFtFmbaJYRvcs9WmBsr4cbJbJ1EV9zBbqG7A6i) || XHX(3EB4702F2AC2FD3FA4FDC46A4FC05AE8CDEE1A85))
Comment: -----@@@----- (why not this comment?)
42yQm4hGTJYWkPg39hQAUgP6S6EQ4vTfXdJuxKEHL1ih6YHiDL2hcwrFgBHjXLRgxRhj2VNVqqc6b4JayKqTE14r
2D96KZwNUvVtcapQPq2mm7J9isFcDCfykwJpVEZwBc7tCgL4qPyu17BT5ePozAE9HS6Yvj51f62Mp4n9d9dkzJoX
2XiBDpuUdu6zCPWGzHXXy8c4ATSscfFQG9DjmqMZUxDZVt1Dp4m2N5oHYVUfoPdrU9SLk4qxi65RNrfCVnvQtQJk
```

Transaction

- SIG: dans un Unlock, SIG(index) débloque une condition Output = SIG(pubkey) où Issuers[index] = pubkey, et bien sûr pubkey vérifiée par la signature du document.
- XHX: fonction de déblocage portant sur un mdp. La condition Output=XHX(HASH) est débloquée par un Unlock = XHX(PASSWORD) avec SHA256(PASSWORD)=HASH.

Transaction

- CLTV: condition pour débloquer l'argent après une certaine date. Output=CLTV(TIMESTAMP)
- CSV: condition pour débloquer l'argent après un certain delai. Output=CSV(Delay)

```
Version: 10
Type: Transaction
[...]
Outputs
25:2:CLTV(1489677041)
```

Block

Les transactions sont accumulées au sein de blocks, représentés par des documents Blocks. Ce sont ces documents qui sont échangés entre les noeuds, qui vont vérifier leur validité et choisir lequel inclure dans la blockchain en cas de fork (plusieurs blocks arrivés en même temps avec le même PreviousHash). Ces documents contiennent l'historique des transactions, mais aussi d'autres informations sur l'état du réseau duniter.

Block

Quelques champs importants:

- PowMin: difficulté de la preuve de travail (nb de zéros que doit contenir le début du hash du block)
- UniversalDividend: montant du dividende universel.
- IssuersFrame & IssuersFrameVar: ??? j'ai pas compris
- PreviousHash: hash du block précédent
- Parameters: paramètres du protocole (voir section)

```
Version: VERSION
Type: Block
Currency: CURRENCY
Number: BLOCK_ID
PowMin: NUMBER_OF_ZEROS
Time: GENERATED_ON
MedianTime: MEDIAN_DATE
UniversalDividend: DIVIDEND_AMOUNT
UnitBase: UNIT_BASE
Issuer: ISSUER_KEY
IssuersFrame: ISSUERS_FRAME
IssuersFrameVar: ISSUERS_FRAME_VAR
DifferentIssuersCount: ISSUER_KEY
PreviousHash: PREVIOUS_HASH
PreviousIssuer: PREVIOUS_ISSUER_KEY
Parameters: PARAMETERS
MembersCount: WOT_MEM_COUNT
Identities:
PUBLIC_KEY:SIGNATURE:I_BLOCK_UID:USER_ID
...
Joiners:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Actives:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Leavers:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Revoked:
PUBLIC_KEY:SIGNATURE
...
Excluded:
PUBLIC_KEY
...
Certifications:
PUBKEY_FROM:PUBKEY_TO:BLOCK_ID:SIGNATURE
...
Transactions:
COMPACT_TRANSACTION
...
InnerHash: BLOCK_HASH
Nonce: NONCE
BOTTOM_SIGNATURE
```

Block

Quelques champs importants:

- Identities : pas bien compris l'intérêt puisque les identités non vérifiées sont censée rester dans la sandbox…
- Joiners, Actives, Leavers : concerne respectivement les identités fraîchement acceptées, les identités dont la certif est renouvelée, et les identités quittant le réseau.

```
Version: VERSION
Type: Block
Currency: CURRENCY
Number: BLOCK_ID
PoWMin: NUMBER_OF_ZEROS
Time: GENERATED_ON
MedianTime: MEDIAN_DATE
UniversalDividend: DIVIDEND_AMOUNT
UnitBase: UNIT_BASE
Issuer: ISSUER_KEY
IssuersFrame: ISSUERS_FRAME
IssuersFrameVar: ISSUERS_FRAME_VAR
DifferentIssuersCount: ISSUER_KEY
PreviousHash: PREVIOUS_HASH
PreviousIssuer: PREVIOUS_ISSUER_KEY
Parameters: PARAMETERS
MembersCount: WOT_MEM_COUNT
Identities:
PUBLIC_KEY:SIGNATURE:I_BLOCK_UID:USER_ID
...
Joiners:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Actives:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Leavers:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Revoked:
PUBLIC_KEY:SIGNATURE
...
Excluded:
PUBLIC_KEY
...
Certifications:
PUBKEY_FROM:PUBKEY_TO:BLOCK_ID:SIGNATURE
...
Transactions:
COMPACT_TRANSACTION
...
InnerHash: BLOCK_HASH
Nonce: NONCE
BOTTOM_SIGNATURE
```

Block

- InnerHash: le hash de tous les champs précedents de ce block. Utilité : vérifier qu'un même block n'a pas été posté deux fois ? A vérifier
- Nonce: champ avec une valeur à trouver pour que BlockHash commence par PowMin zeros.

```
Version: VERSION
Type: Block
Currency: CURRENCY
Number: BLOCK_ID
PoWMin: NUMBER_OF_ZEROS
Time: GENERATED_ON
MedianTime: MEDIAN_DATE
UniversalDividend: DIVIDEND_AMOUNT
UnitBase: UNIT_BASE
Issuer: ISSUER_KEY
IssuersFrame: ISSUERS_FRAME
IssuersFrameVar: ISSUERS_FRAME_VAR
DifferentIssuersCount: ISSUER_KEY
PreviousHash: PREVIOUS_HASH
PreviousIssuer: PREVIOUS_ISSUER_KEY
Parameters: PARAMETERS
MembersCount: WOT_MEM_COUNT
Identities:
PUBLIC_KEY:SIGNATURE:I_BLOCK_UID:USER_ID
...
Joiners:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Actives:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Leavers:
PUBLIC_KEY:SIGNATURE:M_BLOCK_UID:I_BLOCK_UID:USER_ID
...
Revoked:
PUBLIC_KEY:SIGNATURE
...
Excluded:
PUBLIC_KEY
...
Certifications:
PUBKEY_FROM:PUBKEY_TO:BLOCK_ID:SIGNATURE
...
Transactions:
COMPACT_TRANSACTION
...
InnerHash: BLOCK_HASH
Nonce: NONCE
BOTTOM_SIGNATURE
```

Block

Voici les paramètres du protocole, inclus dans le champ “Parameters” des documents Block.

Parameter	Goal
c	The %growth of the UD every `[dt]` period
dt	Time period between two UD.
dtReeval	Time period between two re-evaluation of the UD.
ud0	UD(0), i.e. initial Universal Dividend
udTime0	Time of first UD.
udReevalTime0	Time of first reevaluation of the UD.
sigPeriod	Minimum delay between 2 certifications of a same issuer, in seconds. Must be positive or zero.
msPeriod	Minimum delay between 2 memberships of a same issuer, in seconds. Must be positive or zero.
sigReplay	Minimum delay between 2 certifications of a same issuer to a same receiver, in seconds. Equals to `msPeriod`.
sigStock	Maximum quantity of active certifications made by member.
sigWindow	Maximum delay a certification can wait before being expired for non-writing.
sigValidity	Maximum age of an active signature (in seconds)
sigQty	Minimum quantity of signatures to be part of the WoT
idtyWindow	Maximum delay an identity can wait before being expired for non-writing.
msWindow	Maximum delay a membership can wait before being expired for non-writing.
xpercent	Minimum percent of sentries to reach to match the distance rule
msValidity	Maximum age of an active membership (in seconds)
stepMax	Maximum distance between each WoT member and a newcomer
medianTimeBlocks	Number of blocks used for calculating median time.
avgGenTime	The average time for writing 1 block (wished time)
dtDiffEval	The number of blocks required to evaluate again `PoWMin` value
percentRot	The percent of previous issuers to reach for personalized difficulty
txWindow	`= 3600 * 24 * 7`. Maximum delay a transaction can wait before being expired for non-writing.

Peers

Etant donné que seuls les membres du Wot peuvent écrire dans la blockchain, il faut garder une table des adresses des pairs pour pouvoir leur envoyer sans ambiguité les blocks. On ajoute les pairs à cette tables via l'envoi de document Peers:

```
Version: VERSION
Type: Peer
Currency: CURRENCY_NAME
Issuer: NODE_PUBLICKEY
Block: BLOCK
Endpoints:
END_POINT_1
END_POINT_2
END_POINT_3
[...]
```