# Appendix A: functional roles and participants

**Roles and participants**. DLT Technology: Hyperledger fabric. This appendix lists all functional roles identified on TheFundschain platform to support all business scenarios.

## Terms and definitions used

* TFDC: TheFundsChain platform
* CSD: Central Securities Depositary
* Certificate: a X509 certificate
* E-Cert: Enrollment Certificate
* T-Cert: Transaction Certificate
* CA: Certificate Authority: a Certificate Authority (CA) is in charge of managing identities on the platform. There may be several such CA’s in a complex set up
* Intermediate CA: CA’s may be organized hierarchically
* MSP: Membership Service Provider. An abstraction of identity for Hyperledger Fabric.
* H.F: Hyperledger Fabric
* Channel: a partition of the H.F DLT network
* PKI: public key infrastructure
* SPKI: simple public key infrastructure (for attribute certificates)

## High level requirements for TheFundsChain

The notion of functional role is specific to TheFundsChain: it is an applicative notion, independent from Hyperledger native roles. What is the purpose?

1. Provide a fine-grained authorization mechanism for chaincode operations, at the method level rather than a the chaincode level (e.g. “readers” and “writers” is not enough).
2. Provide a mean for a regulator to charter organizations with certain business roles, e.g. CACEIS France is a chartered bank allowed to operate in France as an investor’s bank, a fund custodian and a fund transfer agent: in this sense, functional roles are termed “regulatory roles” (i.e. defined within the context of a particular jurisdiction).
3. Provide a mean for an endorsing peer, within the execution of a common chaincode, to branch to specific processing according to its role (e.g. the account holder of an investor may want to perform some external checks to its own view of the investor cash position). Thus, for a given fund, all endorsing peers get an accurate view of what specifically is required from them. In this sense, functional roles are termed “contractual roles” (i.e. defined within the context of a particular chaincode).
4. There a many functional roles (to date, more than 20 and counting…). Ideally, they should be organized as a hierarchy to allow for further refinement without disrupting existing registered identities.
5. Functional roles may be delegated to a third party. This delegation must be traceable.

## High level Hyperledger Fabric solution outlines

1. **Identify management: more or less validated design**
   1. An H.F “channel” equates to a jurisdiction (sub-jurisdictions may or may not get their own channel, depending on further requirements).
   2. An H.F Certificate Authority is an instance of the H.F C.A. implementation
      * H.F CA is okay for testing purposes
        + More demanding deployments would require something like EJBCA
        + Hard to say for now, since H.F CA provides unique features:
          - Integrated nodeJS client
          - T-certs
          - Attributes (since 1.0.2)
      * In particular, H.F CA does not support certificate revocation protocol (OCSP)
        + CRL are supported in 1.0.2. See roadmap for OCSP
      * H.F CA does not support Attribute Certificates
        + Since 1.0.2, attributes are supported but not SPKI A-Certs
        + Idemix certificates annouced
      * H.F C.A. will be deployed in its simplest setup: no LDAP, SQLLite. TLS communication shall be required.
   3. TheFundsChain maintains one root CA to register all participants and deploys subordinated intermediate CA’s for each distributor, to register: (i) individual nodes belonging to a participant (MRPs) and (ii) investors sponsored by a distributor.
   4. Using intermediate CA’s for identifying end-investors avoid the need for a central “know-everything” authority. Distributors would feel more comfortable managing the identities of their own customers. Intermediate C.A’s provide a simple identity brick to the more general KYC problem. Certificate don’t make for KYC by themselves.
   5. All participants to TheFundsChain must obtain an identity from their regulator to operate on a given channel (i.e. a jurisdiction realm)
2. **ACL management: attribute certificates**
   1. No solution as of H.F. V1 beta
   2. ACL support is a V1 GA promise – v0.6 early design with attribute certificates has been ditched
   3. Our requirements define indeed some sort of fine-grained ACL (method level, independent from H.F roles, independent from CA)
   4. As for V1 beta, we checked the designs of:
      * H.F C.A.: attributes support is a joke (purely internal to the CA, to binding with the certificate)
      * H.F channel policies: more or less hardcoded for internal H.F use (e.g. orderer nodes) – no ready to be extended to our goals.
      * H.F E-Certs: could be, but would require a lot of custom code to build attributes in E-Cert (custom CA, custom verification)
      * V0.6 ACL nascent ACL support (and specification) has gone deprecated (was illustrated assetManagement.go example)
      * Fabric Composer has built an interesting ACL support at the applicative level (full of embedded JS instructions, you might imagine). Perhaps could be reused as a starting point.
   5. H.F v1.0.2 (or 1.0.3) Fabric CA provides attribute support
   6. H.F v1.1 (?) will support idemix credentials, similar to SPKI attribute certificates
   7. The H.F community seems to have messed up quite a bit regarding this critical feature
      * Uncoordinated work between CA and Fabric teams
      * “Out­ of the box” magical channel policy system as a last minute release, which simply does not fit
   8. V0.6 design gone to the scrap yard
   9. … it just looks like the guys have lost the objective
   10. Wonder what they are about to for V1 GA. Must inquire and investigate further
   11. We need a SKPI to generate attribute certificates
       * For testing purpose: strongswan SPKI is available on Linux as a command line tool
       * Spki golang library provides some low level support
       * A-cert validation methods must be hand-coded: didn't find any existing piece of code to sort out attributes from a given a-cert

## High level issues

After some preliminary study, we think convenient to use attributes set in the X509 certificate to model our functional roles. This choice raises some questions that are still open.

To follow RFC recommandations, we shall split responsibilities:

* CA is in charge of identity management (i.e. delivering e-certs)
* SPKI is in charge of authorization management (i.e delivering a-certs)

1. **Attribute inheritance.** If the regulator delegates some of its CA registration privileges, does the H.F CA checks for attribute inheritance? Example: CACEIS has been registered by the regulator as an organization with a number of functional roles. Now CACEIS needs to register 100 peers to operate its business (each one becomes a H.F MSP ready to operate a chaincode, either as a peer or an ordering node).

Short answer: H.F. CA does not check delegation for custom attributes beyond the ones it currently uses (i.e. affiliation, identity type and hf.\* attributes). Without customizing the CA ourselves, the regulator would have to register every single MSP on its channel. Our target is that the regulator registers an organization owner then delegates the details of peer registration to this owner.

1. **Attribute extraction from certificates.** In order to check for one or several functional roles, a peer (within the context of a chaincode) or a client (within the context of a SDK application) will have to retrieve the certificate of: (i) himself (who am I?), (i) the transaction submitter (who is he?) or (iii) other peers of the current chaincode (who are they?). This begs the question of how we may do that in (i) H.F Golang framework and (ii) in node SDK

Short answer: H.F does not provide primitives to do that directly: this has to be part of our custom library (presumably built on top of spki golang package).

1.0.2/1.0.3 (?): API extension to check attributes

Note: our preliminary design used to involve a chaincode (“Party”) to manage the vector of attributes, as a chaincode database. We don’t know how it is going to be needed, even after V1 GA. Anyhow, the “Party” chaincode will still be useful to manage the organization registration/agreement process (a documentary & regulatory process). The result of this process will be precisely the registration of an attribute certificate (a-cert).

This application would be able, in particular, to accurately keep track of delegation patterns, an important issue which is barely addressed by certificates (at least by H.F CA: it seems to be better with more advanced Cas such as EJBCA).

Just in case: CA customization could be performed here: <https://github.com/hyperledger/fabric-ca/blob/master/lib/serverregister.go>

# Conclusion regarding an implementation

1. Too early to engage in some hard-core developments
2. Wait and see V1 GA
3. Try to do best we can in between, perhaps with transaction/chaincode metadata.

# Meaning and purpose of roles

Each party interacting with the platform may engage in a number of actions. The various sets of actions available to parties are stereotyped by roles.

There are different types of roles to be considered, depending of which subpart of the platform a party is interacting with.

1. Roles regarding a Certificate Authority: registering, enrolling and revoking parties.
2. Roles regarding channel policy administration: a channel defines a partition of the DLT, say a jurisdiction or a market place. Policies are defined on a channel to figure out parties engaged in the maintenance of the DLT state.
3. Roles defining interactions with a given application (chaincode or SDK application), aka functional roles.

Roles of type (1) are predefined by the CA. Hyperledger CA manages the following roles, using certificate attributes:

* hf.revoker : capability to revoke certificate
* hf.Registrar.Roles : capability to register identities according to specific types
* hf.Registrar.DelegatedRoles : couldn’t figure out use case
* hf.IntermediateCA : capability to act as an intermediate Certificate Authority

Note: that Hyperledger CA is completely independent from the Hyperledger Fabric.

Identities may be attributed an arbitrary type, defining identity registration domains (i.e. a CA registrar may rule over only a set of identity types). Types may be customized. Examples of identity types: peer, user, validator…

Note: it looks like the identity type is not currently used by H.F. Policy channel groups are used instead to verify channel policy. The idea is interesting however.

User identities vs Peer identities: couldn’t figure out, really. C.A “identity type” is irrelevant (totally disconnected from the H.F ledger). It is probably dealing with channel policy groups.

Roles of type (2) are predefined by Fabric (channel policy module), in order to split Principals (i.e. organizations) into groups. Policies regulate the participation to Hyperledger Fabric transaction validation mechanisms (endorsement, ordering).

* Orderers : participate the ordering consensus after transaction endorsement
* Application : couldn’t figure out

Orderers, Readers and Writers seem to be hard-wired in the solution, at least regarding orderers.

Groups are further refined with policies: Admin, Readers, Writers. Thus, a given Principal may be affected to a pair (group, policy).

During endorsement and ordering, a policy will evaluate according to 2 different (predefined) roles: member or admin.

Roles of type (3) are defined by our functional needs. Roles could be affected to parties using certificate attributes (or any other ACL mechanism provided at the application’s level).

# Detailed requirements for TheFundsChain

## General concepts with functional roles

Classification of identified parties:

* Participating members: first class citizens on the DLT, able to deploy and operate chaincodes
* Sponsored members: second class citizens on the DLT, with SDK capabilities only
* Non-member identified parties: identified parties which have to operate through a member (e.g. retail investors)

Notes:

* The investor acts as a SDK-only participant [exception: institutional investors may be participants
* The fund issuer is necessarily a peer participant
* A fund manager is not necessarily a participant (e.g. in the context of a fund management delegation)
* All other roles may either played by peer participants or via SDK-based interfaces, depending on the scenario

*Relationship with HL identity*

All participating members have a valid MSP identity on at least a channel. Sponsored members (e.g. corporate investors, some service providers…) must be known (KYC sense) to at least one participating member. Non-member identified parties do not need a MSP.

However, our registries (FUND or PTF° do require every party to be identified, so we’ll need a certificate for everybody.

Roles for any given MSP are defined according to type (1), (2) and (3) roles explained above.

A given MSP relates to a single fabric node. A single node may have several roles (“role stacking”). Several nodes may have the same role (redundancy setup).

TFDC functional roles differ from MSP roles, which are limited to a member’s role regarding identity management (i.e. CA admin or CA member).

TFDC roles defined a notion orthogonal to MSP organizations and organization units.

Examples:

MSP organization: CACEIS France

MSP organization unit: CACEIS France – Fund Custody Services

MSP organization CA admin: CACEIS Group Certificate Authority

MSP organization member: CACEIS France – Fund Custody Peer #0998

Similar, but distinct, member declaration:

MSP organization member: CACEIS France – Fund Custody Peer #0997

Functional roles attributed to members:

* CACEIS France – Fund Custody Peer #0998: { juridisdiction=’France’; roles=[‘fundCustodian, ‘fundAccountant’, ‘paymentAgent’ ]}
* CACEIS France – Fund Custody Peer #0997: { juridisdiction=’France’; roles=[‘fundCustodian, ‘fundAccountant’, ‘paymentAgent’,’transfertAgent’ )}

*Investors*

Investors are identified (that’s the whole point with this platform anyway), and the shared registry provides – in effect – a type of securities under a registered form. The French “bearer” form is tweaked into an “identifiable bearer” form, with which the issuer is entitled to identify its shareholders (up to which point, that remain to be defined).

We must distinguish two categories of investors:

* Non-member investors: no MSP required, an identity is required, central identity management must be avoided
* Member investors (large corporations, funds of funds)

*Fiduciary & regulatory responsibilities*

Our consensus design must ensure that all involved actors abide by their fiduciary and regulatory responsibilities. This depends of course on the jurisdiction. Besides, each party has its own operation requirements.

In our setup, the same responsibilities are collectively assumed by all peers. Therefore, from a legal standpoint, each actor fulfils its regulatory requirements (the “shared” registry is also “its” registry).

Technically, each endorsing peer could execute only a part of the transaction endorsement code (if allowed to retrieve its current functional role). This could also have a legal meaning (e.g. endorsing peers with disjoint roles don’t mess with their peer’s responsibilities).

## Role stacking

One single party may act with several roles. The way disintermediation is achieved is precisely by stacking roles on a single party. Depending on the regulation, stacking roles may be restricted. Each platform scenario shall define how roles may be staked. Stacking roles in different ways, together with a different chaincode deployment scheme, results in a different business scenario. Our design is intended to accommodate many such different business scenarios.



Figure 1: Sample graphical representation of a platform scenario

## Role delegation

It is important to support role delegation. For instance, some service company may be allowed to act, for some workflows, on behalf of a regulator. This feature is essential to start smooth operations while not every actor is actually participating to the network.

Example:

* DLT Shift France – Custodian Mirror #0126: { juridisdiction=’France’; roles={

{‘CUST’, onbehalfof = {‘Bank Of America Mellon’} },

{’ACC’, … }

{’PAG’ }, … }

}

## Role attribution

Roles are not freely available to actors. Available roles for a given actor are (mostly) regulated.

For instance, to act as a bank account holder, an actor may probably have to be a registered bank. A directory of platform participants must therefore be deployed to manage authorized roles for any given party, under a given jurisdiction. This feature is supported thanks to the PARTY chaincode.

Note: Retail investors (sponsored members) are not listed by the PARTY chaincode (one should have a distributor-linked investor repository).

Regulated roles

For a given jurisdiction, organizations or organization units are allowed to act with a role by the domestic regulator.

For each jurisdiction, the regulator maintains a registry of organizations, organization units or members with such an authorization.

Contractual roles

For a given fund, the issuer contracts with some other organizations their role regarding the administration of the fund (i.e. custody & administration agreement). These definitions are part of the fund chaincode and defined either at deploy time or at run time.

**TheFundsChain design choices**

There is a requirement to manage functional roles at the application level. This means:

* Identifying various parties (name, address, contact…)
* Attributing each party a vector of allowed roles x jurisdictions
* Attributing each party a public and a private key for signatures
* Identifying role delegations to service providers
* Adding the signed seal generated by a regulator for regulated roles
* Seals may expire
* Ideally, producing a single homomorphic hash any party could exhibit to a peer to instantly verify its (role, jurisdiction) pairs. Since the “party” chaincode is always available locally, this is just crypto sugar…
* Beside regulated roles (who is entitled to do what), define chaincode specific contractual roles (in the context of this chaincode, who own a given role).

We consider 3 options below: 1 applicative (fully custom), 1 for V0.6 (looks like it has been deprecated in V1, maybe it will reenter in V1.1) and 1 for V1. The latter sounds most promising.

For convenience and adaptability, functional roles should be defined as a hierarchy (e.g. “LEI authority” is a subrole of “regulator”).

**Appendix: platform participants and functional roles**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role trigram code** | **Role** | **Responsibilities** | **Operational notes** | **Legal notes** |
| ACC | Fund accountant | Compute NAVs  Establishes fund’s fees |  |  |
| CLR | Clearer | Clear Delivery vs Payment transactions | Guarantees D against P |  |
| CSH | Cash account holder | Investor’s bank account holder  Acknowledge cash transfers to and from investors | Subscriptions: verify beforehand that the cash is on the INV account  Redemption: verify that the cash has arrived on the INV account | Party must be registered as a bank |
| CTK | Caretaker | Certifies smart contracts | Operate a secure, common time service (allows for secure cut-off checks) | Authorizes the deployment of certified chaincodes |
| CUS | Fund custodian | Acknowledge cash transfers to and from investors  Keep the record of a fund’s asset |  | In the simplified context of feeder funds, the full role may also be managed by TFDC |
| DIS | Distributor | Sponsors investors  Collects and certifies KYC data about investor | DIS may be an issuer (DLT direct) or a Fund Manager (funds investing in funds)  In case of a direct investment, orders are posted to the TAG | May be the issuer  There may be several DIST stacked for a given order |
| FMG | Fund Manager | The principal fund manager. Secondary FMG, e.g. in charge of delegated fund carve-outs are not identified | May be the issuer | FR: contributes NAV validation consensus, possibly with some delay |
| INV | Investor | Initiates orders (subscriptions or redemptions) | INV may be a participant to *TFDC* or simply accessing it as a sponsored participant  INV may be a Fund Manager | May be a physical person, a company, or another fund |
| ISR | Issuer (or promoter) | Registers funds with regulator |  |  |
| LEI | Legal Entity identification authority | Delivers a LEI whenever a new fund is registered | FR: INSEE |  |
| PAG | Paying Agent | Transfers dividend to investor |  | Likely merged with TAG under most circumstances. |
| REG | Regulator | Delivers agreements for registration and distribution | Regulator covers a given *jurisdiction* |  |
| RGR | Registrar | Investor’s registry book-keeping  Responsible for issuing and cancelling shares  Shares may be issued in either a bearer form, which proof of ownership is guaranteed by the investor’s bank or a registered form, whose proof of ownership is guaranteed by the registrar (which may be the issuer) | FR: Teneur de compte émetteur  CSD (“Dépositaire Central”) (e.g. Euroclear France). Responsible for maintaining an official view of investor’s custodians position. This view may be aggregated or detailed, depending on the CSD practice. | As the Euroclear-France case stands out, several RGR actor may collaborate  This role may be handled by the ISR, if needed.  LUX: maintains the proof of ownership. Responsible for declaring domestic investors holdings to domestic CSD (Clearstream) |
| SEC | Investor’s securities custodian | Keep a record of securities the investor holds | FR: Teneur de compte conservateur (TCC)  LUX: usually intermediated through a “nominee” | FR : mandatory for securities issued under a bearer’s form |
| SIA | Security Identification authority | Delivers a security code (e.g. ISIN) whenever a new fund is registered | FR: Euroclear FR  LUX: Clearstream |  |
| STL | Settlement system operator | Operates DVP system | Ex: Euroclear, Clearstream |  |
| TAG | Transfer Agent | Captures investors orders, possibly transmitted by “domestic agents” established abroad, triggers irrevocability cut-off and value orders upon NAV reception | FR: Centralisateur | May be the issuer |
| VSP | Value-added service provider |  |  |  |