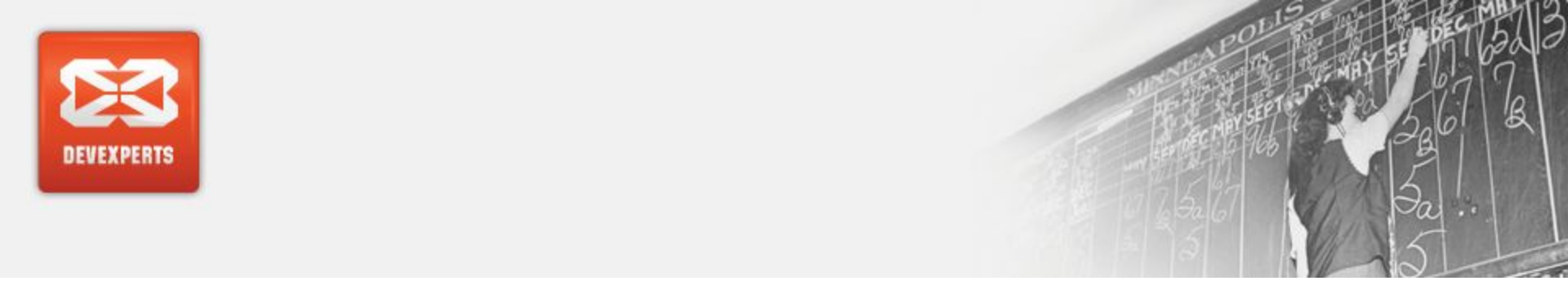

ASP solution: Project goal and the current implementation

dxPlatform release 1: Summer 2015

Presentation plan

- Initial project goal and scope
  - Goal
  - Business requirements
  - Additional input from foreseen platform future
  - What was NOT in scope
- Current implementation: essential concepts
  - Account groups
  - Organizational hierarchy and Broker entity
  - User roles
  - Dealing settings framework

Goal

- To be able to host multiple independent brokers in one dxFX installation.

Business requirements

- The multiple hosted installations mode is supposed to be transparent - or almost transparent - for the customer (broker).
  Each broker is supposed to have:
  - fully functional access to the existing trading administration tools meaning any so-far-system-wide configurations (like quoting) would be expected to become broker-wide
  - client's account's management (per-account settings and profiles) and client's trading monitoring tools (books, activity log, messaging etc.) restricted to access only own clients for the broker - transparently
  - per-broker cumulative metrics for any so-far-system-wide (like exposure)
  - ability to use own independent rules or sequence of account and client's identification; that is, independent namespaces and, possibly, customizable client's on-boarding process
  - *own financing profile (rules for triple swap day etc.), independent set of SWAP rates and rate schedules (commissions) – postponed so far*

Additional input from foreseen needs of multi-asset platform

- Custom hierarchy of account groups with settings inheritance down the hierarchy
- Solve the well-known problem of settings ambiguity (multiple inheritance paths, implicit rules ordering etc.) by controlling invariants at the configuration time

What was NOT in scope

- End-user UI tools to manage account groups:

Groups were introduced as a part of solution design which might be later offered for end-user admins – then the required tools should be developed

- End-user UI to manage brokers (firms), trading accounts and traders such as:
  - creation of new Brokers and permissions configuration for the pre-defined role of Dealer (DPG)
  - creation of dealers / traders logins and permissions configuration

Any users / accounts management cases except of the basic case “create a client with a trading account for a certain broker” were supposed to be handled with the help of support.

Thus, as UI tools we have dxCore console and dxBackoffice UI and provide some means to configure the relations between the two systems manually.

Current implementation: essential concepts

- Account groups
- Organizational hierarchy and Broker entity
- User roles
- Dealing settings framework

Account groups (dxCore & dxDealPro)

Account groups are:

- hierarchical
- a group can be marked with one or many categories, which allows for:
  - category type can be used e.g. for category-based invariants like disjunction
  - payload (category “value”) can be used as a flag (sub-type)

Examples:

- Settings category: a list of settings domains to mark groups created in view of different purposes such as Margining or Execution (orthogonal categorization of clients’ accounts)
- Broker category: Offset or Client accounts groups

Account groups (dxBackoffice)

- Account groups are flat
- Used as markers (flags) for certain BO logic

Ex.: FINANCING

Groups sync dxBackoffice->dxCore

- A business ID of a dxCore'a group can be set as an external code for dxBackoffice group manually
- If such link is configured, dxBackoffice performs **content synchronization** dxBackoffice->dxCore
- Applies to both accounts and principals groups

Organizational hierarchy (dxCore & dxDealPro)

- Organizational hierarchy is implemented on the hierarchy of account groups:
  - “Broker” entities are linked to certain nodes of the groups tree: those, marked with Broker category with a payout Root (RAG)
  - Parent-child links between RAG groups define the hierarchy of Brokers

Broker entity (dxCore & dxDealPro)

- Broker entity:
  - Code, Name, Domain
  - 3 “special” account groups:
    - Broker’s root (RAG) and direct children:
      - Clients accounts (CAG)
      - Offset accounts (BAG)
  - 2 “special” principal groups:
    - Dealers (DPG)
    - Clients (CPG)

Organizational hierarchy (dxBackoffice)

- Organizational hierarchy is implemented as parent-child on “Subjects”
- “Broker” is one of the possible subjects types
- Other relations are:
  - Accounts groups, subject groups, trading accounts are related to nodes within the subjects tree by “ownership” relation

Relations between dxBackoffice & dxCore entities

- Similar approach as for groups: entities on dxBackoffice and dxCore sides are related by entering a dxCore's entity business code as an "external code" of dxBackoffice's entity
- Synchronization procedure dxBackoffice->dxCore for client's logins and accounts exists as a part of "a client on-boarding" scenario
- *Not for a Broker entity yet – planned soon as an extension required by TradeX*

User roles (dxCore & dxDealPro)

- Dealers, Clients
- Implemented as principals groups with “typed links” (DPG, CPG) from a Broker entity
- Used in a few scenarios:
  - dxFX-specific validation rules to detect a type of principal who placed an order
  - “Dealer” role is checked on login to dxDealPro

The scheme is very specific for FX-retail and unsatisfactory for institutional trading cases like TradeX or Alpha projects. A notion of a role as a permissions template should be introduced for further extensions.

User roles in dxBackoffice and relations to dxCore

- dxBackoffice roles are configurable as permissions templates applied to dxBackoffice users
- Those are not in any way related to dxCore's users
- No way to manage dxCore's users' permissions from dxBackoffice
- *A separate management UI is planned for that purpose*

Dealing settings framework (dxCore & dxDealPro only)

- Settings-related category type
- Settings domains
- Profiles
- Definition levels by org hierarchy:
  - SYSTEM-WIDE, BROKER, ACCOUNT
- Definition levels by instruments hierarchy:
  - ROOT, INSTRUMENT (ANY)
- Settings space:
  - {org hierarchy, instruments hierarchy[, conditions]}