The Registry Model

Page Audience: IAM designers, developers, and consumers.
Page Purpose: Describe an important design pattern for middleware services.
Document Status: Incomplete draft

Abstract

There are many kinds of entities that we wish to identify, describe, manage, and share among multiple business functions and application systems at an institution: people, groups, departments, roles, courses, budgets, computers, services, etc. Proper management of information about these entities allows this information to be well-maintained and easily shared among all systems. A registry is a design pattern supporting effective management this kind of information. A successful registry must deal with several issues and choose from a number of deployment options.

Introduction

In a large university there are many academic and business processes supported by hundreds or thousands of IT systems. There are a number of entities ("things", "objects") that are of interest to many of these systems. "People" is the best example. People are users of systems, and are in system data as students in courses, holders of positions as employees, researchers on projects, etc. It is very important to each system using person information that that information be accurate, complete, and up to date. Since most systems share person data with other systems, it is also important to each system that person information be consistent among the set of systems with which data is shared, to avoid mismatches and confusion about correct values. So across the large set of systems there is a need for a body of person information that is sharable, consistent, accurate, complete, authoritative, and up to date.

This same set of requirements applies to other kinds of core institutional entities. Examples include: organizations, courses, computer names and addresses, budgets, roles, computer-based services, groups, system privileges. Historically data about each of these entities has been managed in its own set of business processes and systems, leading to a wide variation of capabilities regarding data flow, ownership, quality control, access, extensibility, etc. As the institution becomes more dependent on a larger number of systems it is compelling to provide a consistent architectural approach to management of institutional entity data to better support its processes.

Requirements

An effective system for management of institutional entities has a number of desirable characteristics. Many of these characteristics are typical requirements for business systems for any business function.

Consistency

Authority:

Accuracy

Timely updates:

consistency, authority, access control, auditability, transparency, distributed operations, transactional control, different update and distribution methods, high volume

Registry Defined

A registry keeps track of all instances of a particular kind of thing, i.e. a set of entities in some context. This document is concerned with the institutional context.

Appropriate "kinds of thing" are those which have:

- unique instances;
- a common set of attributes among instances, including identifying attributes that reliably distinguish one from another;
- enough instances to make it worth keeping track of them, eg more than 100;
- instances that are created and updated on a regular basis (typically several times daily);
- multiple application systems (subscribers) that need to access their info.

An entry in a registry has:

- identifying attributes (identifer(s) and/or name(s));
- management attributes, including
    - created/updated-by, create/update-date
    - manager(s) (aka owner(s))
    - other access control info;
- attributes related to the organizational function of the entry.

The functional attributes are those that are most likely to be shared among multiple systems. An application system will typically maintain information about registered entities that is specific to that system, and remains internal to that system.

A registry provides guarantees about accuracy, completeness, timeliness, and accessibility of its records, appropriate to the needs of its subscribers. Those with registry update capability (maintainers) must all agree to abide by conditions necessary to maintain the service guarantees of the registry.

A registry may be maintained (ie, entries created/updated/deleted) via a single business process or application system, or via multiple processes/systems. Supporting multiple maintenance processes raises several design issues, including whether to extend one process to support all others or creating a new one; reconciling conflicting attributes; etc.

A registry has business logic for maintaining quality of entries.

Examples of institutional registries include:

- persons;
- network names and addresses (i.e., DNS names and IP addresses);
- organizational units;
- groups;
- accounts/user-ids;
- budgets.

Some architectural assumptions:

- Few updaters, more readers
- Relevant to some organizational scope, eg dept person reg may feed univ PR or reverse

random notes

dealing with distributed sources

entries aren't removed, just marked with non-current status

appness: put in stuff that only one app needs?