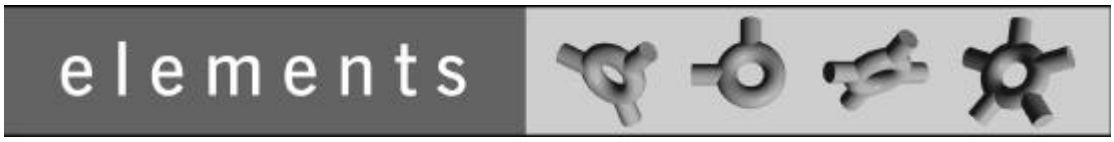

Permissions - Introduction

Table of Contents

Introduction to the Permissions Model

1. Purpose of Document

The purpose of this document is to introduce the Permissions package in the elements object model. This document will provide sufficient background to the business problem to explain and motivate the subsequent description of the model. It will also give an overview of the model, outlining the 'mental picture' that drove the development of the object model, and then summarize the key features of the model, as documented in UML.

The readers of these documents will be:

- Anyone assessing the usefulness of the elements object model for their purpose.
- People preparing to study the associated package in detail.
- Anyone who wants an outline picture of a particular package because it is used in another package.

2. The Business Problem

Within each software component we need to be able to restrict the operations that various entities (such as users and external components) are allowed to perform. This is the process of authorization. We must also be able to confidently identify those entities that have permission to perform restricted operations within our system. This is the process of authentication.

There are many different types of operations that need to be restricted to only authorized entities including, but not limited to, permissions on reference data and permissions on deals. Some examples of permissions in these areas are shown below.

We need to be able to restrict operations associated with reference data based upon three things:

1. What is being done to the reference data?
2. Who created the reference data?
3. What domain was the reference data created in?

Some examples of permissions on reference data are:

- Permission to create reference data.
- Permission to view all reference data.
- Permission to modify all pieces of reference data that you have created.
- Permission to delete all pieces of reference data in the "London" domain.

We need to be able to restrict operations associated with deals based upon a number of the possible aspects of a deal, such as:

1. The book that the deal is made from.
2. The counterparty that the deal is made to.
3. The type of deal that is being made.
4. Whether or not the deal has been done for a particular purpose (such as an automatically generated deal).
5. The currencies or currency pairs of the deal.
6. The securities or equities that the deal is composed of.

Some examples of permissions involving deals are:

- Permission to trade a normal (USD, AUD) FX deal from "Dave's book" to J.P.Morgan bank.
- Permission to trade a normal bond deal from any book to either J.P.Morgan, BZW, or the Hong Kong Bank with any security.

There are also other miscellaneous operations that need to be restricted:

- Permission to open the position screen.
- Permission to perform "End of Day".
- Permission to create a new user.
- Permission to suspend a user's account.
- Permission to grant other entities permission.

It should also be possible for user interfaces to display only the options that the current user is authorized to perform.

We also need to be able to audit all the restricted operations that have been performed, and all the failed attempts to authenticate an entity.

3. Overview of the Conceptual Model

The key concepts in the Permissions model are 'permission owner', 'permission grant' and 'operation'.

3.1. Permission Owners

The focal point of the permissions model is the entity that has permission to do something. In the general area of permissions this entity is referred to as a 'principal', however, to avoid ambiguity with the financial meaning of principal we refer to such an entity as a *permission owner*. A permission owner is any entity that can be given permission to perform an *operation*. Permission owners achieve this by holding a collection of *permission grants*. Examples of permission owners include users and external software components.

A permission owner is also responsible for *authentication* of entities. Thus, for example, if an external process or person wants to be recognized as a particular permission owner, it must submit some authentication information to that permission owner object for validation. Typically this validation process will take the form of password checking, but the model allows for more general types of validation.

PermissionOwners hold an *authenticator* that is used to authenticate permission owners using information transmitted from a secure data source. There are a number of different authentication methods that can be used when the information is received from a secure source, ranging from passphrases to retinal scans. This model includes a passphrase class for such a purpose. A different approach will be required if the authentication information comes from an unsecured source, such as from across the internet. Transmission of data across unsecured lines will be done using strong encryption techniques, the details of which are outside the scope of the permissions package. The encryption technique used will be able to provide authentication of entities.

The permission owner is required to log events for the purpose of auditing. There are currently two different types of information that are logged. The first is the logging of all attempts of an entity to be recognized as a permission owner, regardless of success. The second is the logging of every operation that this permission owner attempts to perform, regardless of success. The associated logging objects determine what information, regarding these events, is sent to the auditor.

3.2. Operations

A permission owner is able to determine whether it is allowed to perform a given *operation*. This operation is defined as an *action* applied to a particular object or set of objects. For example, changing a deal would be an operation whose action is 'modify' and whose object is a deal.

Operations are subdivided into different *types*. The type of an operation is used to simplify the matching of operations, and to specify the possible set of actions for an operation. For example, 'deal operation' is one type of operation, with available actions of 'create', 'modify', 'browse' and 'cancel'.

A *classifier* is used to specify the object to which the operation applies. A classifier is a dictionary of named attributes of an object. For example, a deal classifier could specify the book, deal type, and currency of a deal. Any deal with these characteristics would then satisfy this classifier. Note that classifiers can specify set of values, not just individual values.

In summary, a permission owner can have permission to perform an 'operation'. An operation is an 'action' performed on a set of possible objects. An operation is characterized by an 'operation type', an action, and a specification of the objects to which the action can be performed. This set of objects is specified by a 'classifier'.

3.3. Permission grants

A 'permission grant' specifies a set of operations, and is capable of answering whether it includes a given operation. A permission owner holds a collection of permission grants that together specify the operations that the permission owner is allowed to perform. An individual permission grant is limited to specifying operations of a particular type. It can however specify a set of allowable actions, and it can specify a set of allowable values for each aspect of the operation's object.

For example a single permission grant could specify an operation of type "deal", with the actions "read" and "update", and with an object that has the following attributes: book → {"Jo's Book", "Doug's Book", "Mike's Book"}, counterparty → {"JPMorgan", "BZW", "CitiBank"}, deal type → {FX}, deal purpose → {normal}, currency pairs → {(USD, AUD), (USD, FRF)}.

An operation will be allowed by a permission grant if the type of the operation matches that of the permission grant, the action of the operation matches one of the operations of the permission grant, and the classifier of the operation is a 'subset' of the permission grant's classifier. Classifier A is a subset of classifier B if A's keys are a subset of B's keys, and values corresponding to each of A's keys are a subset of the values corresponding to the same keys on B.