elements

# Object Identity Package

TARMS Inc.

September 07, 2000

Typeset in LaTeX.

## Contents

## List of Figures

## List of Tables

## Package Description

The object identity package contains the basic models for deciding whether two objects refer to the same business object.

As objects are modified, they have a number of versions. Each version contains information as to when the version was created, by whom and for what purpose. Versioning allows an audit trail to be followed for the object, as each version can be saved individually to persistent storage.

# 1 Use Cases

## 1.1 New Object

From an initial object, create a new family of objects with a unique key and an initial version number.

For example: create a new deal object, ready for editing. Once the object has been accepted, assign it a new deal number (key) from a central pool of keys and a version number of 1.

## 1.2 Edit Object

To edit a versioned object, make a copy of the object and modify the copy. When the modifications are complete, the copy is turned into the current version, with a new version number.

## 1.3 Get Object

Get the current version of an object, identified by its key.

## 1.4 Compute Delta

Get the current version of an object, identified by the object key and the previous version to the current version. Compute the differences between the two versions and return the delta.

# 2 Interfaces

## 2.1 Domain

Domains represent the business structure of an organization. An organization is assumed to have a number of semi-independent databases. Each database is located in a separate domain. Ownership, update and modification rights to an object are assigned to a single domain.

Domains have a hierarchical structure, reflecting the large-scale structure of the organization. For example, the Singapore business unit domain is contained within the Asia-Pacific region domain which, in turn, is contained within the Global domain.

### 2.1.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇑ | Comparable | | |
| ↑ | Comparable | | |
| ↓ | DomainModel §4.1 | | |

⇑:Inherits ↑:Realizes ↓:Realized by

### 2.1.2 Operations

**String domainId()** domainId

The domain name. Return a unique identifier for the domain.

**Domain parent()** parent

The parent domain. Return the parent domain of this domain, or nil if this has no parent.

**Boolean equals(Comparable arg)** equals

**arg: Comparable** The object to compare this object against.

The equality relationship. Two domains are equal if their domain identifiers are equal.

## 2.2 Frank

When a versioned object is created or modified, it needs to be franked with information on the details of who made the change and when the change occurred. A Frank groups this information together.

### 2.2.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↓ | FrankModel §4.2 | | |

↓:Realized by

### 2.2.2 Operations

**Responsible performedBy()** performedBy

The party which made the change. Return the party responsible for the creation of the Frank.

**Datestamp performedOn()** performedOn

Processing date of change. Returns the processing date on which this change was made. Note that the processing date may not be the same as the actual date.

**Timestamp performedAt()** performedAt

Time of change. Returns the date and time at which the change occurred.

**String action()** action

The action that created this Frank. Returns a string description of the change performed.

## 2.3 Keyable

Keyable objects have a unique key that can be used to retrieve an object from a Depot §3.1 or to see whether two objects refer to the same logical object or family of objects.

Each keyable object is associated with a Domain §2.1 that acts as an authority for the object's key. Domains allow key spaces to be partitioned.

### 2.3.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ⇑ | PartiallyOrdered | | |
| ⇓ | Version §2.4 | | |

⇑:Inherits ⇓:Inherited by

### 2.3.2 Operations

**Domain domain()** domain

Domain of the supplied key. Returns the domain that this key was issued under.

**Integer key()** key

The object key. Returns the integer that, within the domain that supplied the key, uniquely identifies that object or versioned family.

**Boolean equals(Comparable arg)** equals
**arg: Comparable** The object to compare this object against.

The equality relationship. Two Keyables are equal if their domains and keys are equal.

**Boolean lessThanOrEqualTo(PartiallyOrdered arg)** lessThanOrE-qualTo
**arg: PartiallyOrdered** The object to compare this object against.

The less than or equal to relationship. Keyables from different domains are not orderable. Within a single domain, Keyables are ordered by identifier number.

## 2.4 Version

A Version represents a group of objects, each representing a sequence of versions of that object. An example is a deal, with each deal modification creating a new version.

### 2.4.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| $\Uparrow$ | Keyable §2.3 | | |
| $\downarrow$ | VersionModel §4.3 | | |
| $\leftrightarrow$ | DepotModel §5.1 | store | ◊ |

$\Uparrow$:Inherits $\downarrow$:Realized by $\leftrightarrow$:Association $\rightarrow$:Navigable ◊:Aggregate ⧫:Composite

### 2.4.2 Operations

**Integer versionNumber()** versionNumber

Version number. Returns the version number associated with this object.

**Version predecessor()** predecessor

Previous version to this version. Returns the previous version to this version, or nil if this is the first version.

**Version successor()** successor

Next version to this version. Returns the next version to this version, or nil if this is the most recent version.

**Boolean isCurrent()** isCurrent

Current version? Returns true if this is the current version of the object. The current version is the version with the largest version number.

**Frank created()** created

Creation frank. Returns the Frank for the first version of this family of objects.

**Frank modified()** modified

Modification frank. Returns the Frank associated with the creation of this version of the family of objects. If this is the first version, then this frank is the same as the creation frank.

**Boolean equals(Comparable arg)** equals

**arg: Comparable** The object to compare this object against.

The equality relationship. Two Versions are equal if their domains, keys and version numbers are equal.

**Boolean lessThanOrEqualTo(PartiallyOrdered arg)** lessThanOrEqualTo

**arg: PartiallyOrdered** The object to compare this object against.

The less than or equal to relationship. Versions from different domains are not orderable. Within a single domain,Versions are ordered by key and then by version number.

## 3 Architectural Service Interfaces

### 3.1 Depot

A Depot object is one that maintains a cache of versionable objects so that the latest version of that object can be returned.

Depots implicitly act as an intermediary between whatever persistent store scheme is used and the object model. Requests to the cache can cascade as requests to persistent store, updates to the cache can be written through to the store. The exact implementation of Depots (and their subclasses) is dependent upon the underlying architecture of the system. This specification provides the Depot interface from the point of view of objects within the `elements` object model; a place where objects can be stored and retrieved.

### 3.1.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↓ | DepotModel §5.1 | | |
| ↔ | VersionModel §4.3 | depository | |

↓:Realized by ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

### 3.1.2 Operations

**Version currentVersion(Keyable key)** currentVersion
**key: Keyable** The key associated with this versioned family of objects.
**Raises:** NotFoundException

Current version of an object. Returns the most current version of the object with this key. A NotFoundException is raised if the key can not be found.

**Version previousVersion(Keyable key)** previousVersion
**key: Keyable** The key of the object in cache.
**Raises:** NotFoundException

Pre-current version of object. Returns the version previous to the most current version of the object keyed by key. A NotFoundException is raised if there is no such version.

**Version nextVersion(Keyable key)** nextVersion
**key: Keyable** The key of the object in cache.

New version of object. Returns the object that would be the next version to the most current version of the object. A copy of the current version of the object is returned, with the version number updated.

**initializeFamily(Version unversioned)** initializeFamily
**unversioned: Version** The object to which a new version should be assigned.

Create a new version family. Get a new version with a new unique key for the current domain and a suitable initial version number and assign it to unversioned.

**add(Version newVersion, String action, Responsible performer)** add
**newVersion: Version** The new version to add to the cache.
**action: String** The action that has caused the new version to be created or updated.
**performer: Responsible** The party responsible for the update.
**Raises:** OutOfOrderException

Add a new version to the depot. Adds a new version of the object to the depot. This object is associated with an updated frank containing the performer, action, the current processing date and now timestamp. If this is the first version of this object, then the object is associated with a created frank identical to the updated frank.

An OutOfOrderException is raised if the newVersion has an earlier version than the current version.

**Reportable validate()** validate

Validate the depot.

A depot is valid if: for each object with a common identifier held by the depot, the only object which returns true to isCurrent() is the object with the maximum version number.

# 4 Classes

## 4.1 DomainModel

The DomainModel class provides a concrete implementation of the Domain interface. Domains can be viewed as dot-separated strings, with the leftmost domain indicating the most general domain[1]. For example RoboBank.Asia.Singapore refers to the Singapore office of the Asia region of the RoboBank organization.

### 4.1.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↑ | Domain §2.1 | | |
| ↑ | Validatable | | |
| ↔ | VersionModel §4.3 | within 0..n | |

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

### 4.1.2 Attributes

**domainId: String** A string giving the domain identifier. A valid domain attribute is a string which matches the following regular expression:

[1]Domains are similar in structure to Rendezvous[1] subjects.

```
[A-Za-z_][A-Za-z0-9_]*(.[A-Za-z_][A-Za-z0-9_]*)*
```

### 4.1.3 Operations

**Domain parent()** parent

The parent domain. The parent domain is computed by removing the rightmost string matching the regular expression .[A-Za-z_][A-Za-z0-9_]* from the domain attribute and constructing a new domain with the reduced attribute.

If there is no string that can be removed (ie. we are at a top-level domain) then nil is returned.

**Reportable validate()** validate

Validate the domain. A domain is valid if the domain attribute matches the regular expression given in the domainId attribute documentation.

## 4.2 FrankModel

The FrankModel class is a concrete implementation of the Frank interface.

### 4.2.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↑ | Frank §2.2 | | |
| ↑ | Validatable | | |
| ↔ | Responsible | performer 1..1 | → |
| ↔ | VersionModel §4.3 | created 1..n | |
| ↔ | VersionModel §4.3 | modified 1..1 | |

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

### 4.2.2 Attributes

**action: String** The action that caused this frank to come into being.

**performedOn: Datestamp = Datestamp.currentProcessingDate()** The processing date on which the object was created or modified.

**performedAt: Timestamp = Timestamp.now()** The date and time at which the franked object was created or updated.

### 4.2.3 Operations

**Responsible performedBy()** performedBy

The party responsible for the update. Return the associated performer.

**Reportable validate()** validate

Validate the object. A FrankModel is valid if all of the action, dates and times and performer are available. If the time the update was performed at comes before the date the update was performed on, then a warning should be issued.

## 4.3 VersionModel

A VersionModel is a concrete implementation of the Version interface.

Any class that uses versions should hold a VersionModel attribute. The class should also implement the Version interface itself, with the methods simply passing through to the attribute.

### 4.3.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↑ | Version §2.4 | | |
| ↑ | Validatable | | |
| ↔ | FrankModel §4.2 | created 1..1 | → |
| ↔ | FrankModel §4.2 | modified 1..1 | → |
| ↔ | DomainModel §4.1 | within 1..1 | → |
| ↔ | Depot §3.1 | depository | → |

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ♦:Composite

### 4.3.2 Attributes

**key: Integer** The key for the object family.

**versionNumber: Integer** The version number for this object.

### 4.3.3 Operations

**Version predecessor()** predecessor

Previous version to this version. Get the previous version of this object from the associated depot.

**Version successor()** successor

Next version to this version. Get the successor of this version from the associated depot.

**Boolean isCurrent()** isCurrent

Current version? Return true if the associated depot returns this object as the current version of this object family.

**Frank created()** created

Creation frank. Return the associated creation Frank.

**Frank modified()** modified

Modification frank. Return the associated modified Frank.

**Domain domain()** domain

Domain of identification. Return the associated domain that this object is within.

**Reportable validate()** validate

Check for for a valid object. A VersionModel is valid if the creation processing date and timestamp are the same as or earlier than the modification processing date and timestamp.

## 5 Architectural Services

### 5.1 DepotModel

The DepotModel class is an implementation of the Depot interface and provides basic caching services for objects that use the Version §2.4 interface. The Version objects accessed by the cache are the actual objects that the version is attached to, rather than the versions themselves.

The exact implementation of the operations defined in Depot is a function of the underlying architecture of the system — something not addressed by the elements object model. As a result, no semantics have been defined for the Depot operations.

#### 5.1.1 Relationships

| | Class | Description | Notes |
|---|---|---|---|
| ↑ | Depot §3.1 | | |
| ↔ | Version §2.4 | store | → |

↑:Realizes ↔:Association →:Navigable ◊:Aggregate ⧫:Composite

# 6 Exceptions

## 6.1 OutOfOrderException

An exception raised when a versioned object appears to have an incorrect version.

#### 6.1.1 Operations

**Version updatingVersion()** updatingVersion

Version attempting an update. Returns the object that was being presented as the newest version.

**Version existingVersion()** existingVersion

The existing version which the updating version clashes with. Returns the existing version that is on or after the updating version.

# 7 Associations

Table 1: Object Identity— Associations

| Association | | | | |
|---|---|---|---|---|
| | Role | Class | Card. | Notes |
| performer | | | | |
| | performer | Responsible | 1..1 | → |
| | frank | FrankModel §4.2 | 0..n | |
| created | | | | |
| | creator | FrankModel §4.2 | 1..1 | → |
| | version | VersionModel §4.3 | 1..n | |
| modified | | | | |
| | modifier | FrankModel §4.2 | 1..1 | → |

Table 1: . . . continued

| Association | | | | |
|---|---|---|---|---|
| | Role | Class | Card. | Notes |
| | version | VersionModel §4.3 | 1..1 | |
| store | | | | |
| | depositor | Version §2.4 | | → |
| | depot | DepotModel §5.1 | | ◊ |
| within | | | | |
| | domain | DomainModel §4.1 | 1..1 | → |
| | version | VersionModel §4.3 | 0..n | |
| depository | | | | |
| | depot | Depot §3.1 | | → |
| | depositor | VersionModel §4.3 | | |

→:Navigable ◊:Aggregate ♦:Composite

## 7.1 performer

**Role: performer** *Navigable* Responsible, 1..1.
**Role: frank** FrankModel, 0..n.
The party responsible for the change.

## 7.2 created

**Role: creator** *Navigable* FrankModel, 1..1.
**Role: version** VersionModel, 1..n.
The creation action for this family of objects.

## 7.3 modified

**Role: modifier** *Navigable* FrankModel, 1..1.
**Role: version** VersionModel, 1..1.
The modification frank for this version of the object family.

## 7.4 store

**Role: depositor** *Navigable* Version.
**Role: depot** *Aggregate* DepotModel.
The objects that make up the depot.

## 7.5 within

**Role: domain** *Navigable* DomainModel, 1..1.
**Role: version** VersionModel, 0..n.
The domain that was the authority for the identifier for this version.

## 7.6 depository

**Role: depot** *Navigable* Depot.
**Role: depositor** VersionModel.
The depot that holds these objects.

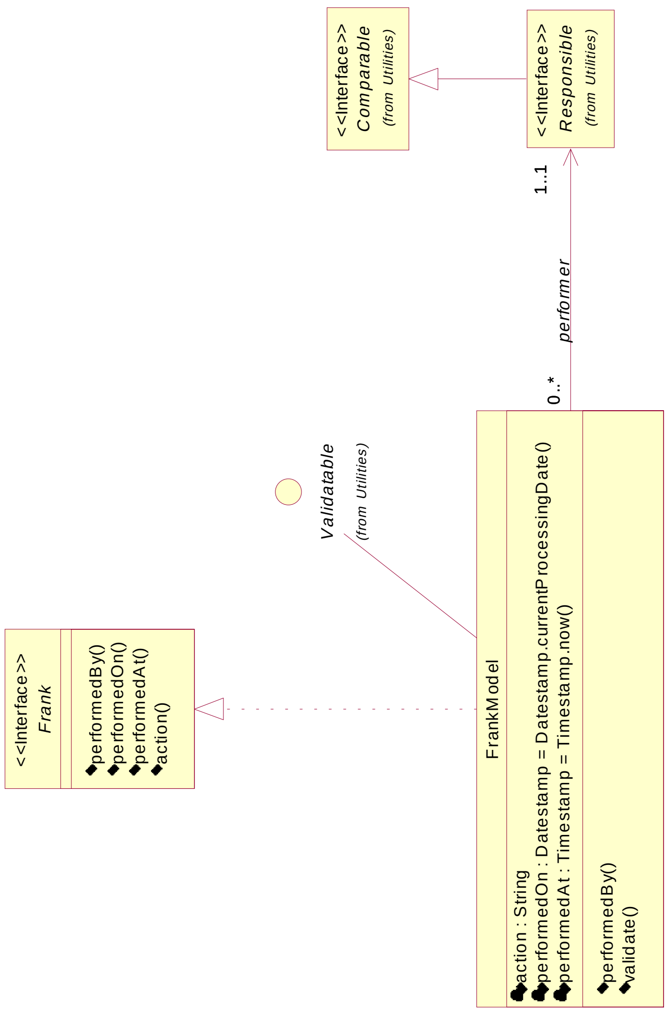

Figure 1: Class Diagram— Franking

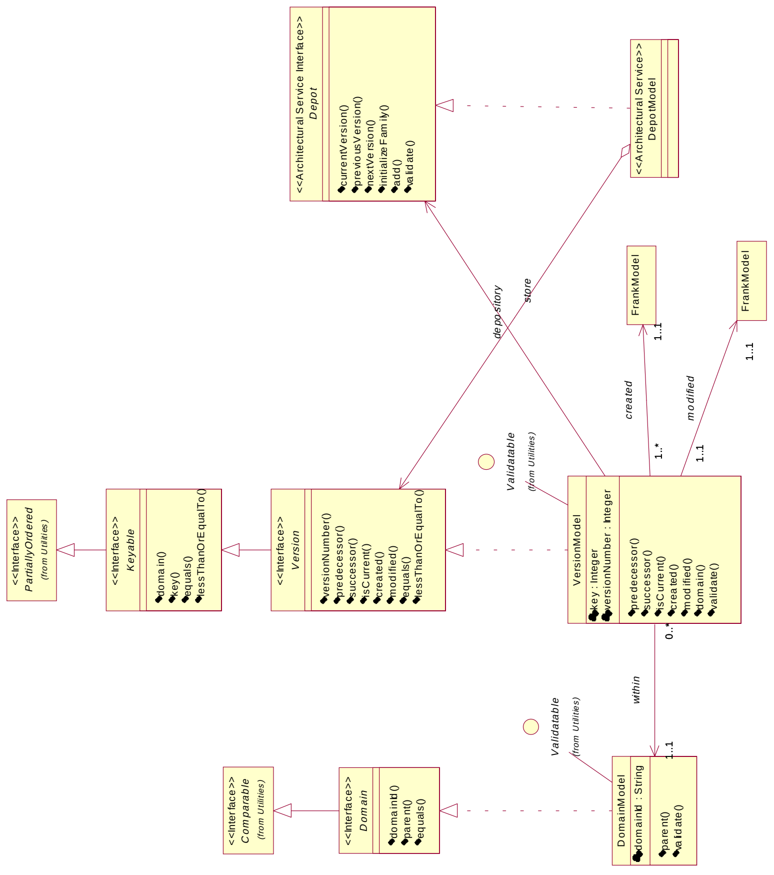

Figure 2: Class Diagram— Keying and Storage

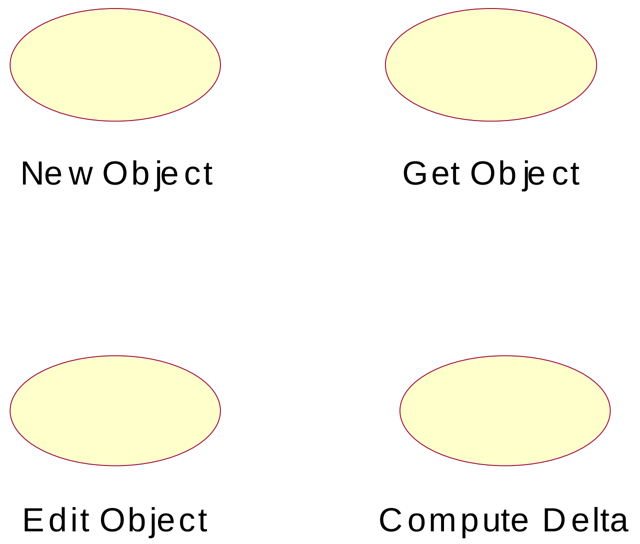

Figure 3: Class Diagram— Examples

## References

[1] *TIB/Rendezvous*.
http://www.rv.tibco.com/index.html.