Organizational Structure Facility

This document represents a consolidated initial submission and will be used as a baseline for continued consolidation toward a final revised submission.

Submitted By:

Cyborg Systems, Inc.

Gazebo Software Solutions, Inc.

Genesys Software Systems, Inc.

Supported By:

US Navy - Program Executive Office (Information Technology)

The MITRE Corporation

Initial Submission, Version 0.1
February 14, 2000
OMG Document Number: bom/00-02-xx

Preface

This specification has been prepared as a joint submission in response to the OMG Business Object Domain Task Force Organizational Structure Facility RFP.

Scope of Submission

This document addresses interfaces for an Organizational Structure Facility that is intended to capture all human resource organizational information. In doing so, it promotes the use of many established CORBA services and design patterns leveraged on subsequent finance related standards to promote an architectural consistency.

Co-Submitting and Supporting Companies

The submitting companies have each worked on aspects of this technology and have contributed their experience to develop this specification. The submitters believe that this specification represents a practical solution to an interoperable Organizational Structure Facility that fits well within the OMA, and that products can and will be developed to comply with these specifications within the time period required by OMG.

Cyborg Systems, Inc.

Cyborg Systems has extensive experience in designing and implementing object-oriented solutions for the Human Resources industry. For more than 25 years, Cyborg has focused solely on bringing quality HRM solutions to a variety of industries worldwide. Cyborg is dedicated to bringing E-Business to Human Resources departments throughout the world and is committed to building enterprise level components for the HR domain.

Gazebo Software Solutions, Inc.

Gazebo Software Solutions, Inc. is a consulting and product services company based in Denver, Colorado. Gazebo specializes in full software solutions through distributed component architectures for the enterprise taking advantage of a rich background in distributed and internet technologies. With a broad set of vertical market experiences in EAI and enterprise application technologies like ERPs, CRMs, HR systems and telecommunications call center architectures, Gazebo is well versed in how to address enterprise organizational information for business systems.

Genesys Software Systems, Inc.

Headquartered in Methuen, Massachusetts, Genesys Software Systems, Inc. develops and delivers both comprehensive outsourcing and powerful software products that take advantage of innovative human resource management system technologies. Across a variety of organization sizes, industries, types of processing, and particularly where requirements for managing HR, payroll and benefits information are large and complex, Genesys provides robust solutions.

Amy Griffis
Cyborg Systems, Inc.
4949 Pearl East Circle, Suite 105
Boulder, CO 80303
amy_griffis@cyborg.com
voice: (303) 218-6035
fax: (303) 444-5745

Robert Mickley
Gazebo Software Solutions, Inc.
7628 Halleys Dr.
Littleton, CO 80125
rmickley@gazebosoft.com
voice: (303) 933-7325
fax: (303) 948-6386

Steve Munini
Genesys Software Systems, Inc.
5 Branch St.
Mathuen, MA 01844
smunini@genesys-soft.com
Voice: (978) 685-5400 ext 3002
Fax: (978) 681-8308

The following companies formally support the content of this submission:

US Navy – Program Executive Office (Information Technology)
The MITRE Corporation

Table of Contents

1 Specification Overview

1.1 The Domain Objectives of the OSF

At the foundation of all enterprise-level computing systems one will find software constructs that organize people into one or more views of an organization. The traditional reporting hierarchy usually is the most common view or structure of an organization that comes to mind, but in actuality, many more exist. Payroll groups, geographic location, and teams working on projects are examples of other (and arguably more important) structures that an organization may need to represent accurately in software.

There is currently very little interoperability (or even open interfaces) to software that encapsulates organizational structure information. The interfaces that do exist are primarily used for limited purposes like charting and reporting. Today's solutions are extremely proprietary and are limited in scope to deliver specific application functionality. Duplication of organizational information across an enterprise is problematic in both small and large organizations alike. When organizational changes are made, updates to multiple systems are often necessary. This, of course, creates redundancy, makes integrity (both referential and consistency) controls difficult, and creates synchronization problems.

This specification defines the interfaces for a CORBA Organizational Structure Facility that can create, define, modify and search on one or more representations of an organization's structure thereby eliminating the interoperability problems that exist with today's software systems.

1.2 OMA Compatability

<Fit within OMA…>

1.3 Domain Model and Design Objectives

This section of the submission describes the core aspects of the Organizational Structure Facility by illustrating the interface hierarchy in UML. The models have been broken into small, manageable packages in an effort to succinctly communicate the design, intent and justification of each logical entity. This section does not explain individual methods or attributes but rather focuses on the overall design. Each IDL interface is described in detail in the next section.

There are many technology-related characteristics that have been indicated in the RFP as being desirable and necessary qualities. For example, the facility must allow for …

<dependency on PMF discussed here >

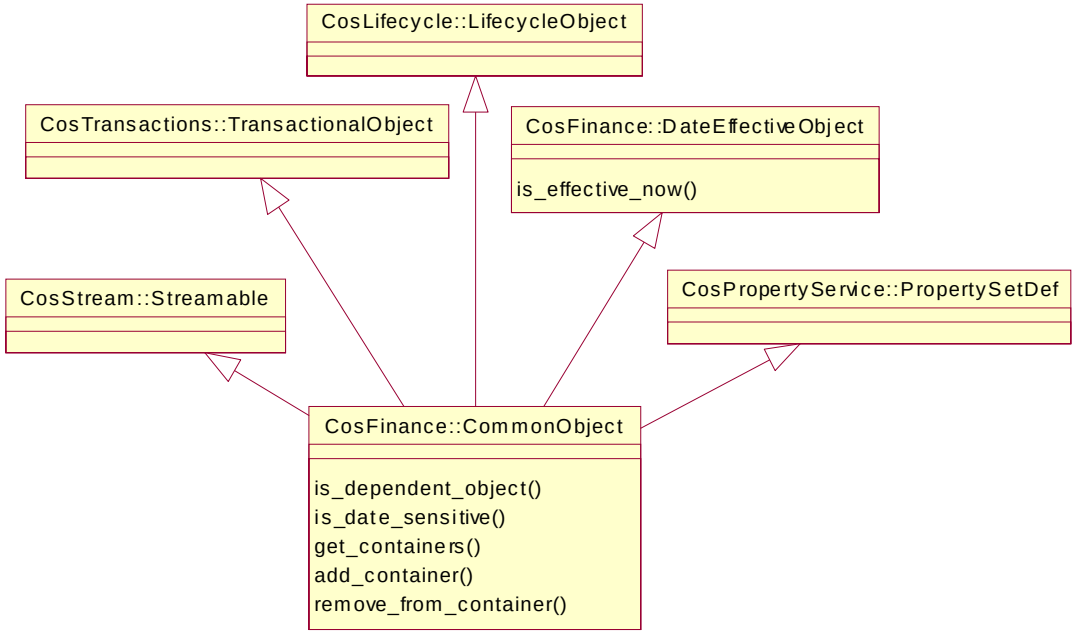
1.4 External Dependencies of the OSF

1.4.1 The Party Management Facility

The Party Management Facility (PMF) defines a pattern for how roles and relationships between parties can be used. PMF is an adopted specification that originated in the FDTF. Its initial purpose was to define parties and their relationships for the Insurance domain. However, at the completion of the standard, it became a generic facility that was not only Insurance or Finance related but allowed any party to party relationship and provided the capability for any type of party role. Because it defines basic role and relationship concepts, which are core concepts used in Organizational Structures, it is used in the Organizational Structure Facility (OSF). The interfaces used in OSF are described below. To find out more detail about these interfaces please reference the latest OMG adopted Party Management Facility specification.

1.4.1.1 The CommonObject from CosFinance

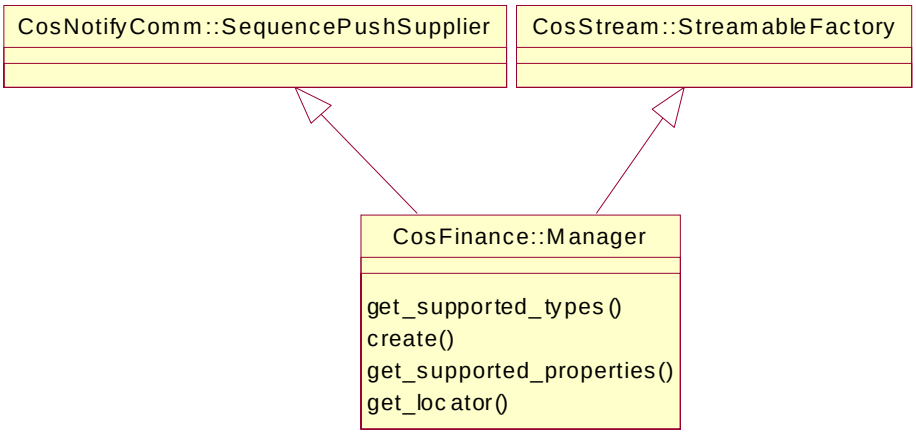
The CosFinance Common Object Base Class



The Party Management Facility provides a way to specify basic characteristics needed for all first class objects, such as having a lifecycle, being streamable, having date effectivity. These characteristics are provided by a PMF interface called CommonObject which inherits from other common CORBA services. The services CommonObject inherits from are: CosPropertyService::PropertySetDef, CosStream::Streamable and CosLifeCycle::LifeCycleObject. CommonObject is used throughout OSF when these base foundation pieces are needed. All PMF interfaces (except for RoleEnabled and other mix-in types) inherit from CommonObject.

1.4.1.2 The Manager from CosFinance

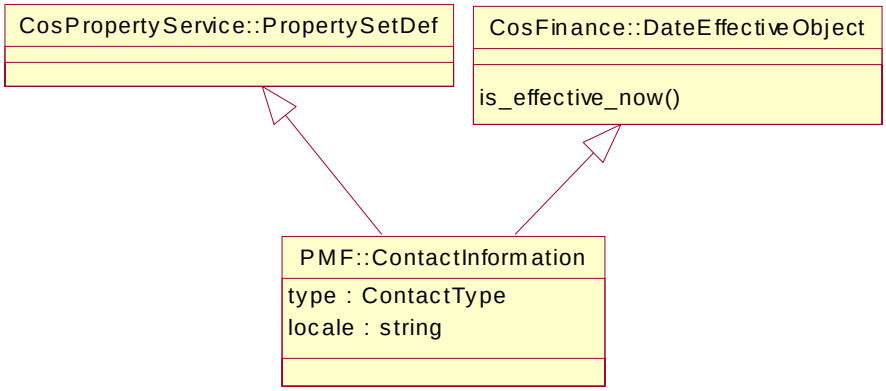
The CosFinance Manager



The PMF Manager interface provides factory interfaces along with collection management capability. It is used to manage the types in OSF, such as PositionManger that manages all of the Positions in the organization.

1.4.1.3 Contact Information from PMF

PMF Contact Information

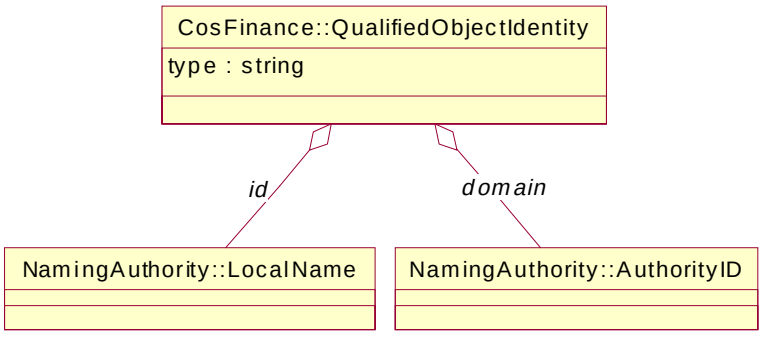


PMF defines ContactInformation. The ContactInformation interface is date-sensitive enabling the client to request all contact information as-of a specific point-in-time. ContactInformation allows any number of different types of contact information for example: multiple telephone numbers representing various ways of contacting the party. ContactInformation also inherits from the PropertySetDef allowing for dynamic attribution. It can also allow for customization, for example a business address that not only needs street information but also a suite, room, or building number. ContactInformation is vital in the Human Resource domain.

1.4.2 The Person Identification Service

1.4.2.1 The Qualified Object Identity from CosFinance

CosFinance Qualified Object Identity



1.4.3 The Resource Access Decision Facility

Most commercial Human Resource Management Systems provide application-level security facilities that permit or deny access to individual data elements or specific HRMS functionality. For example, a person's

salary is sensitive information that perhaps only HR personnel can modify, and managers can view, but other employees in the organization should not be granted either permission. Other information such as an employee's name might be available for all to see. Every organization has custom security requirements which force HRMS vendors to provide a configurable security mechanism. The Resource Access Decision Facility will be used to satisfy these requirements for the OSF.

The Resource Access Decision Facility (corbamed/99-04-04) is an application-level security mechanism that enables a common way for an application to request and receive an authorization decision. In the terminology of the Resource Access Decision Facility, the Organizational Structure Facility will be a client of one or more AccessDecision objects calling its access_allowed( ) or multiple_access_allowed( ) methods to determine if a user has authority to make a request.

RAD may throw the InternalError exception when problems occur within RAD. Implementations of OSF should catch any exceptions thrown by RAD, and translate them into an OSF exception that will be specified in the final submission.

To achieve interoperability among various OSF implementations and RAD implementations, the resource names and operations passed in the access_allowed( ) method of the AccessDecision interface must be standardized. These values will be specified in the final submission.

2 The Org. Structure Facility Interface Specification

2.1 The OrgStructure Module Specification

IDL

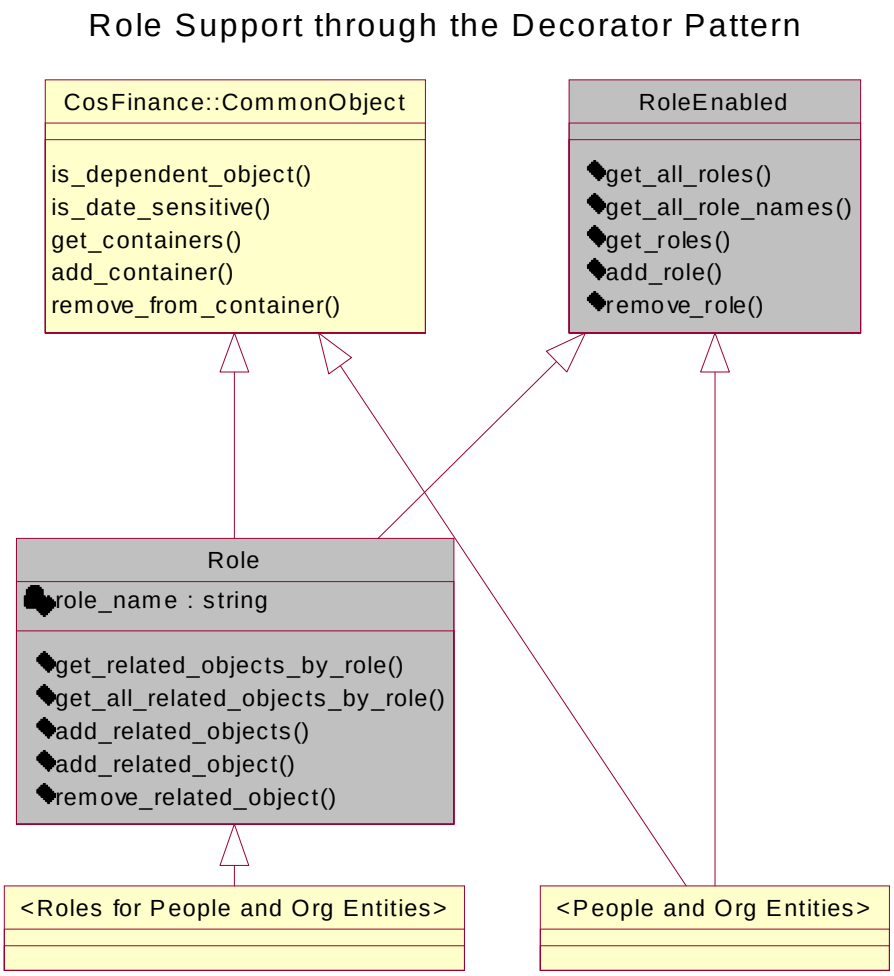
<Put all formal parts of the specification within borders like this>

<Follow it with descriptive text>

Enumerations, Data Types, and Exceptions

2.2 Roles in the Organization

< Description of roles in the organization>



2.2.1 Role Abstractions

2.2.1.1 Role

< Description of the Role Abstraction>

Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.2.1.2 RoleEnabled

< Description of the Role Abstraction>

Attributes

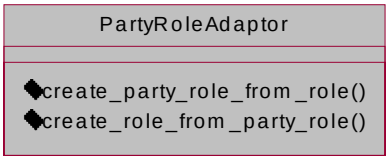
- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.2.1.3 The Adapter for PartyRole

< Description of the Role Abstraction>



Attributes

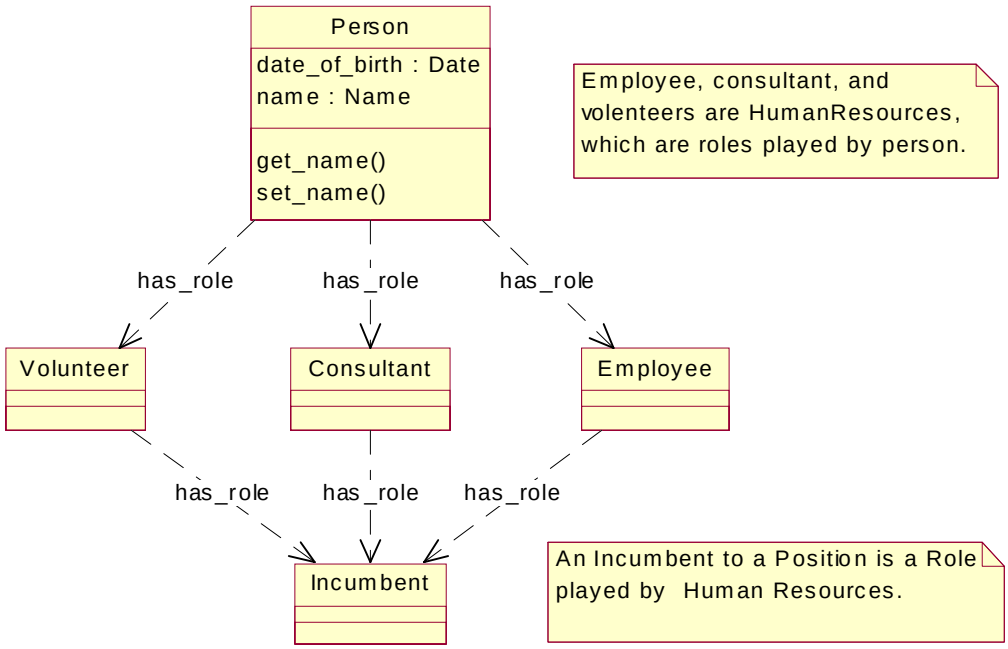
- **None**

Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.2.2 Role Relationships

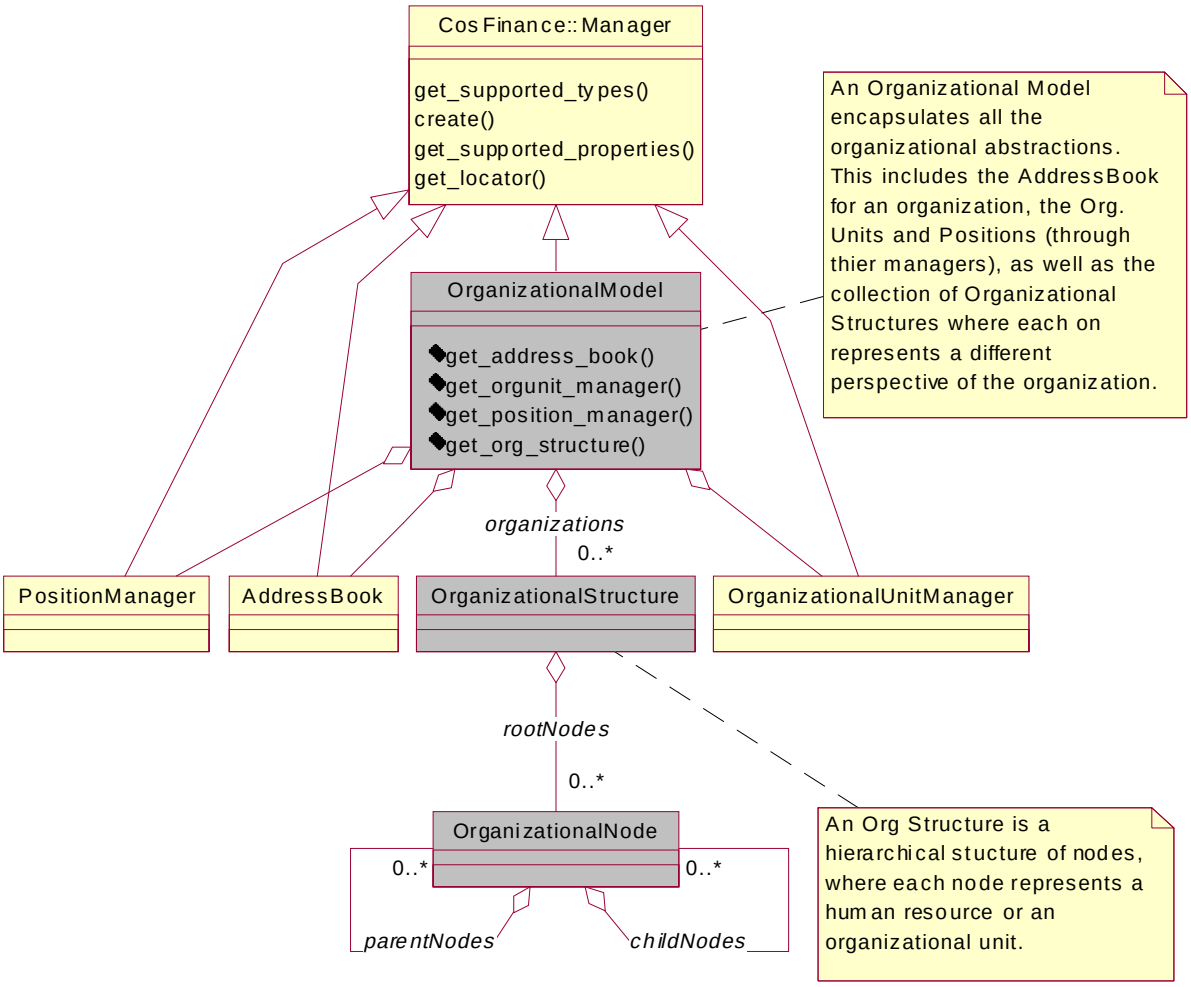
Roles relationships in the Organizational Structure



2.3 Organizational Model Interfaces

< Description of the Organizational Model Abstractions>

An Organizational Model and it's Organizational Structures



2.3.1.1 The Organizational Model

The OrganizationalModel is the container for the organization as a whole. It contains all of the structures that describe the organization (i.e. reporting structures, geographical structures etc.). It therefore acts as a manager of the OrganizationalStructure type.

IDL

```
interface OrganizationalModel : CosFinance::Manager
{
}
```

Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

Relationships

Name	Type	Properties	Purpose

2.3.1.2 The Organizational Structure

< Description of the Organizational Structure Abstraction>

IDL

<Put all formal parts of the specification within borders like this>

Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

Relationships

Name	Type	Properties	Purpose

2.3.1.3 The Organizational Node

< Description of the Organizational Node Abstraction>

IDL

<Put all formal parts of the specification within borders like this>

<Follow it with descriptive text>

Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

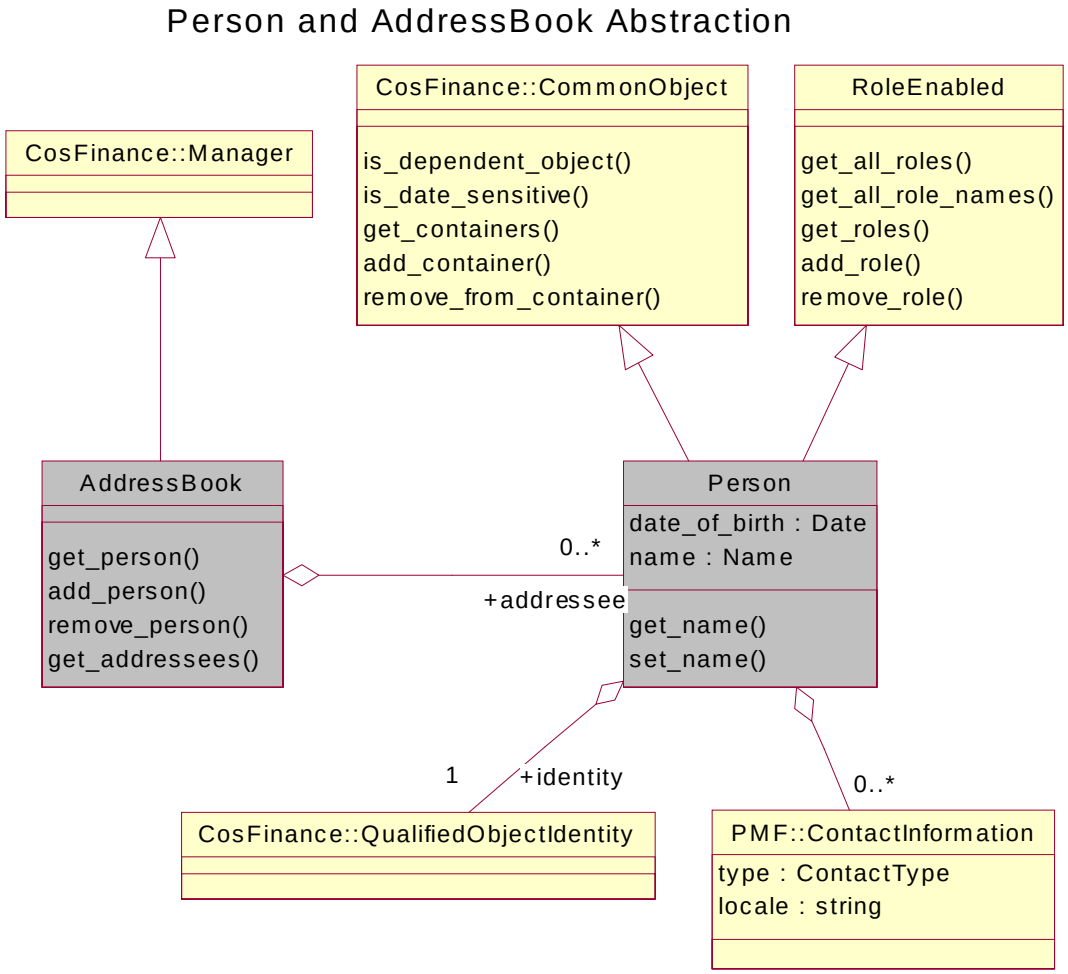
Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

Relationships

Name	Type	Properties	Purpose

2.4 People and the Address Book



2.4.1 Person

The person interface represents an actual person, living or dead whom the organization chooses to recognize, and about whom the organization chooses to collect data. A person plays or may have played one or more roles with respect to the organization, thus it inherits from RoleEnabled. Person also inherits from the CommonObject so roles may be added to its interface using CommonObject's primary_object relationship. Person also has a relationship to ContactInformation for the purpose of maintaining contact information specific to the person.

2.4.1.1 IDL

```
Interface Person: CommonObject, RoleEnabled { };
```

2.4.1.2 Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

2.4.1.3 Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.4.1.4 Relationships

Name	Type	Properties	Purpose

2.4.1.5

2.4.2 AddressBook

The AddressBook interface is a Manager from CosFinance that maintains a collection of Persons. The AddressBook is intended to be the location in the model where software can locate individual people, regardless of the roles that they play with respect to the organization.

2.4.2.1 IDL

```
Interface AddressBook : Manager { };
```

2.4.2.2 Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

2.4.2.3 Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

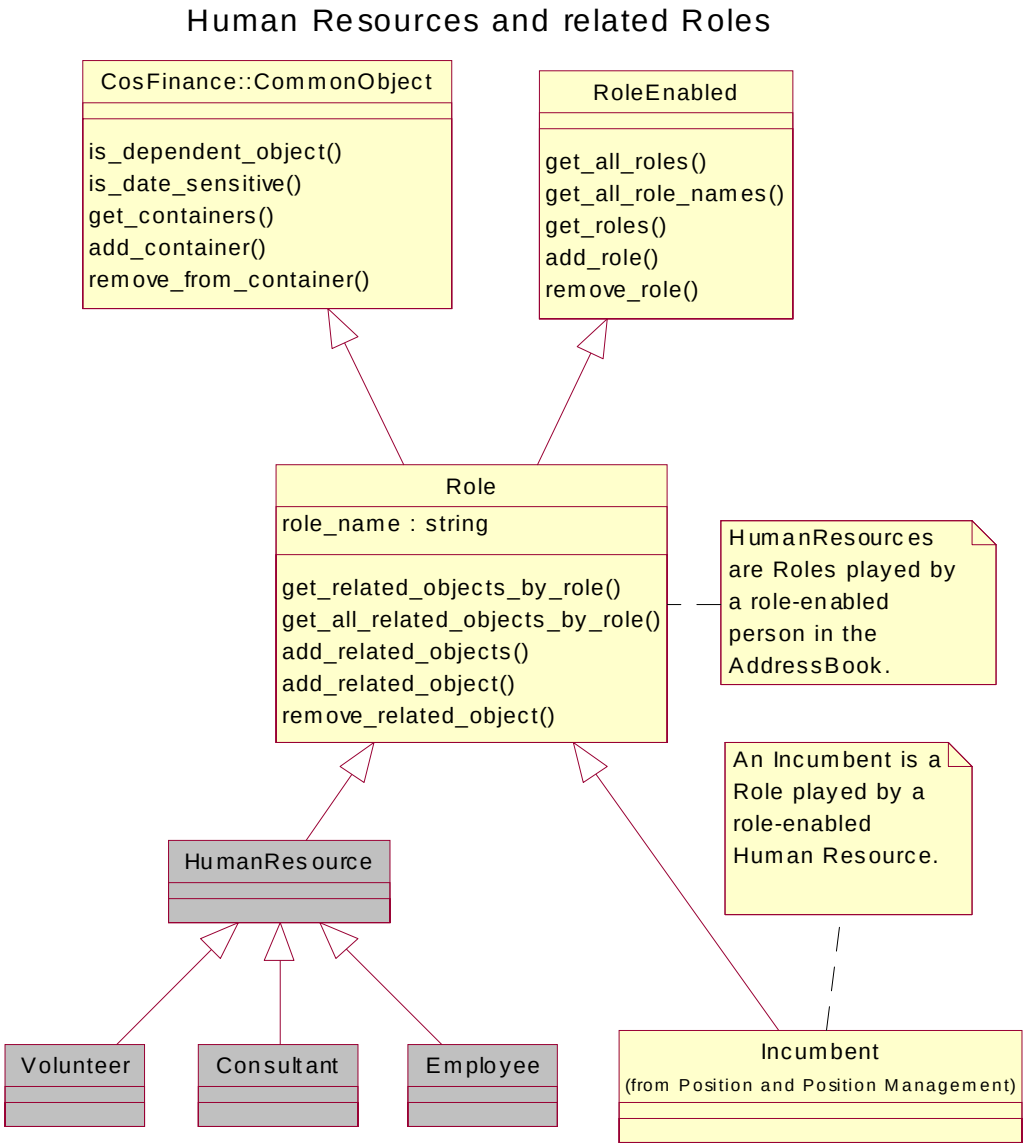
2.4.2.4 Relationships

Name	Type	Properties	Purpose

2.5 Human Resources

Our intent in this section is to describe the key interfaces that relate Person roles to the Organizational Structure Facility (OSF).

HumanResource subtypes are roles played by a Person and therefore subclass Role. Roles like Employee or Consultant conceal a Person concept which typically contains contact-type information such as address and telephone number. Access to some of this information will be available through the role interface, making it unnecessary to expose the Person interface where such exposure would be undesirable.



2.5.1 HumanResource

The HumanResource interface contains generic behavior for Role with Person as it's primary interface. HumanResource is a person role and not a role of an organization. It is a base interface for roles people play when they act as a resource in the organization. It contains operations that would allow a user of this interface to get standard person information without having to get the primary_object (Person) from the HumanResource.

2.5.1.1 IDL

```
Interface HumanResource
{
        attribute QualifiedObjectId id;
        attribute Date start_date;
        attribute Date end_date;

        void set_name(in Name name);
        Name get_name();

        PMF::ContactInformation get_contact_information();
        void set_contact_information(in PMF::ContactInformation information);
};
```

2.5.1.2 Attributes

- **CosFinance::QualifiedObjectID id** – Acts as a unique identifier for the Human Resource. In the case of an employee, this may be the employee id thru a naming authority (i.e. SSN).
- **Date start_date** – The date that this HumanResource has started with the organization (i.e. hire date)
- **Date end_date** - The date that this HumanResource ends their relationship with the organization (i.e termination date).

2.5.1.3 Operations

- **set_name()** - Change the proper name of a HumanResource type. This operation would ultimately change the state of the primary object.
- **get_name()** - Retrieve the proper name of a HumanResource type.
- **get_contact_information()** - Retrieve telephone number, address, etc. for this HumanResource.
- **get_contact_information()** - Change telephone number, address, etc. for this HumanResource.

2.5.1.4 Relationships

Name	Type	Properties	Purpose

2.5.1.5

2.5.2 Volunteer

Some firms depend heavily on Volunteers (unpaid workers) and should be capable of representing such workers within the firm's organization structure.

2.5.2.1 IDL

```
Interface Volunteer
```

```
{
        wstring get_organization_name();
};
```

2.5.2.2 Attributes

- **None**

2.5.2.3 Operations

- **wstring get_organization_name()** - Retrieve the name of the charitable institution with which this Volunteer is associated.

2.5.2.4 Relationships

Name	Type	Properties	Purpose

2.5.2.5

2.5.3 Consultant

Consultants are often not considered part of a firm's formal organizational structure, but the submission will allow the treatment of Consultants as part of the structure.

2.5.3.1 IDL

```
Interface Consultant
{
        wstring get_employer_name();
};
```

2.5.3.2 Attributes

- **None**

2.5.3.3 Operations

- **get_employer_name()** - Retrieve the name of the contracting firm with which this Consultant is associated.

2.5.3.4 Relationships

Name	Type	Properties	Purpose

2.5.3.5

2.5.4 Employee

Employee represents the most common HumanResource type likely to be used within the Organizational Structure Facility. In the binary Employment relationship, it is the role opposing employer, company, etc.

2.5.4.1 IDL

```
Interface Employee
{
        attribute QualifiedObjectId employee_id;

        Date get_date_of_birth();
}
```

2.5.4.2 Attributes

- **CosFinance::QualifiedObjectID employee_id** – A way for an organization to uniquely identify it's employees. This may be through an identifier on the person, such as SSN, or it may be an id uniquely defined by the organization.

.

2.5.4.3 Operations

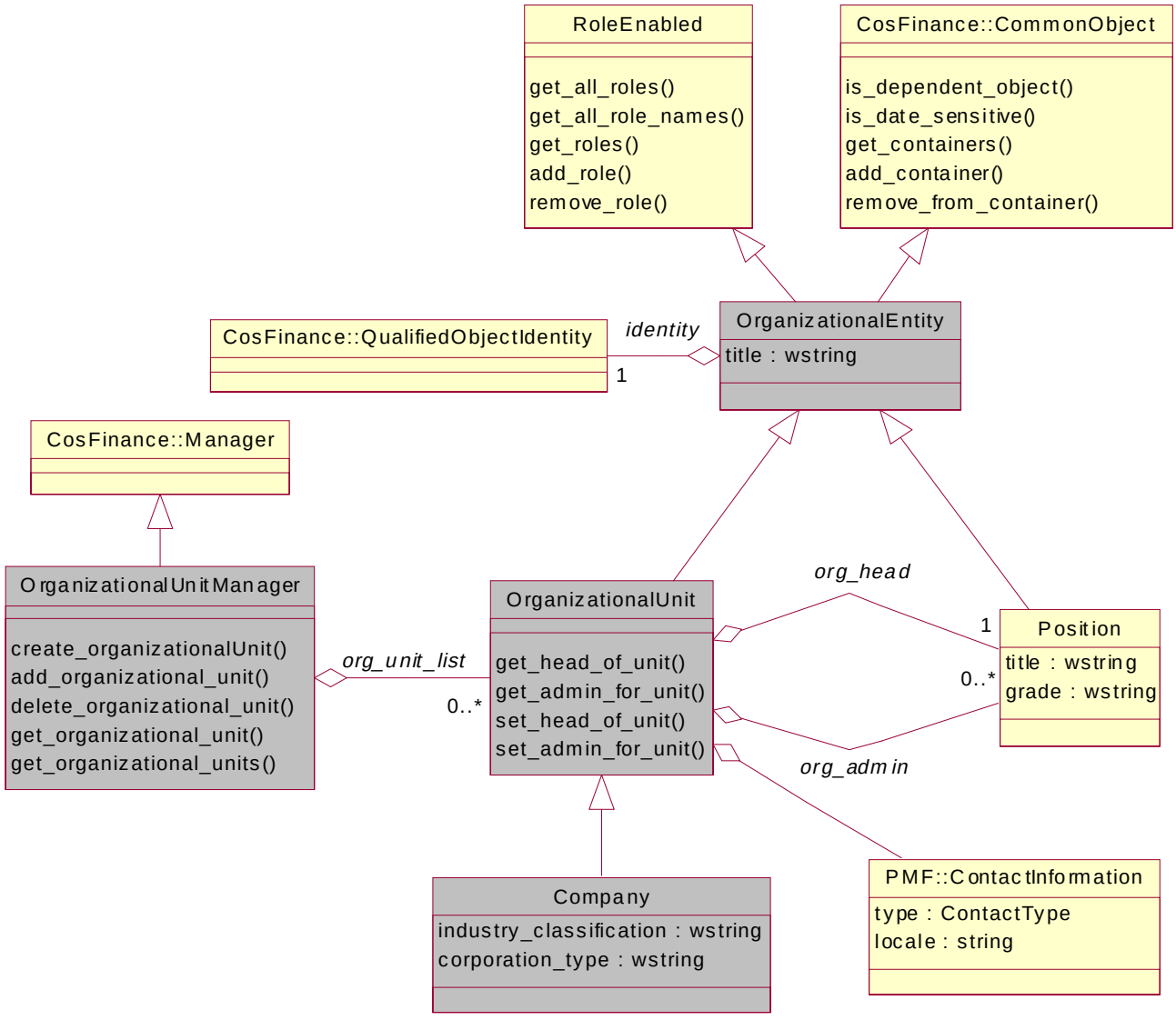
- **Get_date_of_birth()** - Retrieve the date on which this employee was born.

2.5.4.4 Relationships

Name	Type	Properties	Purpose

2.6 Organizations and Organizational Management

Organizational Unit and Organizational Unit Manager



2.6.1 OrganizationalEntity

2.6.1.1 IDL

<Put all formal parts of the specification within borders like this>

2.6.1.2 Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

2.6.1.3 Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.6.1.4 Relationships

Name	Type	Properties	Purpose

2.6.1.5

2.6.2 OrganizationalUnit

2.6.2.1 IDL

<Put all formal parts of the specification within borders like this>

2.6.2.2 Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

2.6.2.3 Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.6.2.4 Relationships

Name	Type	Properties	Purpose

2.6.2.5

2.6.3 Company

2.6.3.1 IDL

<Put all formal parts of the specification within borders like this>

2.6.3.2 Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

2.6.3.3 Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.6.3.4 Relationships

Name	Type	Properties	Purpose

2.6.3.5

2.6.4 OrganizationalUnitManager

2.6.4.1 IDL

<Put all formal parts of the specification within borders like this>

2.6.4.2 Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

2.6.4.3 Operations

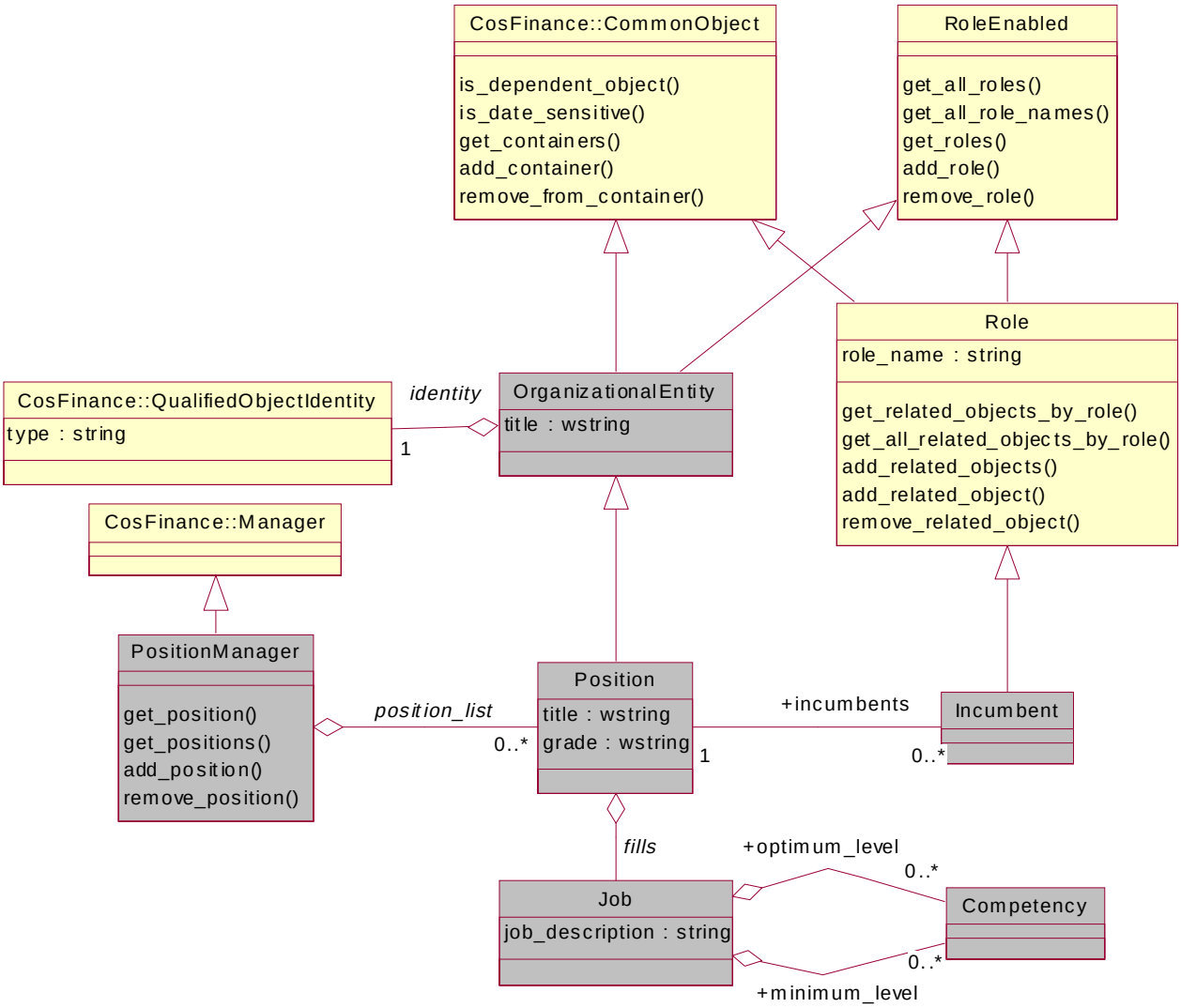
- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.6.4.4 Relationships

Name	Type	Properties	Purpose

2.7 Position, Position Management, and Incumbent

Position, Position Management, and Incumbents to Positions



2.7.1 Position

The Position interface specifies a defined position within the organization. It is a specific occupation in an organization. A position can have zero to many incumbents.

2.7.1.1 IDL

```
interface Position:OrganizationalEntity
{
        attribute string title;
        attribute string grade;
        attribute fixed<> FTE:
```

```
        Incumbents get_incumbents();
        Incumbent get_incumbent(in QualifiedObjectId incumbent)
                raises(INCUMBENT_NOT_FOUND);
        integer get_number_of_incumbents();
        void add_incumbent(in Incumbent new_incumbent);
        void remove_incumbent(in QualifiedObjectId incumbent);
}
```

2.7.1.2

2.7.1.3 Attributes

- **string title** – The position title, for example: Director ofSales, East Coast.
- **Fixed<> FTE** – Indicates the amount allocated for a full time employee. For example, if this position is supposed to have one full time employee assigned to it, FTE = 1.
- **string grade** – The grade for the position.
-

2.7.1.4 Operations

- **get_incumbents()** - returns all of the incumbents for the position. Incumbents is defined as a collection containing the type of Incumbent.
- **get_incumbent()** - This will return only the incumbent matching the key specified. A 'INCUMBENT_NOT_FOUND' exception will be thrown if no incumbent matches the key.
- **get_number_of_incumbents()** – This method will return the number of current incumbents in the position.
- **add_incumbent()** will add a new incumbent to the position.
- **remove_incumbent(I)** will remove the specified incumbent from the position. This essentially removes the Incumbent entirely, since without a position it no longer exists.

2.7.1.5 Relationships

Name	Type	Properties	Purpose
Incumbent			A position can have zero or many incumbents related to it

.

2.7.1.6

2.7.2 Incumbent

Incumbent is a Role that can be played by a HumanResource in its relationship to a Position. An incumbent is an incumbent for only one position. However an employee can have multiple Incumbent roles (i.e. an employee can be in multiple positions). The Incumbent role captures information pertaining to the relationship between the Employee and the Position.

2.7.2.1 IDL

```
interface Incumbent : CosFinance::PartyRole
{
        attribute string pay_rate_type; (in scope?)
        attribute Money pay_rate;   (in scope?)
```

```
        Position get_position();

        // interfaces to get human resource information?
}
```

2.7.2.2

2.7.2.3 Attributes

- **string pay_rate_type** – Attribute Description and purpose in text.
- **FBCCurrency::Money pay_rate** – Attribute Description and purpose in text.

2.7.2.4 Operations

- **Get_position()** - returns the position attached to this Incumbent.

2.7.2.5

2.7.3 PositionManager

The PositionManager interface provides factory mechanisms for creating positions. It manages all of the positions within an organization so it has collection and locator type interfaces.

2.7.3.1 IDL

```
interface PositionManager:CosFinance::Manager
{
}
```

2.7.3.2 Attributes

- **AttributeType AttributeName** – Attribute Description and purpose in text.
- **AttributeType AttributeName** – Attribute Description and purpose in text.

2.7.3.3 Operations

- **Operation Signature()** - Description of the operation.
- **Operation Signature()** - Description of the operation.

2.7.3.4 Relationships

Name	Type	Properties	Purpose

2.7.3.5

3 Detailed Conformance with RFP Requirements

< I think it would be good for us to use a combination of text and document references in this section to cover requirements. I've listed the requirements numbers below. If we just have references to sections in Part 1 that address the requirement (maybe with some supplemental text) we should be OK. It makes it like a requirements traceability matrix and helps us make sure we have lived up to all the requirements.>

3.1 Mandatory Requirements

3.1.1 RFP Requirements for Organizational Elements

- 6.5.1.1 - Define the organizational elements used to create the organizational structures. For example, business unit, company, position, employees, etc.
- 6.5.1.2 - Require organizational elements to be uniquely identifiable.
- 6.5.1.3 - Support customizable attribution for all organizational elements. In specifying this, the submitter shall also discuss the semantics of the customization support (for example, static vs dynamic customization, referential integrity, and impact on existing instances).
- 6.5.1.4 - Indicate how role is handled and support the concept of organizational elements being role enabled.

3.1.2 RFP Requirements for Organizational Structures

- 6.5.2.1 - Provide the ability to build organizational structures from organizational elements. This should include all traditional forms of organizational structure, for example, hierarchical organizations, and matrix organizations, etc.
- 6.5.2.2 - Provide the ability to locate and access an organizational element within a defined organizational structure.
- 6.5.2.3 - Provide the ability to traverse the organizational structure from an organizational element contained within it.
- 6.5.2.4 - Provide the ability to define organizational structures with full, partial, or no position management support.

3.1.3 RFP Requirements for Submission Characteristics

- 6.5.3.1 - Clearly identify pass by value or pass by reference characteristics.
- 6.5.3.2 - Clearly specify all error and exception conditions that may occur in the interface and their associated semantics.
- 6.5.3.3 - Not preclude relationships from within an organizational structure to entities outside. Examples of these relationships are: external liaisons, business relationships, consultant organizations, etc.
- 6.5.3.4 - Indicate how they handle the problem of orphans and roots created explicitly or by organizational changes.

- 6.5.3.5 - Provide for import and export. In relation to this, submitters will specifically discuss the use (or lack thereof) of streaming mechanisms; such as XML or MOF/XMI to facilitate this requirement.

- 6.5.3.6 - For clarification, give semantics for all interfaces and their usage. UML class diagrams and object sequence diagrams may be used to help demonstrate the architecture and clarify the interface use.

- 6.5.3.7 - Provide a clear description of any preconditions, postconditions, and invariants for each behavior specification for classification through proposals.

- 6.5.3.8 - Support the concept of organizational elements as adapters, representing elements external to the facility. For example, representations of elements stored in legacy systems.

- 6.5.3.9 - Shall indicate the security characteristics controlling the receipt and changes in data.

3.1.4 RFP Requirements for Submission Compatibility

- 6.5.4.1 - Address overlaps in functionality with other specifications, by using, extending or proposing modifications to the other specifications interface or by explaining intentional differences, where appropriate. These other specifications include but are not limited to: PDMResponsibility in the PDM Enablers specification; the PIDS specification; the Party Management Facility; the Negotiation Facility; and the Workflow Facility.

3.2 Optional Requirements

- 6.6.1 - Support multiple concurrent structures and the ability to locate and identify them individually.

- 6.6.2 - Support the ability to retrieve current organizational structures or those ‘as of’ a particular date.

- 6.6.3 - Clearly identify performance characteristics for the interface. The submitters will determine the specific characteristics, but they could include characteristics like synchronous and asynchronous calling options, or behavioral semantics.

- 6.6.4 - Support the ability to retrieve current organizational elements or those ‘as of’ a particular date.

- 6.6.5 - Support effective dating for all organizational structure changes.

- 6.6.6 - Support effective dating for all organizational element changes.

- 6.6.7 - Support the ability to define multiple concurrent organizational structures to represent different perspectives of the organization. These structures can be individually identified and located.

- 6.6.8 - Allow for the definition of templates for creating organizational elements.

- 6.6.9 - Support customizable attribution for the templates used for creating the organizational elements.

- 6.6.10 - Clearly identify how already existing instances will be dealt with when templates change.

- 6.6.11 - Address the issue of tentative changes. For example, ‘what if’ changes, approvals, business rule validation, etc.

- 6.6.12 - Support multiple retrieval based upon configured selection criteria, for example, “show me all the organizational entities that began with ‘sale’ on 10 January 1999.”

- 6.6.13 - Support notification capabilities when organizational structure and elements within the structure change through the specification. For example, a business unit is added.

- 6.6.14 - Support relationships from within an organizational structure to entities outside. For example, external liaisons, business relationships, consultant organizations, etc.

3.3 Discussion Requirements

< This section lists out all of the issues in the RFP that required discussion within the submission. Presumably these issues are addressed in the body of the specification. So, this section will just include overviews and references to the submissions perspective on the issue.>

3.3.1 Lifecycle -

3.3.2 Semantics -

3.3.3 Architectural Taxonomy -

3.3.4 Effective Dating -

3.3.5 Referential Integrity -

3.3.6 Integration of Legacy Systems –

3.3.7 Integration with OMG Security -

3.3.8 Levels of Compliance -

4 Compliance with existing OMG specifications

The Party Management Facility

< Here we should specify that we are only depending on a subset of PMF. For the initial submission we should try to outline what those dependencies are. For the final submission, we need the full closed set of PMF dependencies. The position is that you don't need to be fully PMF compliant to be OSF compliant.

All of the bullets below are used by the PMF. We should specify for each one, whether it is or is not within the scope of compliance to this specification. >

- Transaction Service -
- Property Service -
- Lifecycle Service -
- Externalization Service -
- Notification Service -
- Naming Service -
- Trading Service -

4.1 The Person Identification Service

< This needs some research, but it seems like the only dependence is on the fully qualified ID (including unique ID and naming authority).>

4.2 The Resource Access Decision

< This section should only mention that this would be valuable (maybe only for a level 2 compliance) and complementary specification for implementers. And, because of that, this specification will pass through all exceptions defined by that specification. >

4.3 MOF/XMI

< The streaming requirements and the latest surge in XML technology warrant a discussion on our use of these specifications. We should recognize it in the initial submission. We may be able to get away with resolving it by final submission. >

5 Levels of Conformance

<Below are the levels of compliance we've discussed so far. Another potential aspect of levels of compliance can be based on levels of abstraction. This way, we can define a compliance level that is predominantly domain, but doesn't require some of the base class interfaces for things like CommonObject and CommonContainer.>

5.1 Level 1 – Basic Organizational Structure Management

< In this section describe a basic organizational structure facility that can be used in a variety of B2B situations. This level of compliance may correspond to a distinction between mandatory and optional requirements. >

5.2 Level 2 – Flexible Domain Component for HR Systems

< An HR system (or CRM and ERPs) would need a more flexible, enterprise level set of requirements. Like above it can be characterized as a difference between the mandatory and optional requirements. It is a qualitative difference between the functional support for information that is supplementary vs core competence. >

Appendix A - Complete PMF IDL Listing

Appendix B - Known Issues

PMF Dependency

PMF Revisions

< This section will contain an overview of the PMF revisions that are important to the Org Structure RFP submitters. >

Compliance Levels – Functional Level vs Abstraction Level

Option 1 – Functional Level Distinction

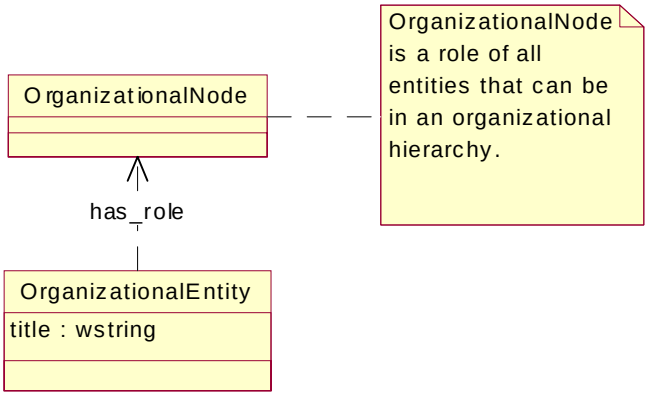
Option 2 – Abstraction Level Distinction

Scope Issues

Job / Competency abstractions for Position

OrganizationalNode – Is it a Role?

Potential Extension to Organizational Node as an Org. Entity Role



Appendix C - Collaboration Diagrams

Although a more comprehensive, domain level, interaction diagrams will be incorporated into the RM-ODP document. This appendix illustrates a few usage models to illustrate the anticipated interaction between the interfaces presented.

Appendix E- References

Appendix F - History of this document

Initial Submission Draft 1	14 February 2000

Initial Joint Submission for Cyborg Systems, Gazebo Software Solutions, and Genesys Software Systems.