

ec/2000-11-15

## Negotiation FTF Report

---

### 1 Statutory Information

The Negotiation Finalization Task Force was established during the Oslo Meeting - 15 June 2000 in order to resolve issues raised with the adopted specification dtc/99-07-03. The process of issue resolution involved the collection of issue by the OMG during which an issue numbers were assigned. The issues were then passed to the FTF for resolution following the closure of the comment period, 7 November 2000. This report details the recommendations of the FTF in respect to 10 raised issues. All members have voted affirmatively on all changes recommended in this report.

#### RTF Membership

name	company	email	votes
Stephen McConnell	OSM (chair)	mcconnell@osm.net	YES, issues 3580, 3950-3952, 3956-3960, 3980
Abdul Akram	Sprint	abdul.akram@mail.sprint.com	YES, issues 3580, 3950-3952, 3956-3960, 3980
Sanjeev Goyal	Xerox	sgoyal@crt.xerox.com	YES, issues 3580, 3950-3952, 3956-3960, 3980

#### Documents resulting from the RTF

Ref.	Title	Description
EC/00-11-14	EC Domain Specifications, Version 2.0	Containing update to Part 1 Overview, Part 2 Collaboration Criteria, Part 3 CollaborationFramework and Part 4 CommunityFramework replacing chapters 2-5, SessionFramework, CommunityFramework, CollaborationFramework and DomFramework.
EC/00-11-15	Negotiation FTF Report	(this document)

Ref.	Title	Description
EC/00-11-16	CommunityFramework 2.0 IDL	Revised CommunityFramework 2.0 IDL source.
EC/00-11-17	CollaborationFramework 2.0 IDL	Revised CollaborationFramework 2.0 IDL source.
EC/00-11-18	Session 2.0 IDL	Complete Session 2.0 IDL incorporating changes to BaseBusinessObject, AbstractResource and Link,

## 2 Summary Report

### 2.1 Summary of Issues addressed

#### *Issues related to underlying services*

Of the 10 issues raised, three (3950-52) concern practical issues relating to the specification of interfaces and services above the adopted OMG Task and Session Specification formal/00-03-05. These issues were raised as a result of implementation of interoperable community and collaboration services that reuse Task and Session abstractions.

3950 resolution of the mechanisms used to associate event suppliers with event consumers,

3951 resolution of restrictions in Session interface reuse brought about by the occurrence of several exception and struct definitions at an interface level

3952 resolution of the model used to disclose resource to resource associations such that new types of associations can be introduced in a manner consistent with the Task and Session Link model

As no Task and Session RTF is currently active, The Negotiation FTF has chosen to address the above issues through the addition of required changes to the Task and Session Specification formal/00-05-03 as part of the revised CommunityFramework specification.

#### *Issues dealing with reconstruction*

Issue 3956 formally documents the structural problems resulting from the Task and Session RTF that led to the adoption of formal/00-05-03. Prior to the release of formal/00-05-03, the Task and Session framework was defined under bom/98-07-05 and the errata 98-07-14. On one-hand changes to the Task and Session specification have substantially broken the specifications adopted under the Negotiation RFP, on the other-hand, the same changes substantially reduce the number of required interfaces to support collaborative business models such as negotiation. Issue 3956 calls for the rationalization of specifications taking into account the improvements in the Task and Session model. Furthermore, the issue recognizes the magnitude of changes that resynchronization involves and recommends the use of valuetypes as control structures as a mechanism to further simplify and improve the final specification.

In addressing this issue, the Negotiation FTF has been able to eliminate the two modules from the four modules adopted under the EC Domain Specifications. These include the SessionFramework that was largely redundant based on the subsequent Task and Session RTF recommendations, and the DomFramework module as a result of the replacement of numerous control structures with valuetype equivalents. Elimination of DomFramework is also significant in that a potential point of confusion existed between the DomFramework module and the recently adopted XML valuetype recommendations.

#### *Issues related to Membership, related policies and associations*

Of the remaining 6 issues, 5 are related to the definition of Membership, associated policies and associations. These issues have been addressed relative to a reconciled CommunityFramework module taking into account the 3956 issue noted above.

3957 re-write of the Membership description to improve clarity and understandability

- 3958 correction to a duplicate inheritance in Membership and derived interfaces
- 3959 resolution of Member type inconsistencies in synchronization with the underlying Task and Session Link specification (refer 3952)
- 3960 separation of policy attributed to a Membership as distinct to the policy attributed to a Member
- 3580 operation insufficiencies in association of business roles to an established Member

### *Issues relating to the Collaboration module*

The single remaining issue 3980 concerns the CollaborationFramework specification and proposes revision to the definition of a collaborative Encounter involving distinct separation of processor semantics from the functional role of a coordinating membership.

## 2.2 Other Modifications

Specification of the composition and state values of valuetypes defining the collaboration models dealing with bilateral negotiation, multilateral agreement and promissory contract fulfillment have been re-written using DPML - a XML schema (as opposed to tables of attribute values). This schema is included as a non-normative supplement facilitating a more precise and complete specification of state than possible under IDL. All IDL related references in the DPML DTD follow the XMI production rules.

Supplementary content introduced as a result of Architecture Board member recommendations but not raised as formal issues include the addition of factory interfaces for all concrete types. This request has been addressed through the definition of Criteria types and abstract factory interfaces. The following interface, specified under section 6.3 of the CommunityFramework defines a factory operation supporting the creation of new AbstractResource instances based on supplied Criteria valuetype.

```

exception ResourceFactoryProblem{
    ResourceFactory source;
    CommunityFramework::Problem problem;
};

abstract interface ResourceFactory
{
    readonly attribute CriteriaSequence supporting;
    Session::AbstractResource create(
        in CORBA::StringValue name,
        in CommunityFramework::Criteria criteria
) raises (
        ResourceFactoryProblem
);
};

```

## 2.3 FTF Recommendations

### *Recommendations of the FTF*

The final recommendations of the Negotiation FTF concerning content of the EC Domain Specifications document dtc/99-07-03 the are as follows:

1. Replacement of the section 1.3 "Summary of Key Features with the revised overview as presented under Part 1 section 3 of EC/00-11-14.
2. Replacement of the sections SessionFramework, CommunityFramework, CollaborationFramework and DomFramework with the chapters "Collaboration Criteria", "CollaborationFramework" and "CommunityFramework" as detailed under EC/00-11-14.

*Part 2*  
*Detailed Issues, Status and*  
*Recommendations*

---

---

No.	OMG	Title
1	3580	Cannot attribute an additional MemberKind to an exiting Member
2	3950	Multiple Consumer association to Community and Collaboration types
3	3951	Accessibility of interface scoped definitions
4	3952	Link definition is broken
5.	3956	Community and Collaboration models severely broken
6	3957	Improvement needed on Membership textual description
7	3958	Duplicate inheritance in Membership interface
8	3959	Inconstancy in the specification of the Member type
9	3960	Separation of Membership versus Role policy required.
10	3980	Separation of Collaboration from Encounter inheritance

---

## 1 Issue 3580

*Cannot attribute an additional MemberKind to an exiting Member*

1	Cannot attribute an additional MemberKind to an exiting Member	3580
	Stephen McConnell	OSM
	Issue Summary	
	<p>Under the Membership interface there are operations to add members to a membership under a specific business role. There are also operations supporting the removal of a business role of an existing member. The approach to the addition of a supplementary business roles is through the operation <code>add_kind_of_member</code>, however, this is ambiguous when the exclusivity policy is set to false (as the client may be adding a Member instance as a new member or adding a MemberKind to an existing Member). Propose elimination of the inconsistency by restricting the semantics of <code>add_kind_of_member</code> to new Member instance creation, and supplementing the Membership interface with an additional operation <code>add_kind</code> for attribution of MemberKind roles to an existing Member instance.</p>	
	Issue Resolution	
	<p>Additional operations provided to support association and retraction of business roles. Existing operations renamed to improve clarity. Specification of Membership operations have been substantially updated – refer section 3 and 4 of CommunityFramework, EC/2000-11-14.</p>	

Membership has been redefined as an abstract interface. Association of a membership model and the membership object has been clarified through the addition of an explicit abstract Simulator interface.

The `add_member` operation is renamed to `join`, `remove_member` operation renamed to `leave`, addition of `add_roles` and `remove_roles` for role modification. Specification of `join` restricted to association of a user and does not support role modification. Role modification enabled under the `add_roles` and `remove_roles` operations. Additional operations are added supporting access to the role to user associations and member listing (influenced by 3950 and 3952).

```

Member join
  in Session::User user,
  in Labels roles
) raises (
  AttemptedCeilingViolation,
  AttemptedExclusivityViolation,
  RecruitmentConflict,
  RoleAssociationConflict,
  MembershipRejected,
  UnknownRole
);

void leave(
  in CommunityFramework::Member member
) raises (
  RecruitmentConflict,
  UnknownMember
);

void add_roles(
  in CommunityFramework::Member member,
  in Labels roles
) raises (

```

```

        UnknownMember,
        RoleAssociationConflict,
        UnknownRole
    );

    void remove_roles(
        in CommunityFramework::Member member,
        in Labels roles
    ) raises (
        UnknownRole,
        UnknownMember,
        CannotRemoveRole
    );

```

## 2 Issue 3950

### *Multiple event Consumer association to Community and Collaboration types*

2	Multiple event Consumer association to Community and Collaboration types	3950
	Stephen McConnell	OSM

#### *Issue Summary*

The revised Task and Session specification of BaseBusinessObject includes inheritance of the CosNotifyComm StructuredPushConsumer and StructuredPushSupplier interfaces. The semantics of StructuredPushSupplier implies association to a single StructuredProxyPushConsumer, however, the BaseBusinessObject interface is intended to support multiple concurrent consumers from potentially different business domains without mandating nor excluding the use of Notification channels as an implementation mechanisms. To enable the documented behaviour an explicit factory operation is required through which a StructuredPushSupplier reference can be exposed for a given consumer. This behavior is required to support association of multiple consumers under the Community and Collaboration interfaces. Current inherited behavior restricts association of event channels to local implementations which inconsistent with the stated semantics.

#### *Issue Resolution*

CommunityFramework specification has been updated to include the required modification of the Task and Session specification of the definition of BaseBusinessObject to the following IDL:

Definition of an identifiable event consumer, exposed as an argument to the **add\_consumer** operation on Session::BaseBusinessObject.

```

interface IdentifiableDomainConsumer :
    Session::IdentifiableDomainObject,
    CosNotifyComm::StructuredPushConsumer
{
};

```

Convenience declaration of a TimeBase::UtcT value.

```

valuetype Timestamp TimeBase::UtcT ;

```

Revision of BaseBusinessObject that excludes event consumer and production interface inheritance in favor of an explicit factory operation named add\_consumer which enables an

implementation to return a StructuredPushSupplier associated to an identifiable StructuredPushConsumer passed in as an operation argument.

```

interface BaseBusinessObject :
    Session::IdentifiableDomainObject,
    CosLifeCycle::LifeCycleObject
    {
    CosNotifyComm::StructuredPushSupplier add_consumer(
        in IdentifiableDomainConsumer consumer);
    Timestamp creation( );
    Timestamp modification( );
    Timestamp access( );
};

```

### 3 Issue 3951

#### *Accessibility of interface scoped definitions*

3	Accessibility of interface scoped definitions	3951
	Stephen McConnell	OSM
	<i>Issue Summary</i>	
	There are several occurrences within the Negotiation and Task and Session specification of exception, enumeration and struct declarations that are defined with the scope of object interfaces. This approach complicates access to these type declarations from the Negotiation Community and Collaboration modules, leading to potential type duplication.	
	<i>Issue Resolution</i>	
	All exceptions, enumeration and typedef specifications declared within the scope of an interface within the Session, CommunityFramework and CollaborationFramework modules have been moved to module scope.	

### 4 Issue 3952

#### *Link definition is broken*

4	Link definition is broken	3952
	Stephen McConnell	OSM
	<i>Issue Summary</i>	
	The definition of a Link (a relationship declaration) under Task and Session formal/00-05-03 is in the form of a struct containing an object reference and relationship type identifier. These identifiers are declared as constants within the Session module. This change to Task and Session between bom/98-07-05 and the current formal/00-05-03 specification effectively breaks the Negotiation model of relationship extension. Restoration of module independent extension of Links is possible if the current Link declaration is replaced with an extendable valuetype definition.	
	<i>Issue Resolution</i>	
	Detailed editorial and IDL changes are included under section 9 "Changes to the adopted Task and Session Specification" of the finalized CommunityFramework specification under document number EC/2000-11-14.	

The Link struct shall be replaced with the following abstract valuetype.

```
abstract valuetype Link {
    AbstractResource resource( );
};
```

The following convenience operation has been added to enable access to a tag value indicating the role of a particular association instance in a usage relationship.

```
abstract interface Tagged {
    CORBA::StringValue tag( );
};
```

Relationships families such as usage, ownership, etc. documented under the Task and Session specification and defined through constant integer declarations shall be replaced by the following abstract valuetype definitions.

```
abstract valuetype Containment : Link{ };
abstract valuetype Privilege : Link{ };
abstract valuetype Access : Privilege { };
abstract valuetype Ownership : Privilege { };
abstract valuetype Usage : Link supports Tagged { };
abstract valuetype Consumption : Usage{ };
abstract valuetype Production : Usage{ };
abstract valuetype Execution : Link{ };
```

The following valuetype definitions replace the constant integer values used to define a particular association type.

```
valuetype Consumes : Consumption {
    public AbstractResource resource;
    public CORBA::StringValue tag;
};
valuetype ConsumedBy : Consumption {
    public Task resource;
    public CORBA::StringValue tag;
};
valuetype Produces : Production {
    public AbstractResource resource;
    public CORBA::StringValue tag;
};
valuetype ProducedBy : Production {
    public Task resource;
    public CORBA::StringValue tag;
};
valuetype Collects : Containment {
    public AbstractResource resource;
};
valuetype CollectedBy : Containment {
    public Workspace resource;
};
valuetype ComposedOf : Collects { };
valuetype IsPartOf : CollectedBy { };
valuetype Accesses : Access {
    public Workspace resource;
};
```

```

valuetype AccessedBy : Access {
    public User resource;
};
valuetype Administers : Accesses { };
valuetype AdministeredBy : AccessedBy { };

valuetype Owns : Ownership {
    public Task resource;
};
valuetype OwnedBy : Ownership {
    public User resource;
};

```

Link related operations defined under Session::AbstractResource have been updated to reflect the revision of the Link definition from struct to valuetype, enabling improvements in implementation flexibility when dealing with link navigation queries.

```

// from Session::AbstractResource

short count(
    in CORBA::TypeCode type
);

LinkIterator expand (
    in CORBA::TypeCode type,
    in long max_number,
    out Links seq
);

```

### *Implementation Impact*

From an implementation point of view the above changes represent a moderate impact. Existing systems will be required to rebuild implementations of the bind, replace, release and expand operations on AbstractResource. Previous assessment of a link type using constant values must be replaced by a test for type equivalence as distinct from explicit knowledge of constant LinkKind integer values.

### *Interface Impact*

The proposed changes break existing interfaces through the replacement of the Link struct with an abstract valuetype and a family of concrete valuetype. This revision is fully consistent with the abstract and concrete link model implied under the current specification and consistent with intent expressed section 2.5.3 Technical Note of formal/00-05-03.

## 5 Issue 3956

*Community and Collaboration models severely broken*


---

5	Community and Collaboration models severely broken	3956
	Stephen McConnell	OSM
	<i>Issue Summary</i>	
	DocumentFramework, DomFramework and CommunityFramework specifications are broken as a result of the changes introduced to the Task and Session RTF specification formal/2000-05-03	
	<ul style="list-style-type: none"> <li>▪ To a large extent the interfaces defined under the SessionFramework are now redundant.</li> <li>▪ Definition of links between resource types concerning community services must be redefined based on an extendable Link architecture (raised under issue 3952).</li> <li>▪ Issues 3461-3474 (issues posted against an interim FTF report) reflect a level of complexity of the original specification in terms of the management of control types relative to containing resource type. During assessment of issues, it has been identified that the specifications can be substantially simplified and clarified through the use of valuetype when defining control structures. This approach will enable elimination of the module thereby simplifying significantly the overall scope of the specifications and should be addressed prior to finalization.</li> </ul>	
	<i>Issue Resolution</i>	
	The extent of editorial changes introduced through the process of synchronization and valuetype rationalization has limited the practicality of included change bars within the revised specification. The FTF recommends replacement of chapters in entirety based on the EC/2000-11-14 consolidated document, thereby avoiding possible errors that could arise under a change by change editorial revision. Issues number 3461-3474 have not been included in this report as they were posted against a draft non-adopted document, however, these issues have been reviewed in detail and the conclusions reached in the resolution of the 10 recognized issues reflect recognition of the comments raised. In order to provide a complete audit trail of the changes introduced under this FTF, the following description of changes is provided.	

---

*Synchronization*

Resolution of the synchronization of the specifications has involved elimination of the SessionFramework module and all dependent references. As a general pattern, interfaces defined at the level of the CommunityFramework and CollaborationFramework modules that inherited from SessionFramework::ActiveXXX have been replaced with Session::XXX. For example, Community was defined as inheriting from ActiveWorkspace. The revised specification of Community inherits directly from Session::Workspace.

Specification of associations prior to the Task and Session RTF were expressed as interfaces. The community and collaboration frameworks used these definitions as a base from which derived relationships could be declared in a manner consistent with the Task and Session association navigation model. The revision of resource to resource associations to structs and constant association type identifiers resulted in a requirement for the specification of a new association model, or modification of the Task and Session module Link definition to accommodate out-of-module extension. The later approach has been selected. Associations dealing with the User to Membership relationship have been redefined as valuetype as noted under the revisions incorporated under issue 3959 based on the link revisions sited under issue 3952.

Other changes supporting rationalization with the new Task and Session model include the following:

- SessionFramework::LegalEntity moved to CommunityFramework

*Usage of Valuetypes on control structures*

All interface derived from the SessionFramework::Element have been replaced with corresponding valuetype definitions. During this process some degree of revision to the object has been required in order to accommodate the imposed single inheritance restrictions and leverage valuetype notions of inheritance support and abstract declarations.

The following points summarizes the equivalence between the adopted specification and the FTF recommendations concerning the CommunityFramework module:

SessionFramework::Element replaced by CommunityFramework::Control valuetype.

```
valuetype Control
{
  public CommunityFramework::Label label;
  public CommunityFramework::Note note;
};
```

SessionFramework::Template replaced by CommunityFramework::Criteria valuetype.

```
valuetype Criteria :
Control
{
  public Arguments values;
};
```

MembershipKind:1.0 replaced by MembershipModel, MembershipPolicy, Role and RolePolicy valuetypes (see also issue 3960)

```
valuetype MembershipPolicy
{
  public PrivacyPolicyValue privacy;
  public boolean exclusive;
};
```

```
valuetype MembershipModel :
Control supports Model
{
  public MembershipPolicy policy;
  public CommunityFramework::Role role;
};
```

```
valuetype RolePolicy
{
  public long quorum;
  public long ceiling;
  public QuorumPolicy policy;
  public QuorumAssessmentPolicy assessment;
};
```

```
valuetype Role :
Control
{
  public RolePolicy policy;
  public CommunityFramework::Roles roles;
  public boolean is_abstract;
};
```

Sequence typedef declarations have been replaced by boxed valuetypes.

```

valuetype Roles sequence <Role>;
valuetype Models sequence <Model>;
valuetype CriteriaSequence sequence <Criteria>;
valuetype Problems sequence <Problem>;
valuetype Note CORBA::StringValue;
valuetype Label CORBA::StringValue;
valuetype Labels sequence <Label>;

```

The following points summarizes the equivalence between the adopted specification and the FTF recommendations concerning the CollaborationFramework module:

CollaborationTemplate interface replaced by the ProcessorModel and CollaborationModel valuetypes

```

valuetype ProcessorModel :
CommunityFramework::Control
supports CommunityFramework::Model
{
public UsageDescriptors usage;
};

valuetype CollaborationModel :
ProcessorModel
{
public CommunityFramework::Role role;
public CollaborationFramework::State state;
};

```

EngagementTemplate interface replaced by EngagementModel valuetype

```

valuetype EngagementModel :
ProcessorModel
{
public CommunityFramework::Role role;
public Duration lifetime;
public boolean unilateral;
};

```

VoteTemplate interface replaced by the VoteModel valuetype

```

valuetype VoteModel :
ProcessorModel
{
public VoteCeiling ceiling;
public VotePolicy policy;
public boolean single;
public Duration lifetime;
};

```

EngagementTemplate interface replaced by EngagementModel valuetypes.

```

valuetype EngagementModel :

```

```

ProcessorModel
{
  public CommunityFramework::Role role;
  public Duration lifetime;
  public boolean unilateral;
};

```

State interface replaced by State valuetype.

```

valuetype State :
  CommunityFramework::Control
  {
    public CollaborationFramework::Triggers triggers;
    public CollaborationFramework::States states;
  };

```

Trigger interface replaced by the Trigger, Guard, Launch, Clock, Directive, and Action valuetypes. The Directive valuetypes have been introduced as part of the resolution of the separation of Encounter semantics from processor semantics (refer issue 3980) supporting declaration of the modification of usage associations.

```

abstract interface Directive {};

valuetype Duplicate
  supports Directive
  {
    public Label source;
    public Label target;
    public boolean invert;
  };

valuetype Move
  supports Directive
  {
    public Label source;
    public Label target;
    public boolean invert;
  };

valuetype Remove
  supports Directive
  {
    public Label source;
  };

valuetype Constructor
  supports Directive
  {
    public Label target;
    public CommunityFramework::Criteria criteria;
  };

```

Clock and Launch separate out the two distinct modes of operation declared by the Trigger 1.0 interface. Each type is derived from the abstract Guard valuetype and is contained by a Trigger valuetype.

```

abstract valuetype Guard {};

valuetype Clock :
    Guard
    {
        public Duration timeout;
    }
};

valuetype Launch :
    Guard
    {
        public TriggerMode mode;
        public CommunityFramework::Role role;
    }
};

abstract valuetype Action {};

valuetype Trigger :
    CommunityFramework::Control
    {
        public long priority;
        public CollaborationFramework::Guard guard;
        public CollaborationFramework::Directives directives; // precondition
        public CollaborationFramework::Action action;
    }
};

```

Transition interface replaced by abstract Transition valuetype, and supporting valuetypes Transition, SimpleTransition, Initialization, LocalTransition and TerminalTransition

```

abstract valuetype Transitional {};

valuetype Transition :
    Action
    {
        public CollaborationFramework::Transitional transitional;
        public UsageDescriptors usage;
    }
};

valuetype Initialization :
    Transitional
    {
    }
};

valuetype SimpleTransition :
    Transitional
    {
        public State target;
    }
};

valuetype LocalTransition :
    Transitional
    {
        public boolean reset;
    }
};

valuetype TerminalTransition :
    Transitional
    {
    }

```

```

    public Completion result;
};

```

CompoundTransition interface has been replaced by the valuetype CompoundTransition and the supporting valuetypes Referral, Map, Directive, Constructor, Move, Duplicate and Remove.

```

valuetype Referral :
    Action
    {
        public CollaborationFramework::Action action;
        public CollaborationFramework::Directives directives;
    };

```

```

valuetype Map
    {
        public ResultClass class;
        public ResultID code;
        public CollaborationFramework::Directives directives;
        public CollaborationFramework::Action action;
    };

```

```

valuetype Mapping sequence <Map> ;

```

```

valuetype CompoundTransition :
    Action
    {
        public CommunityFramework::Criteria criteria;
        public CollaborationFramework::Mapping mapping;
    };

```

Sequence typedef declarations have been replaced by boxed valuetypes.

```

valuetype Roles sequence <Role>;
valuetype Models sequence <Model>;
valuetype CriteriaSequence sequence <Criteria>;
valuetype Problems sequence <Problem>;
valuetype Note CORBA::StringValue;
valuetype Label CORBA::StringValue;
valuetype Labels sequence <Label>;

```

6 *Issue 3957**Improvement needed on Membership textual description*


---

6	Improvement needed on Membership textual description	3957
	Stephen McConnell	OSM
	<i>Issue Summary</i>	
	The description of Membership, Member and MembershipKind presented in 99-07-03 is overly complex and should be re-written (simplification and clarification) taking into account underlying revisions to the Task and Session specification formal/2000-05-02.	
	<i>Issue Resolution</i>	
	Membership interface and MembershipModel valuetype specifications have been rewritten taking into account revisions to operations names and simplification of operation semantics. Revised content is presented under section 3 and 4 of the CommunityFramework module specification under EC/2000-11-14.	

---

7 *Issue 3958**Duplicate inheritance in Membership interface*


---

7	Duplicate inheritance in Membership interface	3958
	Stephen McConnell	OSM
	<i>Issue Summary</i>	
	Membership is defined as a type of AbstractResource. In addition two derived types (Community and Encounter) also inherit from AbstractResource. Elimination of this conflict can be achieved by declaring Membership as an abstract interface.	
	<i>Issue Resolution</i>	
	Membership interface has been redefined as an abstract interface.	
	<b>abstract interface Membership :</b>	

---

8 *Issue 3959**Inconsistency in the specification of the Member type*


---

8	Inconsistency in the specification of the Member type	3959
	Stephen McConnell	OSM
	<i>Issue Summary</i>	
	The Member type represents an association of a User to a Membership, as such it should be re-defined as a type of Link. The current specification is currently broken and must be recast against an extendable Link model.	
	<i>Issue Resolution</i>	
	Redefinition of Member as a type of Link based on a revised Link valuetype. An abstract Link named Privilege is defined as the common relationship between a User and a Membership. The Member link defines the association to a Membership that is held by an instance of User. The Recognizes link held by a instance supporting Membership defines an association of a User to that Membership. This revision brings the User to Membership association in line with the general association model defined under the Task and Session framework.	

---

The definition of Member has been replaced by the following IDL:

```

valuetype Member : Session::Privilege {
    public Membership resource;
};

valuetype Recognizes : Session::Privilege {
    public Session::User resource;
    public Labels roles;
};

```

Operations supporting access to role information previously on Member are replaced by exposure of business roles under the state field named roles on the link held by an instance of Membership. Additional access operations dealing with disclosure of member roles and membership aggregation has been relocated under the Membership interface.

```

// from Membership

    boolean is_member(
        in Session::User user
    ) raises (
        PrivacyConflict
    );

    boolean has_role(
        in Session::User user,
        in Label role
    ) raises (
        PrivacyConflict
    );

    Labels get_member_roles(
        in Session::User user
    ) raises (
        PrivacyConflict
    );

    Session::UserIterator list_members(
        in long max_number,
        out Session::Users list
    ) raises (
        PrivacyConflict
    );

    Session::UserIterator list_members_using(
        in Label role,
        in long max_number,
        out Session::Users list
    ) raises (
        PrivacyConflict
    );
};

```

## 9 Issue 3960

*Separation of Membership versus Role policy required.*

9	Separation of Membership versus Role policy required.	3960
	Stephen McConnell	OSM
	<i>Issue Summary</i>	
	MembershipKind combines both the policy of a Membership and the policy of different business roles within a membership. These notions should be separated into independent control objects.	
	<i>Issue Resolution</i>	
	MembershipKind has been replaced by the MembershipModel and Role valuetypes. Policy applicable to a MembershipModel as distinct to the policy associated to a Role has been separated into the respective valuetypes MembershipPolicy and RolePolicy.	

The Control valuetype replaces the SessionFramework::Element interface and serves as a base type for Role.

```

valuetype Control
{
    public CommunityFramework::Label label;
    public CommunityFramework::Note note;
};

```

Role (together with MembershipModel) replace the MembershipKind interface. Role supports the separation of a business role parameters from Membership parameters and is contained within a MembershipCriteria to describe the structure and policy of logical business roles of associated users.

```

valuetype Role :
Control
{
    public RolePolicy policy;
    public CommunityFramework::Roles roles;
    public boolean is_abstract;
};

valuetype RolePolicy
{
    public long quorum;
    public long ceiling;
    public QuorumPolicy policy;
    public QuorumAssessmentPolicy assessment;
};

```

MembershipModel is a valuetype used within a Criteria instance to define the construction constraints and policies of a new Membership instance. MembershipModel together with Role replace the MembershipKind interface specification.

```

valuetype MembershipModel :
Control supports Model
{
    public MembershipPolicy policy;
    public CommunityFramework::Role role;
};

```

```

valuetype MembershipPolicy
{
    public PrivacyPolicyValue privacy;
    public boolean exclusive;
};

```

## 10 Issue 3980

### *Separation of Collaboration from Encounter inheritance*

10	Separation of Collaboration from Encounter inheritance.	3980
	Stephen McConnell	OSM
	<i>Issue Summary</i>	
	<p>Negotiation FTF, CollaborationFramework module. The definition of Encounter combines the notion of a Membership and rules relating to member association with rules relating to process execution (e.g. derived Collaboration interface). In the case of Collaboration the current inheritance model eliminates the possibility for independent declaration of process focuses role models as distinct from the roles attributed to different members of a membership. While the object model expressing roles from the two different perspectives are equivalent at an interface level, the instance values and lifecycles are independent. It is recommended that the notion of Encounter be restricted to the management of a set of members, sharing a common view on a collaborative process execution. Moving the collaboration process semantics out of the Encounter inheritance hierarchy can be achieved by defining a relationship between Encounter and the active process that an Encounter is coordinating. Once the notion of process is separated from the notion of membership, the respective control models can coexist. Secondly, given formal separation of Membership and process, the existing usage relationships derived from the inherited Task interface by Encounter can be used to manage the subject of collaborative interaction - as such, the subject relationship can be removed from Engagement, Voting and Collaboration.</p>	
	<i>Issue Resolution</i>	
	<p>Issue 3980 raises the potential for substantial interface simplification and improvements in clarity of the overall specification of collaborative interaction. Resolution of changes to incorporate separation have to large extent eliminated the semantic overloading of the Collaboration apply operation and have resulted in improvements in the reuse of Task and Session notions of resource usage. Impact on the CollaborationFramework specifications is summarized below.</p>	

#### *Revisions to the Encounter Interface*

Encounter has been modified to remove the direct declaration of the defining model (now accessible through the abstract Simulator interface inherited through Membership), and retraction of the subject association – supported by the addition of valuetypes within a ProcessorModel that declare usage association preconditions on its coordinating Task. The original definition of the model of an Encounter is now redundant as an Encounter (separated from processing semantics) is fully defined based on a MembershipModel valuetype exposed under the EncounterCriteria valuetype.

```

interface Encounter :
    Session::Task,
    CommunityFramework::Membership
{
};

```

```

valuetype EncounterCriteria :
  CommunityFramework::Criteria
  {
    public CommunityFramework::MembershipModel model;
  };

```

*Specification of Processor and supporting valuetypes*

Separation of processing semantics from Encounter enables separation of control policy concerning a particular community of members from the execution policy of a particular collaboration. This separation introduces a significant simplification of at the level of application level processes such as Collaboration.

A ProcessorModel declares the usage association preconditions that a particular processor imposes on the associated Encounter. ProcessorModel serves as a base valuetype to CollaborationModel, EngagementModel and VoteModel, replacing the equivalent Template interface definitions.

```

abstract valuetype UsageDescriptor { };

valuetype InputDescriptor :
  UsageDescriptor
  {
    public string tag;
    public boolean required;
    public TypeCode type;
  };

valuetype OutputDescriptor :
  UsageDescriptor
  {
    public string tag;
    public TypeCode type;
  };

valuetype ProcessorModel :
  CommunityFramework::Control
  supports CommunityFramework::Model
  {
    public UsageDescriptors usage;
  };

```

The Processor interface has been defined based on the semantics implied by the Task and Session specification. It serves as a base type for the Collaboration, Engagement and Vote processor definitions used in conjunction with collaborative processes such as negotiation and multilateral engagement. Two abstract interfaces Master and Slave have been included to support the association of one process to another as a controlling processor relative to subsidiary processors. While the adopted specification describes this behaviour, the interfaces do not explicitly separate process to sub-process relationships. Processor exposes an operation named coordinator that returns the controlling Task (e.g. an instance of Encounter). Other operations are derived directly from the implied semantics detailed in the Task and Session specification.

```

abstract interface Master {
  Slavelterator slaves (
    in long max_number,
    out Slaves slaves
  );
};

```

```

abstract interface Slave {
        readonly attribute CollaborationFramework::Master master;
};

interface Processor :
        Session::AbstractResource,
        CommunityFramework::Simulator,
        Master, Slave
{

        readonly attribute StateDescriptor state;

        Session::Task coordinator(
    ) raises (
                Session::ResourceUnavailable
    );

        CommunityFramework::Problems verify( );

        void start (
    ) raises (
                Session::CannotStart,
                Session::AlreadyRunning
    );
        void suspend (
    ) raises (
                Session::CannotSuspend,
                Session::CurrentlySuspended
    );
        void stop (
    ) raises (
                Session::CannotStop,
                Session::NotRunning
    );
};

```

Exposure of Processor state has been declared using valuetypes that supplements the state defined in Task and Session with an exploit declaration of Completion status. The Completion valuetypes simplify 1.0 control structures that qualified the result of collaboration model actions.

```

valuetype Completion
{
        public ResultClass result;
        public ResultID code;
};

valuetype StateDescriptor
{
        public ProcessorState state;
        public CollaborationFramework::Completion completion;
        public CommunityFramework::Problems problems;
};

```

*Modifications to the definition of Engagement, Vote and Collaboration*

The interfaces Engagement, Vote and Collaboration have been revised such that the principal base type is Processor instead of Encounter. This revision enables the possibility for a single Encounter to switch from one process to another without requiring reconstruction of a new Membership structure. In each case, the operations specific to the type have been separated out into an abstract interface that is inherited by a concrete interface. Valuetypes defining processor models and criteria are defined as replacements for the corresponding Template interfaces from version 1.0. As a result of these changes the declaration of the 1.0 Manifest type is redundant and has been withdrawn. The following two interface replace the version 1.0 Engagement interface definition.

```

abstract interface Engagement
{
    Proof engage(
        in CollaborationFramework::Evidence evidence
    ) raises (
        EngagementProblem
    );
};

interface EngagementProcessor :
    Engagement,
    Processor
{
};

```

The following valuetype replaces the version 1.0 EngagementTemplate interface.

```

valuetype EngagementModel :
    ProcessorModel
{
    public CommunityFramework::Role role;
    public Duration lifetime;
    public boolean unilateral;
};

```

The following two interface replace the 1.0 definition of the Vote interface.

```

abstract interface Vote
{
    readonly attribute VoteCount vcount;

    VoteReceipt vote(
        in VoteDescriptor value
    );
};

interface VoteProcessor :
    Vote,
    Processor
{
};

```

The following valuetype declaration replaces the 1.0 definition of VoteTemplate interface.

```

valuetype VoteModel :
  ProcessorModel
  {
    public VoteCeiling ceiling;
    public VotePolicy policy;
    public boolean single;
    public Duration lifetime;
  };

```

The definition of Collaboration is replaced by the following two interfaces. An important simplification of the semantic of the apply operation (detailed under EC/2000-11-14) has been achieved as a result of rationalization of the CollaborationModel exposure of usage constraints and usage modification directives.

```

abstract interface Collaboration
  {
    readonly attribute Label active_state;
    readonly attribute TimeoutSequence timeout_list;

    void apply(
      in Label identifier
    ) raises (
      InvalidTrigger,
      ApplyFailure
    );

    void apply_arguments(
      in Label identifier,
      in ApplyArguments args
    ) raises (
      InvalidTrigger,
      ApplyFailure
    );
  };

interface CollaborationProcessor :
  Collaboration,
  Processor
  {
  };

```

The following valuetype replaces the CollaborationTemplate interface.

```

valuetype CollaborationModel :
  ProcessorModel
  {
    public CommunityFramework::Role role;
    public CollaborationFramework::State state;
  };

```

Specifications of valuetypes and revised interfaces concerning the above revisions shall be taken from EC/2000-11-14. UML for all interfaces and valuetypes is presented at the end of each module specification in the consolidated specifications (EC/2000-11-14).

- CollaborationFramework UML, pages 79-82
- CommunityFramework UML, page 115.