Technical specifications of collaboration support tools as web services

Manolis Tzagarakis, Nikos Karousos, George Gotsis, Nikos Karacapilidis, Dora Nousia, Chui Man Yu, Sandy El Helou, Amagoia Madina

Palette
Pedagogically sustained Adaptive Learning through the Exploitation of Tacit and Explicit Knowledge

Information Society Technologies

Project no. FP6-028038

PALETTE

Pedagogically sustained Adaptive LEarning Through the exploitation of Tacit and Explicit knowledge

Integrated Project

Technology-enhanced learning

D.MED.09 –Technical specifications of collaboration support tools as web services

Due date of deliverable: October 31, 2007
Actual submission date: December 11, 2007

Start date of project: February 01, 2006 Duration: 36 months

Organisation name of lead contractor for this deliverable: CTI, EPFL

Project co-funded by the European Commission within the Sixth Framework Programme		
Dissemination Level		
P	Public	**PU**

Keyword List: Collaboration Services, Web Services, Interoperability, Integration, e-Logbook, CoPe_it!
Responsible partner: CTI, EPFL

MODIFICATION CONTROL			
Version	***Date***	***Status***	***Modifications made by***
v.09	Nov 2, 2007	draft	First complete draft (prepared by CTI & EPFL)
v.12	Nov 9, 2007	draft	Version sent to evaluators (modifications by CTI & EPFL)
v.15	Nov 29, 2007	draft	Version sent to SC members (modifications by CTI & EPFL)
v.16	Dec 10, 2007	Final	Version sent to SCO and AFC (modifications by CTI & EPFL)

Deliverable manager

- Manolis Tzagarakis, CTI
- Chiu Man Yu, EPFL

List of Contributors

- Manolis Tzagarakis, CTI
- Nikos Karousos, CTI
- George Gotsis CTI
- Nikos Karacapilidis, CTI
- Dora Nousia, CTI
- Chiu Man Yu, EPFL
- Sandy El Helou, EPFL
- Amagoia Madina Berastegi, EPFL

List of Evaluators

- Adil El Ghali, INRIA
- Yannick Naudet, CRP-HT

Summary

This document presents the technical specifications of the services that will be published by the two tools being developed in the context of WP4, namely CoPe_it! and e-Logbook. The intended audience of this document are designers and developers of the PALETTE project. The document informs them on which collaboration services will be available and how they can use them into their tools. More specifically, it describes in detail all published functions along two dimensions: the functional (i.e. what they do) and the technical (i.e. what parameters they require and what data they return) one. For each published service, the CoP needs are also discussed. With respect to technical aspects, the W3C Web Services standard [W3C-WS 2004] has been adopted as the proposed technology to publish services. For the foreseen services, a complete specification in WSDL [W3C-WSDL 2007] or WADL is given in the appendices of this document. All presented services are in

accordance with the architectural decisions taken in the context of the PALETTE Services Platform (PSP) [D.IMP.04].

Table of contents

1. Introduction

In the framework of PALETTE, collaboration services allow members of Communities of Practice (CoPs) to participate in diverse collaboration activities. Collaboration services in PALETTE, namely CoPe_it! and e-Logbook, have been designed to be used by end users. Therefore, they provide elaborated user interfaces that enable access to the various services they offer in their infrastructure. Nevertheless, the services offered by these tools need to be able to be requested not only by humans, but by other programs as well. In the context of PALETTE, this is required due to the fact that interoperability and integration of tools is considered as a key factor towards increasing CoPs abilities to address challenging problems that surface during their activities.

The interoperability and integration efforts within PALETTE have been conceived around the notion of "service" that an individual tool has to formalize and publish (i.e. make available to other tools). To overcome the high degree of heterogeneity with respect to the available services, PALETTE has introduced a conceptual framework to uniformly address and discuss existing services [D.IMP.04]. In accordance with this framework, we have adopted **both the W3C Web Services standard (SOAP) and REST** for the formal description of the services presented in this document.

In this document, we describe at the technical level the services that CoPe_it! and e-Logbook publish and make available for invocation by other tools in the PALETTE project. Three sources of requirements have influenced the publishing as well as the technical specifications of the foreseen services: (i) the needs of CoPs, as described in [D.PAR.03], [D.IMP.04] and [PALETTE Task force 2007], which were used in deciding **what services to publish,** (ii) the PALETTE Services Platform (PSP) that was used in deciding **how the selected services will be published,** and (iii) the interoperability issues elaborated in [D.IMP.04] that dictated **how the interfaces of the services will be developed**. The set of services provided by PALETTE's collaboration tools and presented in this document is by no mean complete; future releases of the tools may make additional services available. The introduction of new services may be decided upon taking into account either new CoPs needs or

new interoperability-related requirements (to be specified by the WP5 task force when dealing with generic scenarios).

This document first describes the PALETTE Service Platform Architecture, which provides the framework of all PALETTE services. Then, the published services of CoPe_it! and e-Logbook are presented at a technical level. The presentation focuses on functional as well as technical aspects of the proposed services. In addition, every presented service is related to particular CoP needs (as described in [D.PAR.03], [D.IMP.04] and [PALETTE Task force 2007]) by discussing the way in which they cover them. For the currently foreseen services, descriptions in WSDL or WADL[1] are given in the appendices of this document.

2. PALETTE Web Services

2.1. PALETTE Web Services Definition

PALETTE services are defined from two different perspectives (with respect to the definitions given in [D.IMP.03]), namely, the learning and technological ones. In this document, we consider only the second one. PALETTE technological services are materialized by one or more (composite) functions of software tools provided by PALETTE partners. Moreover, we report on PALETTE **Web** Services, which provide a set of functionalities that have been implemented in the context of the PALETTE Web applications (CoPe_it! and e-Logbook). These Web Services are based both on the Simple Object Access Protocol (SOAP) and the Representational State Transfer – (REST)[2].

The proposed PALETTE Web Services are *atomic* (i.e. they export some particular functionality of the existing PALETTE Web applications). However, *composite* Web Services, addressing various combinations of needs, can be easily created from the composition of the proposed ones.

---

1 The ability of services to interoperate is not affected by the choice to describe them in WSDL or WADL.

[2] Definitions of both SOAP and REST protocols, as well as of Service Oriented Architecture, can be found at [SOAP] [REST] [SOA] respectively.

2.2. Summary of decisions made about the PALETTE Service Platform Architecture.

All the proposed PALETTE Web Services will be provided according to guidelines that are presented in [D.IMP.04]. A brief description of the agreed PALETTE Service Platform Architecture (PSPA), together with some technical specifications, are given below.

2.2.1. Overall architecture

According to PSPA (Figure 1), a set of PALETTE services (including PALETTE Web Services) will be available to people through the PALETTE Service Delivery subsystem. This will contain both a web PORTAL, aiming at visually integrating PALETTE Services, and a web-based User Interface of the Service Registry. The latter will allow: (i) users to search for PALETTE Services, and (ii) Service Providers to make PALETTE Services public by describing and then registering them in the PALETTE Service Registry Framework (PSRF).

The PSRF is responsible for storing, managing and publishing of the PALETTE Service Descriptions. Service Descriptions are in the form of XML and are given by the associated service providers. An access and identity mechanism is also operating as part of the PSRF for enabling authentication and authorized access to the Service Registry. Finally, a Service Orchestration module assists PALETTE Service developers to create composite Web Services by reusing and combining the existing (published) PALETTE Web Services.

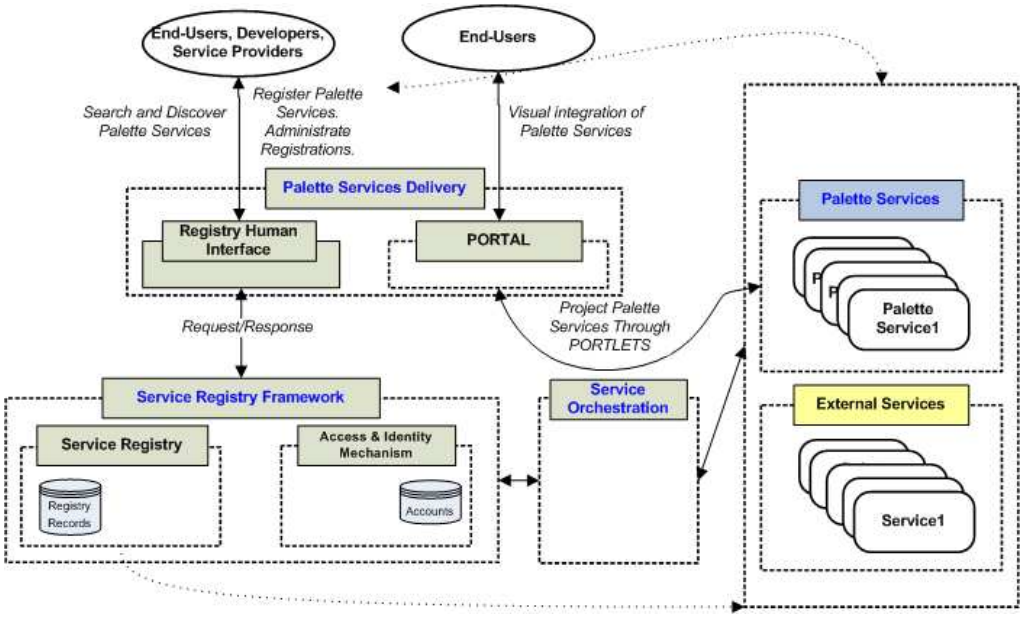

Figure 1. The overall PALETTE Service Platform Architecture.

2.2.2. Specifications concerning PALETTE Web Services

All the PALETTE Web Services will be available through the described PALETTE Service Platform. For both authentication and authorization purposes, the PALETTE Web Services should be able to handle the credentials of the requestors, in order to check the appropriate privileges for each function invoked. Although a specific security handling process is not yet decided, the Web Services developers should consider that the passing of the credential parameters of the caller (requestor) may be achieved either by including them in the header of each request message or as a part of the arguments of each function. To ensure safe transactions, credentials may be able to be encrypted when transferred over the network. Some existing standards dealing with the particular issue are presented in [http://www.w3.org/TR/SOAP-dsig]. Additional specifications of PALETTE Web Services can be found at D.IMP.04.

2.3. PALETTE Web Services registration in the PALETTE Service Platform (PSP)

For each created PALETTE Web Service, developers must follow a particular procedure in order to make it public (in the context of PALETTE):

- Create the appropriate *Web Service Technical Description File*. In case of SOAP-based Web Services, a Web Service Description File (written in WSDL) has to be produced and published in an accessible URL. If a Web Service is REST-based, the Description file has to be in the form of Web Application Description file (written in WADL).
- Create the appropriate *PALETTE Service Description Record*. This is a human-readable XML document that contains useful information about the service's functionality, location, providers etc. The schema of the PALETTE Service Description Model is available in [D.IMP.04].
- Publish the PALETTE Web Service through a registration procedure, which is taking place in the PALETTE Service Registry through the Web UI (a sub-module of the PALETTE Service Delivery, entitled "Registry Human Interface"; see Figure 1).

Once these tasks have been carried out, Web Services will be available for being discovered through the Registry Web UI or API. Furthermore, detailed human-readable descriptions of these services will be provided to requesters, so as to assist both their searching and deployment.

3.Web Services provided by CoPe_it!

In this section, we present the functionalities of CoPe_it! that have been published as Web Services and are hence available to any tool interested in using them. In the current version of CoPe_it!, a first implementation of some of the proposed services is already available. These include: repository services as well as the user management and the import/export services. Future releases of CoPe_it! will further elaborate these services and implement the remaining ones mentioned in this document.

3.1. Repository Service

As described in [D.MED.02], CoPe_it! provides an object repository for the persistent storage and retrieval of objects (including resources uploaded on the tool's workspaces as well as users involved). Some functions of CoPe_it!'s repository will be available as independent or standalone services to external applications. This means that applications will be able to use these services independently of whether they make use of CoPe_it!'s collaboration services or not. In other words, the proposed Repository Service can be exploited separately from the tool's collaboration services. Nevertheless, its existence can increase the usefulness of the collaboration related services offered by CoPe_it!.

Objects stored in the repository of CoPe_it! consist of content and metadata in the form of attribute-value pairs. The term 'content' denotes an unlimited sequence of bytes that is used to store the multimedia files that users upload during their collaboration. The repository supports all popular content-types from text, PDF and MS Word, to images and video. Although the content repository is used every time resources are brought in the collaboration workspaces (ensuring their storage and retrieval), it is also possible to request repository-related services outside the context of CoPe_it!. In this case, invocation of storage and retrieval services can be directly requested from the tool's repository. During the storage of an object, a set of arbitrary attribute value pairs can be provided that act as metadata for the object.

An integral part of the repository is the referencing and addressing mechanism for objects. The referencing mechanism allows transparent access to any object by simply specifying either the globally unique identification of the object or the object's name. The difference between object identification and object name is that while the former is automatically generated by CoPe_it!, the latter is user-chosen (it is provided by the user during the storage of objects). Object names must be unique within CoPe_it! and are the only mandatory attribute when requesting storage of objects in the repository. Retrieval of objects, given a reference, is constrained by the tool's RBAC[3] model. This means that when using a reference to retrieve (view) a resource,

[3] Role Based Access Control

access will be granted only in case that the supplied credentials have the appropriate privileges. Access to resources marked as public will be accessible to anyone, even to users and applications that have not been authenticated in CoPe_it! (i.e. are considered as 'guests').

Relationship with CoPs needs

During their collaboration, CoPs require mechanisms that improve the exploitation of common resources. In this respect, one of the activities that have to be improved is the ability to archive resources and make them retrievable from within various PALETTE services. This implies that resources must be able to be reused (a) within a particular Service, and (b) across PALETTE Services.

To illustrate further what needs are covered, the aforementioned independent repository services can be useful mainly in two situations: (i) When CoPs members, engaged in a collaboration session, encounter resources on the WWW that they consider useful for a particular discussion, but they do not yet want to post them on the collaboration workspace. In such situation, members need to *store resources into the repository in one point in time*, but *use them in a collaboration workspace at a different point in time*. (ii) When CoPs members working with one particular tool (e.g. SweetWiki) may need to reference a resource that has been used in a collaboration workspace of CoPe_it!. Or, in situations when members need to import (instead of simply referencing) a resource stored in the repository of CoPe_it! into a different PALETTE Service for further processing.

Both examples make use of the Repository Service, i.e. storing of resources by users so that they can be reused in any collaboration workspace (the attribute value pairs, which can be supplied along with the content, allow storage of the associated metadata), and referencing of the stored resources from within any tool via the referencing mechanism.

Service Implementation

The following functions are provided by the Repository Service (the detailed Web Service Description appears in Appendix A).

Web Method

```
Store(byte[] content, AVpairs attributes)
```

Rest Operation

```
http://copeit.cti.gr/services/repository/?opName=Store&content=...AVpairs=...
```

Description: Stores a resource into the repository along with a number of attribute-value pairs. Checks if all mandatory attributes are provided. Currently, object-name is the only mandatory attributes.

Input: The content as a BLOB and a set of attribute-value pairs specifying some object attributes.

Output: The ID of the newly created resource.
Exception: An exception is thrown if caller has not passed all the required user fields or AVpairs are not in the correct form.

Web Method

```
RetrieveByName(string objectname)
```

Rest Operation

```
http://copeit.cti.gr/services/repository/?opName=RetrieveByName&objectname=…
```

Description: Access a stored resource according to its unique name.

Input: Object unique name

Output: Returns the content of the resource along with its metadata.
Exception: If object does not exist, returns an appropriate exception.

Web Method

```
RetrievebyID(int identification)
```

Rest Operation

```
http://copeit.cti.gr/services/repository/?opName=RetrieveByID&identification
=…
```

Description: Access a stored resource according to its unique name.

Input: Object unique id

Output: Returns the content of the resource along with its metadata.
Exception: If object does not exist, returns an appropriate exception.

Web Method

```
Update(int id, byte[] content, AVpairs)
```

Rest Operation

```
http://copeit.cti.gr/services/repository/?opName=Update&id=…&content=...
&AVpairs=...
```

Description: Updates a stored resource.

Input: Object unique id, the new content and new AVPairs

Output: **Exception:** Exception is thrown if no object found or AVPairs are not valid.

3.2. Search Service

CoPe_it! offers services that permit the searching of resources stored in the repository based on keywords. These services will be published and made available to external tools so that the entire object repository can be made searchable. The search service, given a keyword, will return a list of resources (and in particular their unique identification along with their title) whose attribute values match the particular keyword. The initially published version of the Search Service will take only one keyword as the search term. If more keywords are provided, these are used as a phrase e.g. supplying the terms 'teaching methods' (without the single quotes) will return all resources that contain in any of their attribute values the string 'teaching methods'. This behaviour will change in future releases of this service. Access to the resources in the result set (through their reference) will be constrained by the RBAC model of CoPe_it!, as previously described.

Relationship with CoPs needs

As in the case of the Repository Service, the Search Service covers the needs of CoPs for improving the exploitation of common resources. Search services can be considered as a generalization of the addressing mechanism, in the sense that using a keyword a set of resources will be returned (rather than a single resource).

Service Implementation

Web Method `Search(string query, searchOptions options)`
Rest Operation `http://copeit.cti.gr/services/search/?opName=Search&query=…&options=...`
Description: Searches all resources in the repository of CoPe_it! for resources whose attribute values contain *query*. **Input**: A matching string (query) and a set of parameter options. A parameter

options may specify additional configuration settings for the search. These may include:

1. the maximum number of resources to be returned by the requested search. A negative value indicates no limit on the returned number of resources[4].
2. if 'page-based' navigation through the result set is required, a parameters may specify which result page to display. To control the number of pages, parameters can override the default settings of how many results per page to display.
3. the types of documents where the search will be constraint. E.g. searching for PDF or PS documents
4. the storage location: the requestor can choose if he/she wants to search only for resources uploaded in the tool's workspaces or to extend the search to resources that have been stored through the abovementioned Repository Web Service.

Output: For each matching resource, its unique identification, along with its title is returned.
Exception: Exception is thrown if parameter options are not in a valid form.

3.3. User Management Service

The User Management Service allows third-party web applications to achieve a first level of integration with CoPe_it! in a user-oriented perspective. In particular, this service assists web applications to let their users silently register/unregister to CoPe_it! (i.e. without the use of the corresponding CoPe_it! forms). Furthermore, it supports the capability of the creation and deletion of communities. In this way, it is possible for a web application to automatically add a user to CoPe_it!, whenever the particular user registers to the application (similarly, to remove a user from CoPe_it! whenever he/she unregisters). Finally, aiming at facilitating the future establishment of a single sign-on system, the CoPe_it! login page will be altered in order to support auto-login when requests come from external applications.

[4] The number of resources returned may be different from the number of resources displayed. The term 'numbers returned' denotes when the search will stop scanning for matching resources in the repository.

Relationship with CoPs needs

The User Management Service enables the creation of integrated environments in which communication with CoPe_it! is achieved in a silent mode, without forcing the users to make repeated registration operations. Thus, realization of activities in common environments can be supported. Furthermore, its main target is to "develop memberships in CoPs". By this way, this particular web service contributes to the "support of commitment" [PALETTE Task force 2007].

Service Implementation

Web Method `RegisterUser (user u)`
Rest Operation `http://copeit.cti.gr/services/um/?opName=RegisterUser&u=...`
Description: Registers a new user in CoPe_it!. **Input**: A set of user attributes that fully describes a user (see WSDL). **Output**: A user_id. **Exception:** An exception is thrown if caller has not passed all the required user fields or if the user name or email already exists.
Web Method `UnregisterUser (int user_id)`
Rest Operation `http://copeit.cti.gr/services/um/?opName=UnregisterUser&user_id=…`
Description: Removes a user from CoPe_it! by passing the participant's unique identifier. **Input**: The participant's unique identifier **Exception:** A user can be removed only if he/she has been created by the requestor and does not belong to any Community. Otherwise an appropriate exception is thrown. If user_id does not exist another exception is thrown.
Web Method `LoadCoPs()`
Rest Operation `http://copeit.cti.gr/services/um/?opName=LoadCoPs`
Description: Loads a list of the Communities in which the requestor has at

least “view” privileges.

Input:

Output: Returns a list of Communities.
Exception:

Web Method

`CreateNewCoP(string title)`

Rest Operation

`http://copeit.cti.gr/services/um/?opName=CreateNewCoP&title=...`

Description: Creates a new Community with a particular title. The requestor automatically becomes the community’s administrator.

Input: The title of the Community

Output: Returns a positive cop_id.
Exception: An exception is thrown if caller has not passed the title or if the community already exists.

Web Method

`DeleteCoP(int cop_id)`

Rest Operation

`http://copeit.cti.gr/services/um/?opName=DeleteCoP&cop_id=...`

Description: Removes a Community from CoPe_it!.

Input: The community’s unique identifier.

Output:
Exception: A Community can be removed only if it has been created by the requestor and does not contain any members. Otherwise, it returns an exception. If `cop_id` does not exist another exception is thrown.

Web Method

`AddUserToCoP(int user_id, int cop_id)`

Rest Operation

`http://copeit.cti.gr/services/um/?opName=AddUserToCoP&user_id=…&cop_id=…`

Description: Sets a CoPe_it! user as a member of a particular Community.

Input: Both the user’s and the community’s ids.

Output:
Exception: If `cop_id` or user_id does not exist an exception is thrown. Furthermore, exception can be thrown if the user is already contained by the community.

Web Method

`RemoveUserFromCoP(int user_id, int cop_id)`

Rest Operation

`http://copeit.cti.gr/services/um/?opName=RemoveUserFromCoP&user_id=…&cop_id=…`
Description: Removes a user from a Community. **Input**: Both the user's and the community's ids. **Output**: **Exception:** If `cop_id` or user_id does not exist an exception is thrown. Furthermore, exception can be thrown if the user is already out of the community.
Web Method `LoadCopUsers(int cop_id)`
Rest Operation `http://copeit.cti.gr/services/um/?opName=LoadCopUsers&cop_id=...`
Description: Loads the members of a particular Community. **Input**: The community's ID. **Output**: A list of users. **Exception:** If `cop_id` does not exist an exception is thrown.
Web Method `LoadUserProfileByID(int user_id)`
Rest Operation `http://copeit.cti.gr/services/um/?opName=LoadUserProfileByID&user_id=…`
Description: Load the user's personal information. It can also be used to check the existence of a user_id. **Input**: The ID of a user. **Output**: Returns an object of type *user* containing all the attributes that describe the user. **Exception:** If `user_id` does not exist an exception is thrown.
Web Method `LoadUserProfilebyName(string name, string pass)`
Rest Operation `http://copeit.cti.gr/services/um/?opName=LoadUserProfileByName&name=…&pass=...`
Description: Load the user's personal information. It can also be used to check the existence of a user given his name/password. **Input**: The name and the password of a user. **Output**: Returns an object of type *user* containing all the attributes that describe the user. **Exception:** If `no user` exists with those credentials an exception is thrown.
Web Method `UpdateUserProfile(user u)`
Rest Operation

`http://copeit.cti.gr/services/um/?opName=UpdateUserProfile&u=...`
Description: Replaces a user profile with a new one. **Input**: The set of all user attributes that have to be update including user_id **Output**: **Exception:** Exceptions can be thrown if user_id does not exist or user attributes are not valid (e.g. the new email already exists)

3.4. Reasoning Service

Within CoPe_it!, a collaboration taking place in a workspace may be transformed to a formal one, where the associated projection is constrained by a strict set of rules. This is in contrast to the informal projection, where no constraints on discourse moves are imposed. A consequence of the presence of argumentation rules during formal collaboration is that the tool can further facilitate the decision making process. This means that it can execute argumentation-related algorithms that augment user tasks in many ways: it can update the discourse status by identifying the winning alternatives or arguments that have been defeated, as well as check for inconsistencies among preferences. This set of algorithms constitute the Reasoning Service of CoPe_it!. In the context of PALETTE, this functionality is offered as a separate service. Clients calling this service can provide a discussion in XML (structured according to some predefined rules) and request the invocation of reasoning mechanisms. The outcome to be provided will be an annotated discussion, following the format of CoPe_it!'s formal projection (includes status of individual items, winning alternative, etc.). The discussion must be formatted in XML and validated against a specific DTD (available at Appendix A).

As follows from the above, the Reasoning Service does not necessarily require to operate within CoPe_it!. An application that exploits this service may upload an XML formatted document and the reasoning algorithm will produce the desired annotated document. For a formal description of the reasoning mechanism of this service, see [HERMES].

Relationship with CoPs needs

During argumentative collaboration, decision makers may be confronted with situations where information is either insufficient or in abundance. Both pose serious problems for the decision making process as they increase the cognitive overhead of users. The existence of reasoning mechanisms can be a valuable catalyst in these situations, as they facilitate and augment the decision making process and enable greater "capitalization of resources" [PALETTE Task force 2007].

Service Implementation

Web Method   `AnnotateDiscussion (XmlDocument cnt, Options opt, string ALGORITHM)`
Rest Operation   `http://copeit.cti.gr/services/Reasoning/?opName=AnnotateDiscussion&cnt=... &opt=...&ALGORITHM=...`
Description: Annotates a discussion described in an XML document. Annotations include winning and defeating alternatives and positions.   **Input**: Annotation may be selected from a list of available algorithms. Input format must be XML complying to CoPe_it! guidelines (see Discussion DTD in the Appendix A). Options may include identifying winning alternative or locating arguments that violate first order logic.   **Output**: Returns an annotated XML document.   **Exception:** Exceptions can be thrown if XML document is not in a valid format.
Web Method   `ValidateDiscussion (XmlDocument cnt, Options opt)`
Rest Operation   `http://copeit.cti.gr/services/Reasoning/?opName=ValidateDiscussion &cnt=...&opt=...&ALGORITHM=...`
Description: Provided an XML document, the system validates it against the required XML schema (see Discussion DTD in the Appendix A).   **Input**: The XML Document together with some validation options. Options, for example, may include locating arguments that violate first order logic.   **Output**: True or False. In false case a detailed message is also returned.   **Exception:**
Web Method

`ListReasoningAlgorithms()`
Rest Operation `http://copeit.cti.gr/services/Reasoning/?opName=ListReasoningAlgorithms`
Description: Returns a list of algorithms available as reasoning mechanisms. **Input**: **Output**: A list of algorithm names. **Exception:**

3.5. Import/Export Service

This web service allows importing and exporting of workspaces. This functionality will provide some added-value, since it will allow users of CoPe_it! to share workspaces through traditional communication methods like e-mail. Moreover, users will be able to take a snapshot of every collaboration space they have access and import them to other tools. CoPe_it! exports workspaces in XML format which consists the de-facto language for constructing documents in both machine and human readable formats. With this service, it is possible for third party applications to take advantage of CoPe_it! workspaces.

In the context of this work, and as a first application of the tool's generic Import/Export Service, we have already implemented the importing and exporting of workspaces created in a very popular application, namely Compendium (CoPe_it! workspaces are interpreted as "maps" in Compendium). The above functionality is provided as a web service as well. For a detailed study of Compendium Importing/Exporting functionality, see http://compendium.open.ac.uk/developers/os_documentation.php. The DTD of Compendium maps is available at http://compendium.open.ac.uk/developers/docs/Compendium.dtd.

Relationship with CoPs needs

Importing and exporting workspaces to and from CoPe_it! will allow users of the tool to "capitalise all the resources produced by CoP members" and access their "activities outcomes more easily" [PALETTE Task force 2007].

Service Implementation

Web Method `Export (int spaceID, Options opt)`
Rest Operation `http://copeit.cti.gr/services/IO/?opName=Export&spaceID=...&opt=...`
Description: Exports a CoPe_it! collaboration space. **Input**: The space ID together with some export options. Options allow the caller to select exporting through various methods, e.g. XML, zip file including attachments etc. **Output**: A byte array with the exported content. **Exception:** If space does not exist or export is not possible an exception will be returned.
Web Method `Import (int spaceID, int copID, byte[] cnt, Options opt)`
Rest Operation `http://copeit.cti.gr/services/IO/?opName=Export&spaceID=...&copID=...&opt=...`
Description: Imports a CoPe_it! collaboration space. **Input**: Options specify the type of file imported (e.g. XML format, zip file, compendium export etc). *CopID* is the unique identifier of the community where the new space will be created. If *spaceID* is negative, a new space will be created. **Output**: A spaceID in which the file has been imported. **Exception:** If space or cop does not exist or import is not possible due to validation error an exception will be returned.
Web Method `ExportToCompendium(int spaceID, bool attachments)`
Rest Operation `http://copeit.cti.gr/services/IO/?opName=ExportToCompendium&spaceID=...& attachments =...`
Description: Overloaded method from method "Export" to directly support exporting to Compendium maps. Exports a CoPe_it! collaboration space to Compendium format. **Input**: The space ID together with some export options. If *attachments* is true, export will include the attached files.

Output: A byte array with the exported content. **Exception:** If space does not exist or export is not possible an exception will be returned.
Web Method `ImportFromCompendium(int spaceID, int copID, byte[] thisFile)`
Rest Operation `http://copeit.cti.gr/services/IO/?opName=ImportFromCompendium&spaceID=... &copID=…&thisFile=...`
Description: Imports a Compendium-formatted map to CoPe_it!. **Input**: If *spaceID* is negative, a new space will be created. The parameter *copID* is the unique identifier of the community where the new space will be created. thisFile contains the file that will be imported. **Output**: A spaceID in which the file has been imported. **Exception:** If space or CoP does not exist or import is not possible due to validation error an exception will be returned.

3.6. Event Query Service

Through this service, it is possible to retrieve information on events that happen in CoPe_it!. Such events may concern both the users and the resources that are involved in the tool's workspaces. For instance, through this service, one may get information about who has been logged in within a certain time interval, what are the recently added resources to a certain public workspace, what communities and public workspaces have been created, etc. This information could be used by other PALETTE Services in order to achieve cross-tool awareness.

Relationship with CoPs needs

This service contributes to the need of improving participation and its quality, by addressing various awareness issues (concerning both the users and the resources that are involved in the tool's workspaces) [PALETTE Task force 2007].

Service Implementation

Web Method
`ListEvents(string EventTypes, string Filters)`
Rest Operation
`http://copeit.cti.gr/services/events/?opName=ListEvents&EventTypes=... &Filters=...`
Description: Loads a list of the Events that took place in CoPe_it! and matches the particular filters. **Input**: EventTypes is a list of predefined types of desired Events, and Filters are query criteria (both are in a form of string including sequence of comma-separated name-value attributes). **Output**: Returns a list of Events. **Exception:** A data validation may result to an exception if input is not well formed.

3.7. Summary

The following table summarizes the web services published by CoPe_it!, along with a description of them and the CoPs' needs addressed.

Services	**Description**	**CoPs' needs addressed**
Repository Service	Provides direct resource storing and retrieving capabilities.	Improves exploitation of common resources. Resources are accessible from any other tool.
Searching Service	Allows searching in CoPe_it! Resources.	Improves exploitation of common resources. Resources are accessible from any other tool.
User Management Service	Enables selected CoPe_it! functionality for remote User and CoP Management.	Supports commitment through the development of memberships and realization of activities in common environment through enabling a partial integration.
Reasoning Service	Provides winning/defeated alternatives on the dialogue and highlights conflicts between arguments that violate	Improves participation and its quality; Improves the exploitation of the common resources (to capitalise all the

	first order logic.	resources produced by CoPs)
Import/Export Service	Importing and exporting CoPe_it! workspaces to various XML formats.	Improves the capitalisation of the resources produced by CoPs members.
Event Query Service	Provides requestors with a list of actions that took place in a particular period of time.	Improves participation and its quality

4.Web Services provided by e-Logbook

In this section, we present the functionalities of e-Logbook that have been published as REST Web Services. They are available to any tool interested in integrating services provided by e-Logbook in the framework of the PALETTE project. The services are classified according to the following two categories: Entity management services and query services. For each service, we provide a general description that also justifies its relationships to CoPs needs, as well as a more technical description of it. Section 4.1 describes the Entity Management Services. Section 4.2 describes the Query Services. The description of e-Logbook services is function-oriented. Appendix B contains the WADL description of some e-Logbook Web Services, in particular the "create activities" and "query activity" services.

Web services currently provided

e-Logbook currently provides the following REST Web services.

Entity Management Services:

- *create activities*, *create assets*, *create deliverables*, and *update status on activity*.

Query Services:

- *query activity*, *query actors*, *query asset, and query deliverable.*

Web services to be provided

A number of Web Services is under construction. We plan to provide the following REST Web services for e-Logbook in the future (the details of these services are not described in this document).

Entity Management Services:

- *update {activity/asset/deliverable}, delete {activity/asset/ deliverable}, and post asset in deliverable.*

Metadata Management Services:

- *tag {activity/asset/actor}, rate {activity/asset/actor}, retrieve tags of {activity/asset/actor}, and retrieve ratings of {activity/asset/actor}.*

Relationship with tools and services in the framework of the PALETTE project

The participatory design methodology has led to the definition of scenarios, which have been validated by CoPs members. Based on the current version of these scenarios (reported in [D.PAR.03]), [D.IMP.04] has identified some possible interactions between services. Below we list the interactions related to e-Logbook, as they reveal from the scenarios that concern ePrep and Learn-Nett.

1. *File upload:* SweetWiki, Amaya, and LimSee3 files are uploaded on e-Logbook as assets. e-Logbook supports knowledge management for CoPs through sharing of assets. CoPs members can use annotation tools (e.g., Sweetwiki) to add tags to their files, and store the files on e-Logbook. The CoPs members can then manage the files in the form of assets on e-Logbook, for example, editing, tagging and sharing.
2. *Context awareness:* CoPe_it! invokes e-Logbook's "Context Aware View" awareness function. e-Logbook notifies CoPs members for events involving actors, assets and activities related to the members.

The 'file upload' interaction clearly states the need of "create asset" service to PALETTE Services. This service is especially useful for PALETTE Services or other Web applications that generate files or data for CoPs members.

The 'context awareness' interaction states the need of providing context awareness function to PALETTE Services. To use the e-Logbook context awareness function, the PALETTE Services or other Web applications may manage (create, update and delete) assets, activities and deliverables for their

CoPs members on e-Logbook. The CoPs members can then be aware of events related to their assets and activities. In this view, the "{create/update/delete} {assets/activities/deliverables}" services (Entity Management Services) are useful for PALETTE Services and other Web applications that want e-Logbook to provide context awareness to their CoPs members.

e-Logbook provides activity management, knowledge management, and context awareness functions. Any Web applications that want to use these functions can use the Entity Management Services provided by e-Logbook.

Web applications can retrieve the asset and activity information of their CoPs members through the "query asset", "query activity", and "query actors" services provided by e-Logbook. These services also provide a way for CoPs members to share their information across multiple PALETTE Services and other Web applications.

4.1. Entity Management Services

General description

The major types of entities in e-Logbook include 'actor', 'asset', 'activity', and 'deliverable'. An actor (CoP members) can manage the latter three kinds of entities for knowledge management, activity management, and awareness functions.

The Entity Management Services enable service consumers (actors or applications) to create and update their assets, activities, and deliverables on e-Logbook. The actors can create these entities by providing the names of the entities and optional entity information. The actors can also update status of their activities through the services.

The Entity Management Services are as follows:

- *Create activities*: This service creates an activity. The activity has an activity name, and optionally a description, both provided by the service initiating the request. If the activity is successfully created, the service replies with the activity's information including name, description, ID, default role, creator, built time, update time, and public invitation style.
- *Create assets:* This service creates an asset. The asset has an asset name, and optionally a description and a default right of accessing the asset. Again, the

service initiating the request provides this information. If the asset is created successfully, the service replies with the asset's information including name, description, ID, default access right, creator, built time, update time, and number of asset attachments.

- *Create deliverables*: This service creates a deliverable for an activity. The service initiating the request provides information such as a deliverable name, and optionally a description, a submission deadline and a validation deadline. If the deliverable is created successfully, the service replies with the deliverable's information including name, description, ID, creator, built time, update time, submission deadline, validation deadline, and activity information.

- *Update status on activity*: This service updates the status of an invitation for an actor on an activity. The service consumer has to provide the activity ID, invitation ID and the new status.

Relationship with CoPs needs

CoPs have the need to share common resources among their members. The resources can be information related to actors, activities, and assets. In PALETTE framework, the PALETTE tools need to create, share and retrieve resources.

In e-Logbook, the Entity Management Services enable service consumers to create or update three kinds of entities: activity, asset, and deliverable. These services support the CoPs need of "exploitation of the common resources". Besides, CoPs need to manage their activities for their own evolution. They need functionalities to create activities and update the activities' information. In e-Logbook, the Entity Management Services enable service consumers to create and update activities. These services support the CoPs need of "realization of the activities" [PALETTE Task force 2007].

Service Implementation

Method Name and arguments	*Description*

`Service: create activity`
`Service path:`
`http://e-Logbook.epfl.ch/create/activity.xml`
`Input parameters:`
	Create an activity on e-Logbook.

string name, string description (optional) Output: XML with elements: (name, description, id, created-at, updated-at, initiator-actor-id, default-role-id, public-invitation-style)	
Service: create asset Service path: http://e-Logbook.epfl.ch/create/asset.xml Input parameters: string name, string description (optional), string default-right (optional). Output: XML with elements: (name, description, id, created-at, updated-at, initiator-actor-id, default-right, asset-attachments-count)	Create an asset on e-Logbook.
Service: create deliverable Service path: http://e-Logbook.epfl.ch/activities/[:activity_id]/create/deliverable.xml where [:activity_id] is replaced by the ID of an activity Input parameters: string name, string description (optional), datetime submission_deadline (optional), datetime validation_deadline (optional). Output: XML with elements: (name, description, id, created-at, updated-at, initiator-actor-id, submission_deadline, validation_deadline, activity id, activity name, activity description)	Create a deliverable on e-Logbook.
Service: update status on activity Service path: http://e-Logbook.epfl.ch/activities/[:activity_id]/invitations/ [:invitation_id].xml where [:activity_id] is replaced by the ID of an activity, and [:invitation_id] is replaced by the ID of an invitation Input parameters: string status. Output: XML with elements: (actor-id, created-at, updated-at, id, role-id, status, invitation-style, notify-me)	Update the status of an invitation on an activity for an actor on e-Logbook.

4.2. Query Services

General description

The Query Services enable service consumers (actors, or services that on behalf of actors) to retrieve information from e-Logbook. The information available includes list of entities related to the actors, e.g., activity list, asset list, and deliverable list; as well as list of entities related to activities, e.g., actor list for an activity, and deliverable list for an activity. The generated information is in the form of XML.

The Query Services are:

- *Query for activity list*: This service replies with the activity information for target actor. The activity information includes the activities' name, description, ID, and member actors and roles.
- *Query for activity list by status*: This service replies with the activity information for target actor. The activity information includes the activities' name, description, ID, and member actors and roles.
- *Query for actor list participating on an activity*: This service replies with the participants' information of an activity for target actor. The participant information includes the actors' ID, name, role and invitation status in the activity.
- *Query for asset list*: This service replies with the asset information for target actor. The asset information includes the assets' name, description, ID, and access right.
- *Query for deliverable list*: This service replies with the deliverable information for target actor. The deliverable information includes the deliverables' name, description, ID, submission deadline, validation deadline and the parent activity.
- *Query for deliverable list for an activity*: This service replies with the deliverable information of an activity for target actor. The deliverable information includes the deliverables' name, description, ID, submission deadline, validation deadline and the parent activity.

Relationship with CoPs needs

CoPs have the need to share common resources among their members. They need the functionalities to retrieve the information of the common resources, for example, information of activities. In e-Logbook, the Query Services enable service consumers to query for actor list and deliverable list for individual activities. These services support the CoPs need of "exploitation of the common resources". Besides, CoPs have the need for their members to develop their membership. The CoP members need the functionalities to keep track of their status and related information in order to clarify their own project. In e-Logbook, the Query Services enable service consumers to query for their related activity, asset, and deliverable information. These services support the CoPs need of "supporting commitment" [PALETTE Task force 2007].

Service Implementation

Method Name and arguments	*Description*
Service: query activity Service path: http://e-Logbook.epfl.ch/activities.xml Input: Null Output: XML with elements: (name, description, id, roles)	Query for the activities related to the current actor.
Service: query activity by status Service path: http://e-Logbook.epfl.ch/activities/[:status].xml where [:status] is replaced by either "joined", "pending" or "refused" Input: Null Output: XML with elements: (name, description, id, roles)	Query for the activities related to the current actor with a status.
Service: query actors Service path: http://e-Logbook.epfl.ch/ activities/[:activity_id]/actors.xml where [:activity_id] is replaced by the ID of an activity Input: Null Output: XML with elements: (name, id, role, status)	Query for the participants of an activity related to the current actor.

Service: query asset Service path: http://e-Logbook.epfl.ch/assets.xml Input: Null Output: XML with elements: (name, description, id, rights)	Query for the assets related to the current actor.
Service: query deliverables Service path: http://e-Logbook.epfl.ch/deliverables.xml Input: Null Output: XML with elements: (name, description, id, submission-deadline, validation-deadline, activity id, activity name, activity description)	Query for the deliverables related to the current actor.
Service: query deliverables for an activity Service path: http://e-Logbook.epfl.ch/activities/[:activity_id]/deliverables.xml where [:activity_id] is replaced by the ID of an activity Input: Null Output: XML with elements: (name, description, id, submission-deadline, validation-deadline, activity id, activity name, activity description)	Query for the deliverables related to an activity and the current actor.

4.3. Summary

This section summarizes the published services of e-Logbook in terms of their descriptions and CoPs' needs addressed.

Published service	**Description**	**CoP needs addressed**
Entity Management Services	The entity management services enable service consumers (actors) to create and update their assets, activities, and deliverables on e-Logbook.	Improve the exploitation of common resources. Resources can be created and be updated in e-Logbook; Support realization of the activities through creation and update of activities.
Query Services	The query services enable service consumers (actors) to retrieve information from e-Logbook. The	Improve the exploitation of common resources. Information of resources is accessible from

	information available includes information related to the actors, as well as information related to activities.	any other tool; Support commitment through providing information related to CoP members.

5. Concluding remarks

In this document, the technical specifications of the published collaboration services being developed in the context of WP4 have been presented. For a number of services, detailed descriptions are given in the appendices of this document. Three sources influenced the technical specifications of the collaboration services:

- the needs of CoPs involved in PALETTE, indicating what services to publish;
- the PALETTE Services Platform, indicating how selected services have to be published, and
- the interoperability issues that dictated how the interfaces of the services will be developed.

Although no modification on the functional part of services is expected in the future, the technical aspects (e.g. what input parameters to provide and what output information to give) may be a subject of amendment. This is due to insights that will be gained once these services will be deployed and evaluated in real settings.

6. References

- [SOAP] W3C Recommendation, "Simple Object Access Protocol", 2007, available from http://www.w3.org/TR/soap/
- [REST] Fielding, R.: "Representational State Transfer", PhD dissertation, University of California, Irvine, 2000, available online at: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- [SOA] Working Group of The Open Group, "Service Oriented Architecture", 2006, available from http://opengroup.org/projects/soa/doc.tpl?gdid=10632
- [W3C-WS 2004] W3C Recommendation, "Web Services Architecture", 2004, available from http://www.w3.org/TR/ws-arch/ .
- [W3C-WSDL 2007] W3C Recommendation, "Web Services Description Language (WSDL) Version 2.0 Part 0: Primer", 2007, available from http://www.w3.org/TR/wsdl20-primer/
- [D.PAR.03], PALETTE Deliverable D.PAR.03, "Description of 6 scenarios and results of 6 validated trials", PALETTE project.
- [D.IMP.03] PALETTE Deliverable D.IMP.03, "First functional specifications of services and guidelines for services orchestration", PALETTE project, 2007.
- [D.IMP.04] PALETTE Deliverable D.IMP.04, "Updated version of guidelines for development", PALETTE project, 2007.
- [PALETTE Task force 2007] PALETTE Working document of the task force for the reorganization of the teams, "Proposal for the preparation of the Task Force", PALETTE project, 2007, available from https://bscw.ercim.org/bscw/bscw.cgi/334477
- [D.MED.02] PALETTE Deliverable D.MED.02, "First operational version of the web-based tool supporting argumentative collaboration towards learning", PALETTE project, 2007.
- [HERMES] N. Karacapilidis and D. Papadias: HERMES: Supporting Argumentative Discourse in Multi-Agent Decision Making. In

Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98), Madison, WI, July 1998, AAAI/MIT Press, pp. 827-832.

Appendix A: CoPe_it ! Web Service Description Files (WSDL)

Repository Service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://copeit.cti.gr/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://copeit.cti.gr/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://copeit.cti.gr/">
      <s:element name="Export">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="spaceID"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="opt"
type="tns:ArrayOfTAttribute" />
          </s:sequence>

        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfTAttribute">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded"
name="TAttribute" nillable="true" type="tns:TAttribute" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="TAttribute">
        <s:sequence>

          <s:element minOccurs="0" maxOccurs="1" name="name"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="value"
type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:element name="ExportResponse">
        <s:complexType>
```

```
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="ExportResult"
type="s:base64Binary" />
      </s:sequence>

     </s:complexType>
    </s:element>
    <s:element name="Import">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="spaceID"
type="s:int" />
       <s:element minOccurs="1" maxOccurs="1" name="copID" type="s:int"
/>
       <s:element minOccurs="0" maxOccurs="1" name="cnt"
type="s:base64Binary" />
       <s:element minOccurs="0" maxOccurs="1" name="opt"
type="tns:ArrayOfTAttribute" />

      </s:sequence>
     </s:complexType>
    </s:element>
    <s:element name="ImportResponse">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="ImportResult"
type="s:int" />
      </s:sequence>
     </s:complexType>

    </s:element>
    <s:element name="ExportToCompendium">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="spaceID"
type="s:int" />
       <s:element minOccurs="0" maxOccurs="1" name="opt"
type="tns:ArrayOfTAttribute" />
      </s:sequence>
     </s:complexType>
    </s:element>

    <s:element name="ExportToCompendiumResponse">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1"
name="ExportToCompendiumResult" type="s:base64Binary" />
      </s:sequence>
     </s:complexType>
    </s:element>
    <s:element name="ImportFromCompendium">
```

```
      <s:complexType>

       <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="spaceID"
type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="copID" type="s:int"
/>
        <s:element minOccurs="0" maxOccurs="1" name="thisFile"
type="s:base64Binary" />
       </s:sequence>
      </s:complexType>
     </s:element>
     <s:element name="ImportFromCompendiumResponse">
      <s:complexType>

       <s:sequence>
        <s:element minOccurs="1" maxOccurs="1"
name="ImportFromCompendiumResult" type="s:int" />
       </s:sequence>
      </s:complexType>
     </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="ExportSoapIn">
   <wsdl:part name="parameters" element="tns:Export" />

  </wsdl:message>
  <wsdl:message name="ExportSoapOut">
   <wsdl:part name="parameters" element="tns:ExportResponse" />
  </wsdl:message>
  <wsdl:message name="ImportSoapIn">
   <wsdl:part name="parameters" element="tns:Import" />
  </wsdl:message>
  <wsdl:message name="ImportSoapOut">
   <wsdl:part name="parameters" element="tns:ImportResponse" />

  </wsdl:message>
  <wsdl:message name="ExportToCompendiumSoapIn">
   <wsdl:part name="parameters" element="tns:ExportToCompendium" />
  </wsdl:message>
  <wsdl:message name="ExportToCompendiumSoapOut">
   <wsdl:part name="parameters"
element="tns:ExportToCompendiumResponse" />
  </wsdl:message>
  <wsdl:message name="ImportFromCompendiumSoapIn">
   <wsdl:part name="parameters" element="tns:ImportFromCompendium"
/>

  </wsdl:message>
  <wsdl:message name="ImportFromCompendiumSoapOut">
```

```
   <wsdl:part name="parameters"
element="tns:ImportFromCompendiumResponse" />
  </wsdl:message>
  <wsdl:portType name="legacyServiceSoap">
   <wsdl:operation name="Export">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Exports a CoPe_it!
collaboration space.</wsdl:documentation>
    <wsdl:input message="tns:ExportSoapIn" />

    <wsdl:output message="tns:ExportSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="Import">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Imports a CoPe_it!
collaboration space.</wsdl:documentation>
    <wsdl:input message="tns:ImportSoapIn" />
    <wsdl:output message="tns:ImportSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="ExportToCompendium">

    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Overloaded method from
method “Export” to directly support exporting to Compendium maps. Exports
a CoPe_it! collaboration space to Compendium
format.</wsdl:documentation>
    <wsdl:input message="tns:ExportToCompendiumSoapIn" />
    <wsdl:output message="tns:ExportToCompendiumSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="ImportFromCompendium">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Imports a Compendium-
formatted map to CoPe_it!</wsdl:documentation>
    <wsdl:input message="tns:ImportFromCompendiumSoapIn" />
    <wsdl:output message="tns:ImportFromCompendiumSoapOut" />

   </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="legacyServiceSoap" type="tns:legacyServiceSoap">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="Export">
    <soap:operation soapAction="http://copeit.cti.gr/Export"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>

    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
```

```
   <wsdl:operation name="Import">
    <soap:operation                    soapAction="http://copeit.cti.gr/Import"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>

    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ExportToCompendium">
    <soap:operation  soapAction="http://copeit.cti.gr/ExportToCompendium"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>

    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ImportFromCompendium">
    <soap:operation
soapAction="http://copeit.cti.gr/ImportFromCompendium"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>

    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="legacyServiceSoap12" type="tns:legacyServiceSoap">
   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="Export">
    <soap12:operation                    soapAction="http://copeit.cti.gr/Export"
style="document" />

    <wsdl:input>
     <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="Import">
    <soap12:operation                    soapAction="http://copeit.cti.gr/Import"
style="document" />
```

```
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ExportToCompendium">
    <soap12:operation
soapAction="http://copeit.cti.gr/ExportToCompendium"    style="document"
/>

    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ImportFromCompendium">
    <soap12:operation
soapAction="http://copeit.cti.gr/ImportFromCompendium"
style="document" />

    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="legacyService">

   <wsdl:port name="legacyServiceSoap" binding="tns:legacyServiceSoap">
    <soap:address
location="http://localhost:1566/CopeItWebServices/repository.asmx" />
   </wsdl:port>
   <wsdl:port                                          name="legacyServiceSoap12"
binding="tns:legacyServiceSoap12">
    <soap12:address
location="http://localhost:1566/CopeItWebServices/repository.asmx" />
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Search Service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://copeit.cti.gr/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://copeit.cti.gr/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://copeit.cti.gr/">
      <s:element name="Search">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="query"
type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="options"
type="tns:ArrayOfTAttribute" />
          </s:sequence>

        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfTAttribute">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded"
name="TAttribute" nillable="true" type="tns:TAttribute" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="TAttribute">
        <s:sequence>

          <s:element minOccurs="0" maxOccurs="1" name="name"
type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="value"
type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:element name="SearchResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SearchResult"
type="tns:ArrayOfSearchResult" />
          </s:sequence>
```

```
      </s:complexType>
     </s:element>
     <s:complexType name="ArrayOfSearchResult">
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="unbounded"
name="searchResult" nillable="true" type="tns:searchResult" />
      </s:sequence>
     </s:complexType>
     <s:complexType name="searchResult">
      <s:sequence>

       <s:element minOccurs="1" maxOccurs="1" name="id" type="s:int" />
       <s:element minOccurs="0" maxOccurs="1" name="title" type="s:string"
/>
      </s:sequence>
     </s:complexType>
     <s:element name="throwException">
      <s:complexType />
     </s:element>
     <s:element name="throwExceptionResponse">
      <s:complexType />

     </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="SearchSoapIn">
   <wsdl:part name="parameters" element="tns:Search" />
  </wsdl:message>
  <wsdl:message name="SearchSoapOut">
   <wsdl:part name="parameters" element="tns:SearchResponse" />
  </wsdl:message>

  <wsdl:message name="throwExceptionSoapIn">
   <wsdl:part name="parameters" element="tns:throwException" />
  </wsdl:message>
  <wsdl:message name="throwExceptionSoapOut">
   <wsdl:part name="parameters" element="tns:throwExceptionResponse" />
  </wsdl:message>
  <wsdl:portType name="SearchingSoap">
   <wsdl:operation name="Search">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Searches all resources in
the repository of CoPe_it! for resources whose attribute values contain
query.</wsdl:documentation>

    <wsdl:input message="tns:SearchSoapIn" />
    <wsdl:output message="tns:SearchSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="throwException">
```

```
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web Method used to
return error messages</wsdl:documentation>
    <wsdl:input message="tns:throwExceptionSoapIn" />
    <wsdl:output message="tns:throwExceptionSoapOut" />
   </wsdl:operation>

  </wsdl:portType>
  <wsdl:binding name="SearchingSoap" type="tns:SearchingSoap">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="Search">
    <soap:operation                soapAction="http://copeit.cti.gr/Search"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>

     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="throwException">
    <soap:operation          soapAction="http://copeit.cti.gr/throwException"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>

     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="SearchingSoap12" type="tns:SearchingSoap">
   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="Search">
    <soap12:operation              soapAction="http://copeit.cti.gr/Search"
style="document" />
    <wsdl:input>

     <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="throwException">
    <soap12:operation        soapAction="http://copeit.cti.gr/throwException"
style="document" />
    <wsdl:input>
```

```
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Searching">
   <wsdl:port name="SearchingSoap" binding="tns:SearchingSoap">

    <soap:address
location="http://localhost:1566/CopeItWebServices/search.asmx" />
   </wsdl:port>
   <wsdl:port name="SearchingSoap12" binding="tns:SearchingSoap12">
    <soap12:address
location="http://localhost:1566/CopeItWebServices/search.asmx" />
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

User Management Service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://copeit.cti.gr/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://copeit.cti.gr/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
   <s:schema                                    elementFormDefault="qualified"
targetNamespace="http://copeit.cti.gr/">
    <s:element name="RegisterUser">
     <s:complexType>
      <s:sequence>
       <s:element        minOccurs="0"        maxOccurs="1"        name="u"
type="tns:TUser" />
      </s:sequence>
     </s:complexType>

    </s:element>
    <s:complexType name="TUser">
     <s:sequence>
```

```
        <s:element minOccurs="0" maxOccurs="1" name="name"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="password"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="FName"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="LName"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="email"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="photo"
type="s:base64Binary" />

        <s:element minOccurs="0" maxOccurs="1" name="address"
type="s:string" />
      </s:sequence>
    </s:complexType>
    <s:element name="RegisterUserResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1"
name="RegisterUserResult" type="s:int" />
        </s:sequence>
      </s:complexType>

    </s:element>
    <s:element name="UnRegisterUser">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="user_id"
type="s:int" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="UnRegisterUserResponse">

      <s:complexType />
    </s:element>
    <s:element name="LoadCoPs">
      <s:complexType />
    </s:element>
    <s:element name="LoadCoPsResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="LoadCoPsResult"
type="tns:ArrayOfTCop" />

        </s:sequence>
      </s:complexType>
    </s:element>
    <s:complexType name="ArrayOfTCop">
```

```
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="unbounded" name="TCop"
nillable="true" type="tns:TCop" />
      </s:sequence>
     </s:complexType>
     <s:complexType name="TCop">

      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="title" type="s:string"
/>
       <s:element minOccurs="0" maxOccurs="1" name="description"
type="s:string" />
       <s:element minOccurs="0" maxOccurs="1" name="logo"
type="s:base64Binary" />
      </s:sequence>
     </s:complexType>
     <s:element name="CreateNewCoP">
      <s:complexType>
       <s:sequence>

        <s:element minOccurs="0" maxOccurs="1" name="title"
type="s:string" />
       </s:sequence>
      </s:complexType>
     </s:element>
     <s:element name="CreateNewCoPResponse">
      <s:complexType>
       <s:sequence>
        <s:element minOccurs="1" maxOccurs="1"
name="CreateNewCoPResult" type="s:int" />
       </s:sequence>

      </s:complexType>
     </s:element>
     <s:element name="DeleteCoP">
      <s:complexType>
       <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="cop_id"
type="s:int" />
       </s:sequence>
      </s:complexType>
     </s:element>

     <s:element name="DeleteCoPResponse">
      <s:complexType />
     </s:element>
     <s:element name="AddUserToCoP">
      <s:complexType>
       <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="user_id"
type="s:int" />
```

```
        <s:element minOccurs="1" maxOccurs="1" name="cop_id"
type="s:int" />
       </s:sequence>

     </s:complexType>
    </s:element>
    <s:element name="AddUserToCoPResponse">
     <s:complexType />
    </s:element>
    <s:element name="RemoveUserFromCoP">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="user_id"
type="s:int" />

       <s:element minOccurs="1" maxOccurs="1" name="cop_id"
type="s:int" />
      </s:sequence>
     </s:complexType>
    </s:element>
    <s:element name="RemoveUserFromCoPResponse">
     <s:complexType />
    </s:element>
    <s:element name="LoadCopUsers">
     <s:complexType>

      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="cop_id"
type="s:int" />
      </s:sequence>
     </s:complexType>
    </s:element>
    <s:element name="LoadCopUsersResponse">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1"
name="LoadCopUsersResult" type="tns:ArrayOfTUser" />

      </s:sequence>
     </s:complexType>
    </s:element>
    <s:complexType name="ArrayOfTUser">
     <s:sequence>
      <s:element minOccurs="0" maxOccurs="unbounded" name="TUser"
nillable="true" type="tns:TUser" />
     </s:sequence>
    </s:complexType>
    <s:element name="LoadUserProfileByID">

     <s:complexType>
      <s:sequence>
```

```
        <s:element minOccurs="1" maxOccurs="1" name="user_id"
type="s:int" />
       </s:sequence>
      </s:complexType>
     </s:element>
     <s:element name="LoadUserProfileByIDResponse">
      <s:complexType>
       <s:sequence>

        <s:element minOccurs="0" maxOccurs="1"
name="LoadUserProfileByIDResult" type="tns:TUser" />
       </s:sequence>
      </s:complexType>
     </s:element>
     <s:element name="LoadUserProfileByName">
      <s:complexType>
       <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="name"
type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="pass"
type="s:string" />

       </s:sequence>
      </s:complexType>
     </s:element>
     <s:element name="LoadUserProfileByNameResponse">
      <s:complexType>
       <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
name="LoadUserProfileByNameResult" type="tns:TUser" />
       </s:sequence>
      </s:complexType>

     </s:element>
     <s:element name="UpdateUserProfile">
      <s:complexType>
       <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="u"
type="tns:TUser" />
       </s:sequence>
      </s:complexType>
     </s:element>
     <s:element name="UpdateUserProfileResponse">

      <s:complexType />
     </s:element>
     <s:element name="throwException">
      <s:complexType />
     </s:element>
     <s:element name="throwExceptionResponse">
      <s:complexType />
```

```
    </s:element>
  </s:schema>

 </wsdl:types>
 <wsdl:message name="RegisterUserSoapIn">
  <wsdl:part name="parameters" element="tns:RegisterUser" />
 </wsdl:message>
 <wsdl:message name="RegisterUserSoapOut">
  <wsdl:part name="parameters" element="tns:RegisterUserResponse" />
 </wsdl:message>
 <wsdl:message name="UnRegisterUserSoapIn">
  <wsdl:part name="parameters" element="tns:UnRegisterUser" />

 </wsdl:message>
 <wsdl:message name="UnRegisterUserSoapOut">
  <wsdl:part name="parameters" element="tns:UnRegisterUserResponse" />
 </wsdl:message>
 <wsdl:message name="LoadCoPsSoapIn">
  <wsdl:part name="parameters" element="tns:LoadCoPs" />
 </wsdl:message>
 <wsdl:message name="LoadCoPsSoapOut">
  <wsdl:part name="parameters" element="tns:LoadCoPsResponse" />

 </wsdl:message>
 <wsdl:message name="CreateNewCoPSoapIn">
  <wsdl:part name="parameters" element="tns:CreateNewCoP" />
 </wsdl:message>
 <wsdl:message name="CreateNewCoPSoapOut">
  <wsdl:part name="parameters" element="tns:CreateNewCoPResponse" />
 </wsdl:message>
 <wsdl:message name="DeleteCoPSoapIn">
  <wsdl:part name="parameters" element="tns:DeleteCoP" />

 </wsdl:message>
 <wsdl:message name="DeleteCoPSoapOut">
  <wsdl:part name="parameters" element="tns:DeleteCoPResponse" />
 </wsdl:message>
 <wsdl:message name="AddUserToCoPSoapIn">
  <wsdl:part name="parameters" element="tns:AddUserToCoP" />
 </wsdl:message>
 <wsdl:message name="AddUserToCoPSoapOut">
  <wsdl:part name="parameters" element="tns:AddUserToCoPResponse" />

 </wsdl:message>
 <wsdl:message name="RemoveUserFromCoPSoapIn">
  <wsdl:part name="parameters" element="tns:RemoveUserFromCoP" />
 </wsdl:message>
 <wsdl:message name="RemoveUserFromCoPSoapOut">
  <wsdl:part                                                    name="parameters"
element="tns:RemoveUserFromCoPResponse" />
 </wsdl:message>
```

```
 <wsdl:message name="LoadCopUsersSoapIn">
  <wsdl:part name="parameters" element="tns:LoadCopUsers" />

 </wsdl:message>
 <wsdl:message name="LoadCopUsersSoapOut">
  <wsdl:part name="parameters" element="tns:LoadCopUsersResponse" />
 </wsdl:message>
 <wsdl:message name="LoadUserProfileByIDSoapIn">
  <wsdl:part name="parameters" element="tns:LoadUserProfileByID" />
 </wsdl:message>
 <wsdl:message name="LoadUserProfileByIDSoapOut">
  <wsdl:part                                                    name="parameters"
element="tns:LoadUserProfileByIDResponse" />

 </wsdl:message>
 <wsdl:message name="LoadUserProfileByNameSoapIn">
  <wsdl:part name="parameters" element="tns:LoadUserProfileByName" />
 </wsdl:message>
 <wsdl:message name="LoadUserProfileByNameSoapOut">
  <wsdl:part                                                    name="parameters"
element="tns:LoadUserProfileByNameResponse" />
 </wsdl:message>
 <wsdl:message name="UpdateUserProfileSoapIn">
  <wsdl:part name="parameters" element="tns:UpdateUserProfile" />

 </wsdl:message>
 <wsdl:message name="UpdateUserProfileSoapOut">
  <wsdl:part name="parameters" element="tns:UpdateUserProfileResponse"
/>
 </wsdl:message>
 <wsdl:message name="throwExceptionSoapIn">
  <wsdl:part name="parameters" element="tns:throwException" />
 </wsdl:message>
 <wsdl:message name="throwExceptionSoapOut">
  <wsdl:part name="parameters" element="tns:throwExceptionResponse" />

 </wsdl:message>
 <wsdl:portType name="userManagementSoap">
  <wsdl:operation name="RegisterUser">
   <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Registers a new user in
CoPe_it!.</wsdl:documentation>
   <wsdl:input message="tns:RegisterUserSoapIn" />
   <wsdl:output message="tns:RegisterUserSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="UnRegisterUser">

   <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Removes a user from
CoPe_it! by passing the participant's unique
identifier.</wsdl:documentation>
```

```
    <wsdl:input message="tns:UnRegisterUserSoapIn" />
    <wsdl:output message="tns:UnRegisterUserSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="LoadCoPs">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Loads a list of the
Communities in which the requestor has at least “view”
privileges.</wsdl:documentation>
    <wsdl:input message="tns:LoadCoPsSoapIn" />
    <wsdl:output message="tns:LoadCoPsSoapOut" />

   </wsdl:operation>
   <wsdl:operation name="CreateNewCoP">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Creates a new Community
with a particular title. The requestor automatically becomes the community’s
administrator.</wsdl:documentation>
    <wsdl:input message="tns:CreateNewCoPSoapIn" />
    <wsdl:output message="tns:CreateNewCoPSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="DeleteCoP">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Removes a Community
from CoPe_it!.</wsdl:documentation>

    <wsdl:input message="tns:DeleteCoPSoapIn" />
    <wsdl:output message="tns:DeleteCoPSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="AddUserToCoP">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Sets a CoPe_it! user as a
member of a particular Community.</wsdl:documentation>
    <wsdl:input message="tns:AddUserToCoPSoapIn" />
    <wsdl:output message="tns:AddUserToCoPSoapOut" />
   </wsdl:operation>

   <wsdl:operation name="RemoveUserFromCoP">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Removes a user from a
Community.</wsdl:documentation>
    <wsdl:input message="tns:RemoveUserFromCoPSoapIn" />
    <wsdl:output message="tns:RemoveUserFromCoPSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="LoadCopUsers">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Loads the members of a
particular Community.</wsdl:documentation>
    <wsdl:input message="tns:LoadCopUsersSoapIn" />

    <wsdl:output message="tns:LoadCopUsersSoapOut" />
   </wsdl:operation>
```

```
   <wsdl:operation name="LoadUserProfileByID">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Load the user's personal
information. It can also be used to check the existence of a
user_id.</wsdl:documentation>
    <wsdl:input message="tns:LoadUserProfileByIDSoapIn" />
    <wsdl:output message="tns:LoadUserProfileByIDSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="LoadUserProfileByName">

    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Load the user's personal
information. It can also be used to check the existence of a user given his
name/password.</wsdl:documentation>
    <wsdl:input message="tns:LoadUserProfileByNameSoapIn" />
    <wsdl:output message="tns:LoadUserProfileByNameSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="UpdateUserProfile">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Replaces a user profile
with a new one.</wsdl:documentation>
    <wsdl:input message="tns:UpdateUserProfileSoapIn" />
    <wsdl:output message="tns:UpdateUserProfileSoapOut" />

   </wsdl:operation>
   <wsdl:operation name="throwException">
    <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web Method used to
return error messages</wsdl:documentation>
    <wsdl:input message="tns:throwExceptionSoapIn" />
    <wsdl:output message="tns:throwExceptionSoapOut" />
   </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding                                    name="userManagementSoap"
type="tns:userManagementSoap">

   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="RegisterUser">
    <soap:operation              soapAction="http://copeit.cti.gr/RegisterUser"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="UnRegisterUser">
    <soap:operation          soapAction="http://copeit.cti.gr/UnRegisterUser"
style="document" />
```

```
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="LoadCoPs">
    <soap:operation                soapAction="http://copeit.cti.gr/LoadCoPs"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="CreateNewCoP">
    <soap:operation          soapAction="http://copeit.cti.gr/CreateNewCoP"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="DeleteCoP">
    <soap:operation                soapAction="http://copeit.cti.gr/DeleteCoP"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="AddUserToCoP">
    <soap:operation          soapAction="http://copeit.cti.gr/AddUserToCoP"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
```

```
  </wsdl:operation>
  <wsdl:operation name="RemoveUserFromCoP">
    <soap:operation   soapAction="http://copeit.cti.gr/RemoveUserFromCoP"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

  </wsdl:operation>
  <wsdl:operation name="LoadCopUsers">
    <soap:operation              soapAction="http://copeit.cti.gr/LoadCopUsers"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

  </wsdl:operation>
  <wsdl:operation name="LoadUserProfileByID">
    <soap:operation    soapAction="http://copeit.cti.gr/LoadUserProfileByID"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

  </wsdl:operation>
  <wsdl:operation name="LoadUserProfileByName">
    <soap:operation
soapAction="http://copeit.cti.gr/LoadUserProfileByName"  style="document"
/>
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

  </wsdl:operation>
  <wsdl:operation name="UpdateUserProfile">
    <soap:operation        soapAction="http://copeit.cti.gr/UpdateUserProfile"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
```

```
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="throwException">
    <soap:operation soapAction="http://copeit.cti.gr/throwException"
style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="userManagementSoap12"
type="tns:userManagementSoap">
   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="RegisterUser">
    <soap12:operation soapAction="http://copeit.cti.gr/RegisterUser"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="UnRegisterUser">
    <soap12:operation soapAction="http://copeit.cti.gr/UnRegisterUser"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="LoadCoPs">
    <soap12:operation soapAction="http://copeit.cti.gr/LoadCoPs"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
```

```
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="CreateNewCoP">
    <soap12:operation          soapAction="http://copeit.cti.gr/CreateNewCoP"
style="document" />
    <wsdl:input>
     <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
     <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="DeleteCoP">
    <soap12:operation                soapAction="http://copeit.cti.gr/DeleteCoP"
style="document" />
    <wsdl:input>
     <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
     <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="AddUserToCoP">
    <soap12:operation          soapAction="http://copeit.cti.gr/AddUserToCoP"
style="document" />
    <wsdl:input>
     <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
     <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="RemoveUserFromCoP">
    <soap12:operation
soapAction="http://copeit.cti.gr/RemoveUserFromCoP"     style="document"
/>
    <wsdl:input>
     <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
     <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="LoadCopUsers">
    <soap12:operation          soapAction="http://copeit.cti.gr/LoadCopUsers"
style="document" />
```

```
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="LoadUserProfileByID">
    <soap12:operation
soapAction="http://copeit.cti.gr/LoadUserProfileByID" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="LoadUserProfileByName">
    <soap12:operation
soapAction="http://copeit.cti.gr/LoadUserProfileByName"  style="document"
/>
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="UpdateUserProfile">
    <soap12:operation     soapAction="http://copeit.cti.gr/UpdateUserProfile"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="throwException">
    <soap12:operation         soapAction="http://copeit.cti.gr/throwException"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
```

```
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="userManagement">
   <wsdl:port                                    name="userManagementSoap"
binding="tns:userManagementSoap">
    <soap:address
location="http://localhost:1566/CopeItWebServices/userManagement.asmx"
/>
   </wsdl:port>

   <wsdl:port                                  name="userManagementSoap12"
binding="tns:userManagementSoap12">
    <soap12:address
location="http://localhost:1566/CopeItWebServices/userManagement.asmx"
/>
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Reasoning Service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://copeit.cti.gr/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://copeit.cti.gr/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
   <s:schema                                  elementFormDefault="qualified"
targetNamespace="http://copeit.cti.gr/">
    <s:element name="AnnotateDiscussion">
     <s:complexType>
      <s:sequence>
       <s:element      minOccurs="0"      maxOccurs="1"      name="algorithm"
type="s:string" />
       <s:element minOccurs="0" maxOccurs="1" name="cnt">
        <s:complexType mixed="true">

         <s:sequence>
          <s:any />
         </s:sequence>
        </s:complexType>
```

```
        </s:element>
       </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="AnnotateDiscussionResponse">

      <s:complexType>
       <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
name="AnnotateDiscussionResult">
         <s:complexType mixed="true">
          <s:sequence>
           <s:any />
          </s:sequence>
         </s:complexType>
        </s:element>

       </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ValidateDiscussion">
      <s:complexType>
       <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="cnt">
         <s:complexType mixed="true">
          <s:sequence>

           <s:any />
          </s:sequence>
         </s:complexType>
        </s:element>
        <s:element minOccurs="0" maxOccurs="1" name="opt"
type="tns:ArrayOfTAttribute" />
       </s:sequence>
      </s:complexType>
    </s:element>
    <s:complexType name="ArrayOfTAttribute">

      <s:sequence>
       <s:element minOccurs="0" maxOccurs="unbounded"
name="TAttribute" nillable="true" type="tns:TAttribute" />
      </s:sequence>
    </s:complexType>
    <s:complexType name="TAttribute">
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="name"
type="s:string" />
       <s:element minOccurs="0" maxOccurs="1" name="value"
type="s:string" />
      </s:sequence>
```

```
    </s:complexType>
    <s:element name="ValidateDiscussionResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1"
name="ValidateDiscussionResult" type="s:boolean" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ListReasoningAlgorithms">

      <s:complexType />
    </s:element>
    <s:element name="ListReasoningAlgorithmsResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1"
name="ListReasoningAlgorithmsResult" type="tns:ArrayOfString" />
        </s:sequence>
      </s:complexType>
    </s:element>

    <s:complexType name="ArrayOfString">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="unbounded" name="string"
nillable="true" type="s:string" />
      </s:sequence>
    </s:complexType>
    <s:element name="throwException">
      <s:complexType />
    </s:element>
    <s:element name="throwExceptionResponse">

      <s:complexType />
    </s:element>
  </s:schema>
 </wsdl:types>
 <wsdl:message name="AnnotateDiscussionSoapIn">
  <wsdl:part name="parameters" element="tns:AnnotateDiscussion" />
 </wsdl:message>
 <wsdl:message name="AnnotateDiscussionSoapOut">
  <wsdl:part name="parameters"
element="tns:AnnotateDiscussionResponse" />

 </wsdl:message>
 <wsdl:message name="ValidateDiscussionSoapIn">
  <wsdl:part name="parameters" element="tns:ValidateDiscussion" />
 </wsdl:message>
 <wsdl:message name="ValidateDiscussionSoapOut">
  <wsdl:part name="parameters" element="tns:ValidateDiscussionResponse"
/>
```

```
  </wsdl:message>
  <wsdl:message name="ListReasoningAlgorithmsSoapIn">
    <wsdl:part name="parameters" element="tns:ListReasoningAlgorithms" />

  </wsdl:message>
  <wsdl:message name="ListReasoningAlgorithmsSoapOut">
    <wsdl:part name="parameters"
element="tns:ListReasoningAlgorithmsResponse" />
  </wsdl:message>
  <wsdl:message name="throwExceptionSoapIn">
    <wsdl:part name="parameters" element="tns:throwException" />
  </wsdl:message>
  <wsdl:message name="throwExceptionSoapOut">
    <wsdl:part name="parameters" element="tns:throwExceptionResponse" />

  </wsdl:message>
  <wsdl:portType name="ReasoningSoap">
    <wsdl:operation name="AnnotateDiscussion">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Annotates a discussion
described in an XML document. Annotations include winning and defeating
alternatives and positions.</wsdl:documentation>
      <wsdl:input message="tns:AnnotateDiscussionSoapIn" />
      <wsdl:output message="tns:AnnotateDiscussionSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="ValidateDiscussion">

      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Provided an XML
document, the system validates it against the required XML
schema.</wsdl:documentation>
      <wsdl:input message="tns:ValidateDiscussionSoapIn" />
      <wsdl:output message="tns:ValidateDiscussionSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="ListReasoningAlgorithms">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Returns a list of
algorithms available as reasoning mechanisms.</wsdl:documentation>
      <wsdl:input message="tns:ListReasoningAlgorithmsSoapIn" />
      <wsdl:output message="tns:ListReasoningAlgorithmsSoapOut" />

    </wsdl:operation>
    <wsdl:operation name="throwException">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web Method used to
return error messages</wsdl:documentation>
      <wsdl:input message="tns:throwExceptionSoapIn" />
      <wsdl:output message="tns:throwExceptionSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ReasoningSoap" type="tns:ReasoningSoap">
```

```
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="AnnotateDiscussion">
    <soap:operation      soapAction="http://copeit.cti.gr/AnnotateDiscussion"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="ValidateDiscussion">
    <soap:operation       soapAction="http://copeit.cti.gr/ValidateDiscussion"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="ListReasoningAlgorithms">
    <soap:operation
soapAction="http://copeit.cti.gr/ListReasoningAlgorithms"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
   <wsdl:operation name="throwException">
    <soap:operation            soapAction="http://copeit.cti.gr/throwException"
style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>

   </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ReasoningSoap12" type="tns:ReasoningSoap">
   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="AnnotateDiscussion">
```

```
    <soap12:operation   soapAction="http://copeit.cti.gr/AnnotateDiscussion"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ValidateDiscussion">
    <soap12:operation    soapAction="http://copeit.cti.gr/ValidateDiscussion"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ListReasoningAlgorithms">
    <soap12:operation
soapAction="http://copeit.cti.gr/ListReasoningAlgorithms"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="throwException">
    <soap12:operation         soapAction="http://copeit.cti.gr/throwException"
style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>

    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Reasoning">
   <wsdl:port name="ReasoningSoap" binding="tns:ReasoningSoap">
    <soap:address
location="http://localhost:1566/CopeItWebServices/reasoning.asmx" />
   </wsdl:port>
```

```
   <wsdl:port name="ReasoningSoap12" binding="tns:ReasoningSoap12">
     <soap12:address
location="http://localhost:1566/CopeItWebServices/reasoning.asmx" />
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

DTD for the Discussion document described in XML.

```
<!ELEMENT DISCUSSION
(SUBJECT,CLOSE_DATE,ISSUE*,PREFERENCE*,INCOSISTENTPREFERE
NCEGROUPS?)>

<!ELEMENT ISSUE
(SUBMITTER,DATE,SUBJECT,REFERENCES,(ALTERNATIVE|ISSUE)*)>
<!ELEMENT ALTERNATIVE
(SUBMITTER,DATE,SUBJECT,REFERENCES,POSITION*)>
<!ELEMENT POSITION
(SUBMITTER,DATE,SUBJECT,REFERENCES,(POSITION|VOTES?)*)>
<!ELEMENT PREFERENCE
(SUBMITTER,DATE,SUBJECT,FIRST_REF,SECOND_REF,RELATION,REF
ERENCES,POSITION*)>
<!ELEMENT INCOSISTENTPREFERENCEGROUPS
(INCOSISTENTPREFERENCEGROUP*)>
<!ELEMENT INCOSISTENTPREFERENCEGROUP (PREFERENCE*)>
<!ELEMENT VOTES (AGAINST,NEUTRAL,INFAVOR)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT SUBMITTER (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT SUBJECT (#PCDATA)>

<!ELEMENT REFERENCES ((URL|COMMENTS|ATTACHMENT)*)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT COMMENTS (#PCDATA)>
<!ELEMENT ATTACHMENT (#PCDATA)>
```

```
<!ELEMENT CLOSE_DATE (#PCDATA)>
<!ELEMENT FIRST_REF (#PCDATA)>
<!ELEMENT SECOND_REF (#PCDATA)>
<!ELEMENT RELATION (#PCDATA)>

<!ELEMENT AGAINST (#PCDATA)>
<!ELEMENT NEUTRAL (#PCDATA)>
<!ELEMENT INFAVOR (#PCDATA)>

<!ATTLIST DISCUSSION ID ID #REQUIRED>
<!ATTLIST DISCUSSION Issues CDATA #REQUIRED>
<!ATTLIST DISCUSSION Positions CDATA #REQUIRED>
<!ATTLIST DISCUSSION Alternatives CDATA #REQUIRED>
<!ATTLIST DISCUSSION Preferences CDATA #REQUIRED>
<!ATTLIST DISCUSSION MaximumPositionLevel CDATA #REQUIRED>

<!ATTLIST ISSUE ID ID #REQUIRED>
<!ATTLIST ISSUE Level CDATA #REQUIRED>
<!ATTLIST ISSUE Score CDATA #REQUIRED>
<!ATTLIST ISSUE Win (TRUE|FALSE) "FALSE">
<!ATTLIST ISSUE Active (TRUE|FALSE) "TRUE">

<!ATTLIST REFERENCES Attachments CDATA #REQUIRED>
<!ATTLIST REFERENCES URLs CDATA #REQUIRED>

<!ATTLIST ALTERNATIVE ID ID #REQUIRED>
<!ATTLIST ALTERNATIVE Active (TRUE|FALSE) #REQUIRED>
<!ATTLIST ALTERNATIVE Win (TRUE|FALSE) "FALSE">
<!ATTLIST ALTERNATIVE Level CDATA #REQUIRED>
<!ATTLIST ALTERNATIVE Score CDATA #REQUIRED>

<!ATTLIST POSITION ID ID #REQUIRED>
<!ATTLIST POSITION Status (FAVOR|AGAINST) #REQUIRED>
<!ATTLIST POSITION Active (TRUE|FALSE) #REQUIRED>
```

```
<!ATTLIST POSITION Level CDATA #REQUIRED>
<!ATTLIST POSITION MinWeight CDATA #REQUIRED>
<!ATTLIST POSITION MaxWeight CDATA #REQUIRED>
<!ATTLIST POSITION Score CDATA #REQUIRED>

<!ATTLIST PREFERENCE ID ID #REQUIRED>
<!ATTLIST PREFERENCE Active (TRUE|FALSE) #REQUIRED>
<!ATTLIST PREFERENCE Conflict (TRUE|FALSE) #REQUIRED>
<!ATTLIST PREFERENCE Level CDATA #IMPLIED>

<!ATTLIST VOTES Against CDATA #REQUIRED>
<!ATTLIST VOTES InFavor CDATA #REQUIRED>
<!ATTLIST VOTES Neutral CDATA #REQUIRED>
```

Import/Export Service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://copeit.cti.gr/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://copeit.cti.gr/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://copeit.cti.gr/">
      <s:element name="Export">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="spaceID"
type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="opt"
type="tns:ArrayOfTAttribute" />
          </s:sequence>

        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfTAttribute">
        <s:sequence>
```

```
      <s:element minOccurs="0" maxOccurs="unbounded"
name="TAttribute" nillable="true" type="tns:TAttribute" />
     </s:sequence>
    </s:complexType>
    <s:complexType name="TAttribute">
     <s:sequence>

      <s:element minOccurs="0" maxOccurs="1" name="name"
type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="value"
type="s:string" />
     </s:sequence>
    </s:complexType>
    <s:element name="ExportResponse">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="ExportResult"
type="s:base64Binary" />
      </s:sequence>

     </s:complexType>
    </s:element>
    <s:element name="Import">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="spaceID"
type="s:int" />
       <s:element minOccurs="1" maxOccurs="1" name="copID" type="s:int"
/>
       <s:element minOccurs="0" maxOccurs="1" name="cnt"
type="s:base64Binary" />
       <s:element minOccurs="0" maxOccurs="1" name="opt"
type="tns:ArrayOfTAttribute" />

      </s:sequence>
     </s:complexType>
    </s:element>
    <s:element name="ImportResponse">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="ImportResult"
type="s:int" />
      </s:sequence>
     </s:complexType>

    </s:element>
    <s:element name="ExportToCompendium">
     <s:complexType>
      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="spaceID"
type="s:int" />
```

```
        <s:element minOccurs="0" maxOccurs="1" name="opt"
type="tns:ArrayOfTAttribute" />
       </s:sequence>
      </s:complexType>
    </s:element>

    <s:element name="ExportToCompendiumResponse">
      <s:complexType>
       <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
name="ExportToCompendiumResult" type="s:base64Binary" />
       </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ImportFromCompendium">
      <s:complexType>

       <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="spaceID"
type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="copID" type="s:int"
/>
        <s:element minOccurs="0" maxOccurs="1" name="thisFile"
type="s:base64Binary" />
       </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ImportFromCompendiumResponse">
      <s:complexType>

       <s:sequence>
        <s:element minOccurs="1" maxOccurs="1"
name="ImportFromCompendiumResult" type="s:int" />
       </s:sequence>
      </s:complexType>
    </s:element>
   </s:schema>
  </wsdl:types>
  <wsdl:message name="ExportSoapIn">
   <wsdl:part name="parameters" element="tns:Export" />

  </wsdl:message>
  <wsdl:message name="ExportSoapOut">
   <wsdl:part name="parameters" element="tns:ExportResponse" />
  </wsdl:message>
  <wsdl:message name="ImportSoapIn">
   <wsdl:part name="parameters" element="tns:Import" />
  </wsdl:message>
  <wsdl:message name="ImportSoapOut">
   <wsdl:part name="parameters" element="tns:ImportResponse" />
```

```
 </wsdl:message>
 <wsdl:message name="ExportToCompendiumSoapIn">
  <wsdl:part name="parameters" element="tns:ExportToCompendium" />
 </wsdl:message>
 <wsdl:message name="ExportToCompendiumSoapOut">
  <wsdl:part name="parameters"
element="tns:ExportToCompendiumResponse" />
 </wsdl:message>
 <wsdl:message name="ImportFromCompendiumSoapIn">
  <wsdl:part name="parameters" element="tns:ImportFromCompendium"
/>

 </wsdl:message>
 <wsdl:message name="ImportFromCompendiumSoapOut">
  <wsdl:part name="parameters"
element="tns:ImportFromCompendiumResponse" />
 </wsdl:message>
 <wsdl:portType name="legacyServiceSoap">
  <wsdl:operation name="Export">
   <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Exports a CoPe_it!
collaboration space.</wsdl:documentation>
   <wsdl:input message="tns:ExportSoapIn" />

   <wsdl:output message="tns:ExportSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="Import">
   <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Imports a CoPe_it!
collaboration space.</wsdl:documentation>
   <wsdl:input message="tns:ImportSoapIn" />
   <wsdl:output message="tns:ImportSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ExportToCompendium">

   <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Overloaded method from
method “Export” to directly support exporting to Compendium maps. Exports
a CoPe_it! collaboration space to Compendium
format.</wsdl:documentation>
   <wsdl:input message="tns:ExportToCompendiumSoapIn" />
   <wsdl:output message="tns:ExportToCompendiumSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ImportFromCompendium">
   <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Imports a Compendium-
formatted map to CoPe_it!</wsdl:documentation>
   <wsdl:input message="tns:ImportFromCompendiumSoapIn" />
   <wsdl:output message="tns:ImportFromCompendiumSoapOut" />

  </wsdl:operation>
```

```
  </wsdl:portType>
  <wsdl:binding name="legacyServiceSoap" type="tns:legacyServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="Export">
      <soap:operation                    soapAction="http://copeit.cti.gr/Export"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>

      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Import">
      <soap:operation                    soapAction="http://copeit.cti.gr/Import"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>

      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ExportToCompendium">
      <soap:operation  soapAction="http://copeit.cti.gr/ExportToCompendium"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>

      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ImportFromCompendium">
      <soap:operation
soapAction="http://copeit.cti.gr/ImportFromCompendium"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>

      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="legacyServiceSoap12" type="tns:legacyServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
```

```
   <wsdl:operation name="Export">
     <soap12:operation                    soapAction="http://copeit.cti.gr/Export"
style="document" />

     <wsdl:input>
       <soap12:body use="literal" />
     </wsdl:input>
     <wsdl:output>
       <soap12:body use="literal" />
     </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="Import">
     <soap12:operation                    soapAction="http://copeit.cti.gr/Import"
style="document" />

     <wsdl:input>
       <soap12:body use="literal" />
     </wsdl:input>
     <wsdl:output>
       <soap12:body use="literal" />
     </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ExportToCompendium">
     <soap12:operation
soapAction="http://copeit.cti.gr/ExportToCompendium"    style="document"
/>

     <wsdl:input>
       <soap12:body use="literal" />
     </wsdl:input>
     <wsdl:output>
       <soap12:body use="literal" />
     </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="ImportFromCompendium">
     <soap12:operation
soapAction="http://copeit.cti.gr/ImportFromCompendium"
style="document" />

     <wsdl:input>
       <soap12:body use="literal" />
     </wsdl:input>
     <wsdl:output>
       <soap12:body use="literal" />
     </wsdl:output>
   </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="legacyService">

   <wsdl:port name="legacyServiceSoap" binding="tns:legacyServiceSoap">
```

```
    <soap:address
location="http://localhost:1566/CopeItWebServices/legacyService.asmx" />
   </wsdl:port>
   <wsdl:port                                          name="legacyServiceSoap12"
binding="tns:legacyServiceSoap12">
    <soap12:address
location="http://localhost:1566/CopeItWebServices/legacyService.asmx" />
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Event Query Service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://copeit.cti.gr/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://copeit.cti.gr/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
 <wsdl:types>
  <s:schema                                    elementFormDefault="qualified"
targetNamespace="http://copeit.cti.gr/">
   <s:element name="ListEvents">
    <s:complexType>
     <s:sequence>
      <s:element    minOccurs="0"    maxOccurs="1"    name="EventTypes"
type="s:string" />
      <s:element      minOccurs="0"      maxOccurs="1"      name="Filters"
type="s:string" />
     </s:sequence>

    </s:complexType>
   </s:element>
   <s:element name="ListEventsResponse">
    <s:complexType>
     <s:sequence>
      <s:element  minOccurs="0"  maxOccurs="1"  name="ListEventsResult"
type="tns:ArrayOfTEvent" />
     </s:sequence>
    </s:complexType>
   </s:element>

   <s:complexType name="ArrayOfTEvent">
```

```
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="unbounded" name="TEvent"
nillable="true" type="tns:TEvent" />
      </s:sequence>
     </s:complexType>
     <s:complexType name="TEvent">
      <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="actor"
type="s:string" />
       <s:element minOccurs="1" maxOccurs="1" name="when"
type="s:dateTime" />

       <s:element minOccurs="1" maxOccurs="1" name="affectedObjectID"
type="s:int" />
       <s:element minOccurs="0" maxOccurs="1" name="action"
type="s:string" />
       <s:element minOccurs="0" maxOccurs="1" name="description"
type="s:string" />
      </s:sequence>
     </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="ListEventsSoapIn">
    <wsdl:part name="parameters" element="tns:ListEvents" />

  </wsdl:message>
  <wsdl:message name="ListEventsSoapOut">
    <wsdl:part name="parameters" element="tns:ListEventsResponse" />
  </wsdl:message>
  <wsdl:portType name="EventQuerySoap">
    <wsdl:operation name="ListEvents">
      <wsdl:documentation
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Loads a list of the Events
that took place in CoPe_it! and matches the particular
filters.</wsdl:documentation>
      <wsdl:input message="tns:ListEventsSoapIn" />

      <wsdl:output message="tns:ListEventsSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="EventQuerySoap" type="tns:EventQuerySoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="ListEvents">
      <soap:operation soapAction="http://copeit.cti.gr/ListEvents"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />

      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
```

```
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="EventQuerySoap12" type="tns:EventQuerySoap">
   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="ListEvents">

    <soap12:operation                soapAction="http://copeit.cti.gr/ListEvents"
style="document" />
    <wsdl:input>
     <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap12:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="EventQuery">
   <wsdl:port name="EventQuerySoap" binding="tns:EventQuerySoap">
    <soap:address
location="http://localhost:1566/CopeItWebServices/event.asmx" />
   </wsdl:port>
   <wsdl:port name="EventQuerySoap12" binding="tns:EventQuerySoap12">
    <soap12:address
location="http://localhost:1566/CopeItWebServices/event.asmx" />
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Appendix B: e-Logbook Web Service Description Files (WADL)

In the following, we give the WADL service descriptions of services “create activity” and “query activity”.

Before running the Web services provided by e-Logbook, the service consumers need to authenticate themselves. They can do this by the following HTTP request:

`http://e-Logbook.epfl.ch/?email=[login email]&password=[password],`

where `[login email]` and `[password]` are replaced by the user login email and passwords, respectively.

Create Activity Service

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="wadl_documentation.xsl"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://research.sun.com/wadl wadl.xsd"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:elb="http://e-Logbook.epfl.ch/e-LogbookSchema"
 xmlns="http://research.sun.com/wadl/2006/10">

<grammars>
   <include
    href="e-LogbookSchema.xsd"/>
 </grammars>

<resources base="http://e-Logbook.epfl.ch/">

<resource path="create/activity.xml">
    <method name="POST" id="createActivity">
     <request>
     <representation mediaType="application/xml"
element="elb:RequestCreateActivitySet">
        <doc title="Create Activity Request">Create activity request.</doc>
       </representation>
      </request>
      <response>
       <representation mediaType="application/xml" element="elb:activity">
        <doc title="Create Activity Response">Create activity response.</doc>
       </representation>
       <fault id="elbError" mediaType="application/xml"
element="elb:errors" >
        <doc>Standard e-Logbook Error.</doc>
```

```
      </fault>
    </response>
  </method>
 </resource>

</resources>
</application>
```

Schema (RequestCreateActivitySet and activity)

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://e-Logbook.epfl.ch/e-LogbookSchema"
 xmlns="http://e-Logbook.epfl.ch/e-LogbookSchema"
 elementFormDefault="qualified">

<xs:element name="RequestCreateActivitySet">
        <xs:complexType name="RequestCreateActivtyType">
                <xs:sequence>
                        <xs:element name="name" type="xs:string" />
                        <xs:element      name="description"      type="xs:string"
minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>

<xs:element name="activity">
        <xs:complexType name="ReponseCreateActivityType">
                <xs:sequence>
                        <xs:element name="created-at" type="xs:datetime" />
                        <xs:element name="default-role-id" type="xs:integer" />
                        <xs:element name="description" type="xs:string" />
                        <xs:element name="id" type="xs:integer" />
                        <xs:element  name="initiator-actor-id"  type="xs:integer"
/>
                        <xs:element name="name" type="xs:string" />
                        <xs:element                    name="public-invitation-style"
type="string" />
                        <xs:element name="updated-at" type="xs:datetime" />
                </xs:sequence>
        </xs:complexType>
</xs:element>

</xs:schema>
```

The service “create activity” uses HTTP POST method. The service consumer sends request in XML data with format defined as “RequestCreatedActivitySet”. The service responds with XML data with

format defined as “activity”. The e-Logbook schema defines the “RequestCreatedActivitySet” and “ResponseCreatedActivitySet” formats. In case there is fault of the function, the service will respond with an error message in XML.

The “RequestCreatedActivitySet” format defines the input parameters:

- *name*: string type
- *description*: string type: optional

The following is an example of request, in XML:

```
<RequestCreateActivitySet>
        <name> Activity 1 </name>
        <description> Activity 1 description </description>
</RequestCreateActivitySet>
```

The “activity” format defines the output elements:

- *create-at*: datatime type
- *default-role-id*: integer type
- *description*: string type
- *initiator-actor-id*: integer type
- *name*: string type
- *public-invitation-style*: string type
- *updated-at*: datetime type

Query Activity Service

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="wadl_documentation.xsl"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://research.sun.com/wadl wadl.xsd"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:elb="http://e-Logbook.epfl.ch/e-LogbookSchema"
 xmlns="http://research.sun.com/wadl/2006/10">

<grammars>
   <include
    href="e-LogbookSchema.xsd"/>
 </grammars>

<resources base="http://e-Logbook.epfl.ch/">

<resource path="activities.xml">
    <method name="GET" id="queryActivities">
      <request>
```

```
        <param name="status" type="xsd:string" required="false"
style="query">
          <doc>Status of activity.</doc>
        </param>
       </request>
       <response>
        <representation mediaType="application/xml"
element="elb:activities">
          <doc title="Activity Query Result">Activities Query Result.</doc>
        </representation>
        <fault id="elbError" mediaType="application/xml"
element="elb:errors" >
          <doc>Standard e-Logbook Error.</doc>
        </fault>
       </response>
     </method>
   </resource>

</resources>
</application>
```

Schema (activities and RoleType)

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://e-Logbook.epfl.ch/e-LogbookSchema"
 xmlns="http://e-Logbook.epfl.ch/e-LogbookSchema"
 elementFormDefault="qualified">

<xs:element name="activities">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="activity" type="ResultActivitiesType"
minOccurs="0" maxOccurs="100" />
                </xs:sequence>
        </xs:complexType>
</xs:element>

<xs:complexType name="ResultActivitiesType">
        <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="description" type="xs:string" />
                <xs:element name="id" type="xs:string" />
                <xs:sequence>
                        <xs:element name="role" type="RoleType" minOccurs="1"
maxOccurs="100" />
                </xs:sequence>
        </xs:sequence>
</xs:complexType>
```

```
Schema (RoleType):
<xs:complexType name="RoleType">
        <xs:sequence>
                <xs:element name="role_id" type="xs:string" />
                <xs:element name="name" type="xs:string" />
                <xs:element name="status" type="xs:string" />
                <xs:element name="invitation_id" type="xs:string" />
        </xs:sequence>
</xs:complexType>

</xs:schema>
```

The service “query activity” uses HTTP GET method. The service consumer sends request to the service without the need of parameters. The service automatically replies with the activity information of the actor of the current session. The service responds with XML data with format defined as “activities”. The e-Logbook schema defines the “activities” format.

The “activities” format defines that the output result XML contains a sequence of “ResultActivitiesType” element. The “ResultActivitiesType” element includes the following sub-elements:

- *name*: string type
- *description*: string type
- *id*: string type
- sequence of *role*: Each role element contains *role_id*, *name*, *status*, and *s*. All are string type.

The following is an example of result, in XML:

```
<activities>
   <activity>
      <id>24</id>
      <name>Activity 1</name>
      <description>Activity 1 description.</description>
      <role>
         <role_id>71</role_id>
         <name>Members</name>
         <status>joined</status>
         <invitation_id>53</invitation_id>
      </role>
      <role>
         <role_id>70</role_id>
         <name>Administrator</name>
         <status>joined</status>
         <invitation_id>52</invitation_id>
      </role>
   </activity>

   <activity>
```

```
        <id>17</id>
        <name>Activity 2</name>
        <description>Activity 2 description</description>
        <role>
            <role_id>41</role_id>
            <name>Administrator</name>
            <status>joined</status>
            <invitation_id>43</invitation_id>
         </role>
    </activity>
</activities>
```