ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

FRAMEWORK FOR DISTRIBUTED OFFICE APPLICATION

TR/42

July 1987

Free copies of this document are available from ECMA,
European Computer Manufacturers Association
114 Rue du Rhône – 1204 Geneva (Switzerland)

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

FRAMEWORK FOR DISTRIBUTED OFFICE APPLICATION

TR/42

July 1987

BRIEF HISTORY

ECMA, ISO and CCITT have all started standardization work on distributed office applications, the first which are Message Handling and Directory. In addition to the continued work in these areas, other distributed office applications are now under study, or planned study. Examples are document filing and retrieval, printing and the general field of security.

In CCITT Recommendation X.409 and X.410 there is a powerful notation for specifying some external interactions of distributed applications (the remote operations notation). The ECMA view is that this methodology is likely to be a key factor in the overall success of OSI standardization. An ECMA Technical Report has already been published on that subject.

The purpose of this Report is to emphasize the need for and propose a general Framework for distributed office applications based on the Remote Operations. It examines the general need for office applications of a supportive nature versus other office applications of a more productive nature and how the cooperation between supportive and productive office application works. It also defines certain general design principles in order to ensure that standards for different distributed office applications, which have to function together, are designed in a coherent manner.

This Report is one of a set of standards and reports for Open Systems Interconnection. Open Systems Interconnection standards are intended to facilitate homogeneous interconnection between heterogeneous information processing systems. This Report is within the framework for the coordination of standards for Open Systems Interconnection which is defined by ISO 7498.

This Report is based on the practical experience of ECMA member companies world-wide, and on the results of their active participation in the current work of ISO, CCITT and national standard bodies in Europe and the USA. It represents a pragmatic and widely based consensus.

A particular emphasis of this Report is to specify the homogeneous externally visible and verifiable characteristics needed for interconnection compatibility, while avoiding unnecessary constraints upon and changes to the heterogeneous internal design and implementation of the information processing systems to be interconnected.

In the interest of a rapid and effective standardization, this Report is oriented towards urgent and well understood needs. It is intended to be capable of modular extension to cover future developments in technology and needs.

Adopted as an ECMA Technical Report at the General Assembly of 25 June 1987.

TABLE OF CONTENTS

1. INTRODUCTION

An office can be seen as a group of people performing related tasks. This broad view ranges from the handful of people in a small independent agency to the vast array of white-collar workers and others attending to the business of a multinational corporation.

In recent years, advances in computing and communications technology have expanded greatly the set of tools, available to aid office workers in the performance of their jobs. Chief among these advancements is the whole area of distributed office applications. This field is characterized by the availability of a diverse array of electronic office equipments and enabling software, dispersed geographically but capable of acting cooperatively by means of communications media and techniques.

A wide variety of distributed office applications are now becoming available from a growing number of manufacturers. It is extremely important to facilitate the homogeneous interconnection between these heterogeneous offerings in order to realize the full benefits available from this emerging field. Hence, this Report proposes a Framework for Distributed Office Applications: a general model and a set of design principles to insure that different distributed office applications will function together in a coherent manner.

This Report defines a common Framework that can be used to develop standards for distributed office applications. It provides the applicable need to:

- allow a modular, simple and expandable development of related products;
- facilitate their implementation;
- allow their interworking;
- and reduce the development costs.

Although mainly intended for distributed office applications the content of this Report may be used in any other data processing environment which may take benefit of concepts, structures and organization described in this Report.

1.1 Need and Application

Standards for Open Systems Interconnection permit the functional components of an application to be distributed over a network. Some applications may be distributed in order to reduce costs, e.g. in the case of an office system connected by a high speed Local Area Network, where some expensive resources may be shared. Other applications may be distributed for administrative or functional reasons, e.g. in the case of a world wide electronic mail system. These are examples of distributed applications whose design differs from the traditional "single host" approach. In general, the increasing range of technological options permits a number of design approaches with quite a different distribution of costs and computational load on the various system elements, such as the desk top devices that populate a modern office and the "back room" devices with which these users are networked.

This Report provides unifying principles for structuring distributed applications, and designing the protocols. At the same time, this Report is compatible with the traditional application approach of "single host accessed by multiple terminals".

The applicability of this Report to the different classes of applications is as follows:

- fully centralized applications residing on a single host and accessed by multiple terminals:
  a) there is no need for access protocols to make use of the applications (the terminals may use Virtual Terminal protocols, which are outside the scope of this Report), but
  b) the principles described in this Report can be advantageously used in the design of fully centralized applications.
- centralized applications which all reside on a single host which is accessed by user nodes (e.g. work stations and personal computers):

a) standard access protocols are needed, but

b) there is no need for protocols between different functional units of the applications;

- distributed applications potentially residing on multiple network hosts (which may be owned by different companies) accessed by user nodes:

a) standard access protocols are needed, and

b) the different functional units of the applications, residing on multiple hosts, need standard system protocols to communicate.

The office applications and similar applications, accessed by user nodes, are referred to as productive applications and, at a minimum, enable: distribution , storage and retrieval, and imaging of documents. The list of productive applications will grow in the future (e.g. scanning of documents, data base access, etc.).

The operation of the productive office applications requires additional support from several supportive applications and facilities to enable: location of resources or individuals on the network (directory), insurance that only authorized individuals have access to sensitive resources (authentication and access control), and efficient transfer of documents between servers (third party transfer). In addition, a common time base throughout the network is required.

1.2 Scope

This Report describes a Framework for Distributed Office Applications in an office environment. Its content can be grouped in four major parts:

- An introductory part (clause 1) in which references, definitions and abbreviations are collected together;
- A statement of the requirements to be satisfied by this framework (clause 2);
- A tutorial approach to the definition of the fundamental concepts associated with distributed office applications: basic concepts needed to define the relationships between the generic communicating elements (clause 3), the naming concepts (clause 4) and the security concepts (clause 5) needed for distributed office applications;
- A brief description of the supportive applications available in the framework (clause 6);
- A description of how the supportive applications are used by the productive applications in the framework (clause 7);
- Guidelines for the design of protocols for distributed office applications (clause 8).

This Framework will be the basis for future definitions of applications and protocols in ECMA on this subject.

1.3 Rationale

A number of finished and emerging standards in ECMA, ISO and CCITT are addressing applications that can be considered part of a set of interrelated distributed office applications, primarily associated with office systems. Examples of these standards are the ECMA-93 MIDA and emerging ECMA standards for a MIDA Mailbox Access Protocol and OSI Directory Service; CCITT Recommendations from 1984 on MHS and the emerging joint ISO and CCITT work on MHS and Directory for the 1988 time frame. Other, closely related, applications such as Document Filing & Retrieval, and Printing are being undertaken as future work items in ECMA and other standardization groups.

Several emerging ECMA standards plan to use the remote operations structure defined in CCITT Rec. X.410 and further detailed in ECMA TR/31 "Remote Operations: Concepts, Notation and Connection-oriented Mappings" and lately in the joint CCITT and ISO drafts on "Remote Operations Service Element". The remote operations concept offers a good common basis for the definition of new distributed applications.

This ECMA work results from the consequent need to provide an agreed, stable and common base for future ECMA work on distributed office applications. This should also be applicable to the work of ISO and the CCITT.

1.4 References

ECMA-93	MIDA - Distributed Application for Message Interchange.
ECMA-	MIDA - Mailbox Service Description and Mailbox Access Protocol Specification.
ECMA TR/31	Remote Operations: Concepts, Notation and Connection-oriented Mappings.
ECMA TR/32	OSI Directory Access Service and Protocol.
ISO 7498	Information processing systems - Open Systems Interconnection - Basic Reference Model.
ISO 7498/DAD 2	Addendum on Security Architecture (Working Draft).
ISO 8649/2	Information processing systems - Open Systems Interconnection - Service definition for common application-service-elements - Part 2: Association control
ISO DIS 8824.2	Specification of Abstract Syntax Notation One (ASN.1).
ISO DIS 8825.2	Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).
ISO DIS 9066/1-2	Information processing systems - Text Communication - Reliable Transfer Part 1 and Part 2.
ISO DP 9072/1-2	Information processing systems - Text Communication - Remote Operations Part 1 and Part 2.
ISO DP 9594/1-7	The Directory
CCITT Rec. X.409	Message Handling Systems: Presentation Transfer Syntax and Notation.
CCITT Rec. X.410	Message Handling Systems: Remote Operations and Reliable Transfer Server.

1.5 Definitions

1.5.1 General Terminology

The following terms are used with the meanings defined in ISO 7498:

Application Layer
application-process
application-entity
application-service-element
Presentation Layer
presentation-connection
protocol
service definition.

The following terms are used with the meanings defined in ISO 8824:

macro
macro notation
Coordinated Universal Time

The following terms are used with the meanings defined in ECMA TR/31 or ISO 9072:

Association Control Service Element
binding
object
object instance
object model
object type
operation
Reliable Transfer Service Element
remote operation
remote operation notation
Remote Operation Service Element.

The following terms are used with the meanings defined in ECMA-93:

Message transfer agent
message transfer system
P1
P2

1.5.2 Specific Terminology

The following terms are used with the meaning defined below.

access-control-list

A set of control-attributes. It is a list, associated with a security-object or a group of security-objects. The list contains the names of security-subjects and the type of access that may be granted.

access-control-policy

A set of rules, part of a security-policy, by which human users, or their representatives, are authenticated and by which access by these users to services and security-objects is granted or denied.

access-context

The context, in terms of such variables as location, time of day, level of security of the underlying associations, etc, in which an access to a security-object is made.

authentication-policy

A set of rules, part of an access-control-policy, by which credentials are matched against claims of identity.

authorization-policy

A set of rules, part of an access-control-policy, by which access by security-subjects to security-objects is granted or denied. An authorization-policy may be defined in terms of access-control-lists, capabilities or attributes assigned to security-subjects, security-objects or both.

capability

As used in the context of security, a form of privilege-attribute. It is a token (recognized by a process managing access to security-objects) possession of which grants access to the security-objects to which the token applies.

control-attributes

Attributes, associated with a security-object that, when matched against the privilege-attributes of a security-subject, are used to grant or deny access to the security-object.

credentials

Data that serve to establish the claimed identity of a security-subject relative to a given security-domain.

data-object-format-specification

A data type, in the ASN.1 sense, that is defined independently of x-access-protocols.

distributed(-office)-application

A set of information processing resources distributed over one or more open systems which provides a well defined set of functionality to (human) users, to assist a given (office) task.

node

A data processing facility that provides information processing resources as part of a network. A node may support user-application-processes, server-application-processes or a combination of both kinds of processes.

privilege-attributes

Attributes, associated with a security-subject that , when matched against control-attributes of a security-object are used to grant or deny access to that security-object.

security-administrator

An authority (a person or group of people) responsible for implementing the security-policy for a security-domain.

security-attributes

A general term covering both privilege-attributes and control-attributes. The use of security-attributes is defined by a security-policy.

security-domain

A bounded group of security-objects and security-subjects to which applies a single security-policy executed by a single security-administrator.

security-facility

Procedures, processes, mechanism or set thereof, that models a security related function.

security-object

An entity in a passive role to which access is granted or denied according to an authorization-policy.

security-policy

A set of rules that specify the procedures and mechanisms required to maintain the security of a system(s), and the security-objects and the security-subjects under the purview of the policy.

security-subject

An entity in an active role that is granted or denied access to security-objects according to an authorization-policy.

server-application-process

An application-process that implements all or part of the functionality defined by an x-service-definition.

third-party-transfer

An information transfer between two server-application-processes acting cooperatively on instructions from a third application-process.

user

A software entity, part of the user-application-process, that interacts with the human user and, on the other hand, with distributed(-office)-applications on behalf of the human user.

user-application-process

An application-process that contains a user and one or more clients of distributed(-office)-applications (e.g. x-client, y-client, etc.).

x-,y-,z-, ...

Generic placeholders for specific application names.

x-access-protocol

The protocol between any x-client and any x-server.

x-application

A distributed(-office)-application of a certain kind, e.g., an Electronic Mail Application or a Filing and Retrieval Application.

x-application-interface

The interface to an x-application, as visible between a user and an x-client.

x-client

That part of an x-application that is contained in an application-process acting as an x-user.

x-server

That part of an x-application that is contained in one server-application-process and that provides all or part of the functionality specified by an x-service-definition (synonymous with x-system-agent, or x-agent used in other documents).

x-service-definition

The definition of the functionality of an x-application, as visible between x-clients and the x-system.

x-system

The collection of x-servers that together provide the functionality defined by the x-service-definition.

x-system-protocol

Protocol used between the x-servers.

x-user

A role assumed by an entity when using an x-application.

1.5.3 Abbreviations

ACL	Access-control-list
ACSE	Association Control Service Element
AE	Application-entity
ASE	Application-service-element
OSI	Open Systems Interconnection
ROSE	Remote Operations Service Element
RTSE	Reliable Transfer Service Element
TPT	Third-party-transfer
TPTAE	Third-party-transfer application-entity
UTC	Coordinated Universal Time

2. REQUIREMENTS

In this clause two sets of requirements are listed: the first set describes the standards environment into which this Report has to fit; the second set describes the requirements on a framework suitable for, but not necessarily limited to office systems.

2.1 Requirements on the Report

The requirements to be satisfied by this Report are as follows:

Stability

The Report shall provide a stable base for future use by ECMA distributed-office-applications.

ISO & CCITT

Compatibility is required with emerging work on distributed-office-applications in ISO and CCITT. The proposals in the Report shall be a basis for constructive ECMA contributions to ISO and to CCITT.

Upper layer mappings

The Report is limited to framework aspects in the Application Layer of OSI. It shall de-couple distributed-office-applications specification from the differences between, and possible instabilities of, the various OSI "execution environments" likely to be used, and rely on the Remote Operations Notation.

Portability

The framework specification shall support a high level of abstraction, such that the specifications of distributed-office-applications interactions are highly portable.

Precision

The framework specification shall be such that the specifications of distributed-office-applications interactions are likely to be unambiguous.

Usability

The recommended design principles shall be such that application designers, without previous experience, can quickly achieve proficiency.

Productivity

The recommended design principles shall be such that application designers can quickly specify, develop and deliver the interactive components of distributed-office-applications.

Implementability

The recommended design principles shall be such that complete and compatible implementations can readily be derived for the interactive components of distributed-office-applications.

2.2 Requirements on a Framework for Distributed-office-applications

2.2.1 Introduction

Distributed-office-applications are used by an integrated distributed office system, consisting of user nodes and server nodes linked by a network. The user nodes access the server nodes via the network, using access protocols.

In such an environment, data processing applications, that within a single host act as a single piece, have been split among the different intelligent components of the system. This splitting has led to the need for standardization of inter-relationships between the different parts of an application.

In this environment the distributed-office-applications should satisfy the following objectives:

- Make easier the implementation of application-processes developed for a distributed environment based on micro-processors and large or medium sized mainframes interconnected through local area network or wide are a network means;
- Reduce the processing delay time for document related activities such as document filing and retrieval, document distribution, printing, etc., and group communication related activities such as interpersonal messaging, user directory and authentication processes, etc.;
- Allow concurrent processing of different tasks within the distributed office system;
- Reduce the overall size of an office system and facilitate its modular extension.

2.2.2 Functional Requirements

Amongst the services that may be required by the user from a distributed office system, and which the framework should address, are:

- Inter-personal messaging, to communi?cate with other users.
- Group communication, to communicate with groups of users.
- Conversion, to allow the interchange of documents with different presentation protocols or character coding.
- Filing and retrieval of documents, to allow an ordered filing and multi-key retrieval of documents.
- Input and output of documents to the distributed office system from different physical devices, such as scanners, printers, etc.
- Directory, to know where are and how to access remote communication elements, applications or users.
- Authentication, to avoid unauthorized access to the different applications.
- A time base which is locally accessed, for purposes such as time stamping messages and files throughout the network.
- Direct access to remote servers (e.g. Videotex) and users (e.g. via Teletex).
- Indirect communication (store and forward) with remote systems, to transfer non real-time information or to access other networks for example Message Transfer System.
- Data transfer between different applications or remote servers (third-party-transfer).

The above list of applications will grow in the future (e.g. data base access). Most of these additional applications are likely to be "productive applications" because their main purpose is to provide specific facilities to office workers.

The typical usage of these applications needs a high degree of integration. For example, a document may be fetched from a mailbox server, stored on a document filing and retrieval server, and printed on a printing server.

The operation of some of the above distributed-office-applications (more usually Group Communication, Document Filing and Retrieval, Printing) may need the use of other applications which perform supportive application roles (e.g. Directory, Authentication).

However, applications and their users should not be impacted by the way supportive applications are carried out. In particular, the actual distribution should be as transparent to the user as possible.

Supportive applications are not necessarily visible to the human user, but they do enable secure, reliable and smooth operation of the overall system.

2.2.3 Design Requirements

The design of the protocols should ensure:

Stability: the recommended design principles shall be such that distributed-office-application protocol designers can specify highly stable distributed applications interactions, with comprehensive use of common supportive applications.

Modular design:

a) minimizing interdependencies between different office applications of a yet open-ended list of productive applications;

b) enabling the required high degree of integration.

Common style of protocol design.

Homogeneous usage of supportive applications and facilities.

Homogeneous usage of the Remote Operations Service Element.

Security: of various levels specified by the user.

Simplicity: this is a key point from the user's point of view of distributed-office-applications. This means that the user application should not be involved in the way its request is managed by the application.

This Framework for Distributed Office Applications specifies the principles of interaction between the various applications.

3. BASIC CONCEPTS

3.1 Introduction

Distributed-office-applications are categorized as a number of applications which, from the user point of view, constitute an integrated office system.

A distributed office system consists of nodes connected through a network. While this network exists for the benefit of human users, the word "user" will be applied in a more general sense, including any application-process or human user. When a human user is specifically implied, that term will be used.

A user node is the device with which a human user directly interacts. It provides direct interactive functions.

A server node is a device that manages resources which are shared among many users.

A human user's interactions with a distributed office system may cause a number of activities to be performed on a number of nodes. How these separate activities on one node are represented by processes, tasks, etc, is not the concern of this report. The interactions between one activity on one node and one activity on another node are represented in the OSI model by interactions between a pair of application process invocations, one on each node. An application process invocation executes the functionality of an OSI application-process.

For clarity, clause 3 describes interactions in terms of application-processes.Each functional entity described in clause 3 is related to an application-process; each separate activity executing the functions is related to a separate application process invocation.

3.2 Client-Server Model

This clause adopts a tutorial approach to explain the distribution of a single x-application, an application of a certain kind.

3.2.1 Non-distributed Single Application

In a non-distributed single x-application, the user (in this case software which interacts directly with the x-application on behalf of a human user) and the x-application are co-

located. The user interacts with the x-application through an x-application-interface, which is usually proprietary and is not proposed for standardization (see Figure 1).

If an x-application is a candidate for distribution, an x-service-definition is needed as a separation line for potential future distribution.

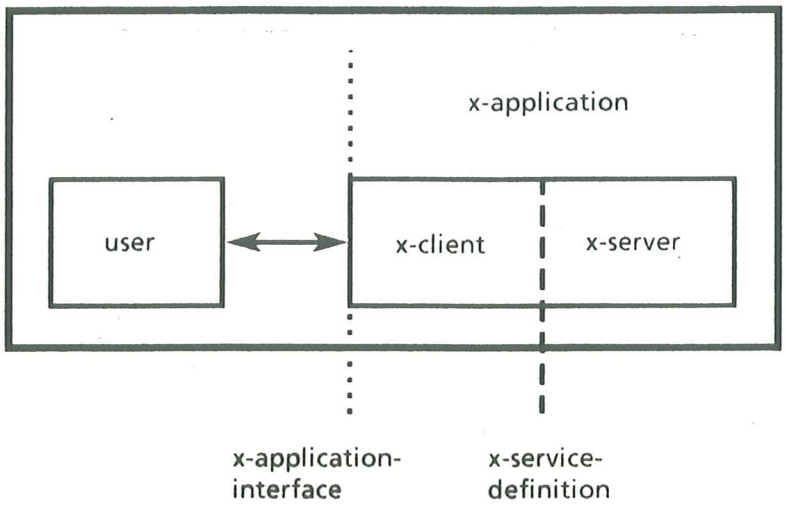


Figure 1 - Non-distributed Office Application

3.2.2 Distributed Single Application

The distribution of an x-application should be transparent to the user, so that the x-application-interface will not change. The x-client is co-located with the user. The user and the x-client are together within one application-process. The special case where the user is interacting with a human user is identified by the use of the term user-application-process.

The x-server is generally remote from the user. The x-server is part of an application-process which is termed a server-application-process.

An x-client and an x-server communicate over the network by means of an x-access-protocol. There may be several independent interactions between an x-client and an x-server.

The new situation is depicted in Figure 2 which shows the x-service-definition of Figure 1 expanded into a "dashed" box encapsulating the x-access-protocol.

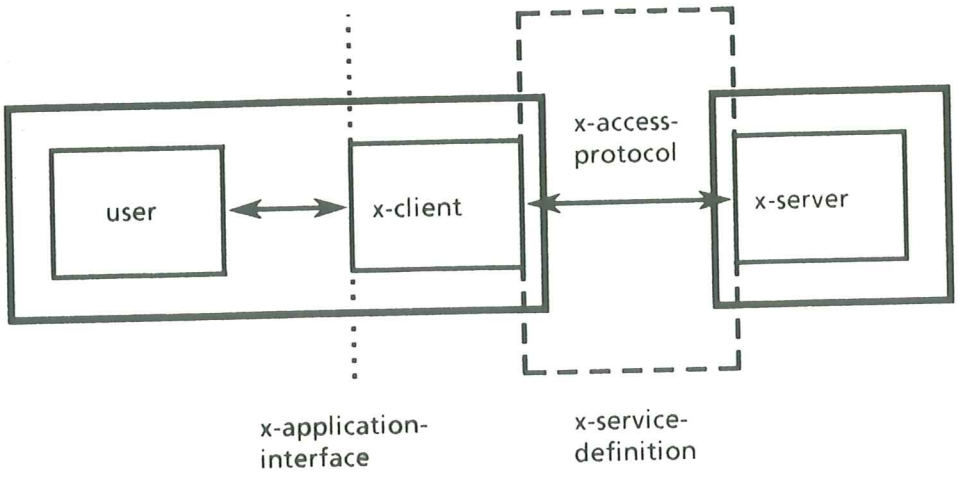


Figure 2 - Distributed-office-application

In the case of distribution, the x-service-definition and the x-access-protocol need to be standardized.

3.2.3 Client/Server OSI Communication

An x-access-protocol is the standard way for an x-client to gain access to its remote x-server. The following model shows how OSI principles are used to specify an x-access-protocol.

According to the OSI Reference Model, a protocol is used between peer entities. In Figure 2, OSI communications occur only within the dashed box. Thus, the OSI communication peer entities are inside, not outside that dashed box.

In compliance with the OSI Reference Model, the x-client and the x-server are considered application processes, and have application-entities associated with them. The application-entities are part of the Application Layer of the OSI Reference Model and contain sets of application-service-elements. The application-service-elements provide the communication functions, in accordance with the service definition, to the x-client and the x-server, and implement the x-access-protocol. In doing so, an application-service-element may use services provided by other application-service-elements in the same application-entity, and by the Presentation Layer of the OSI Reference Model.

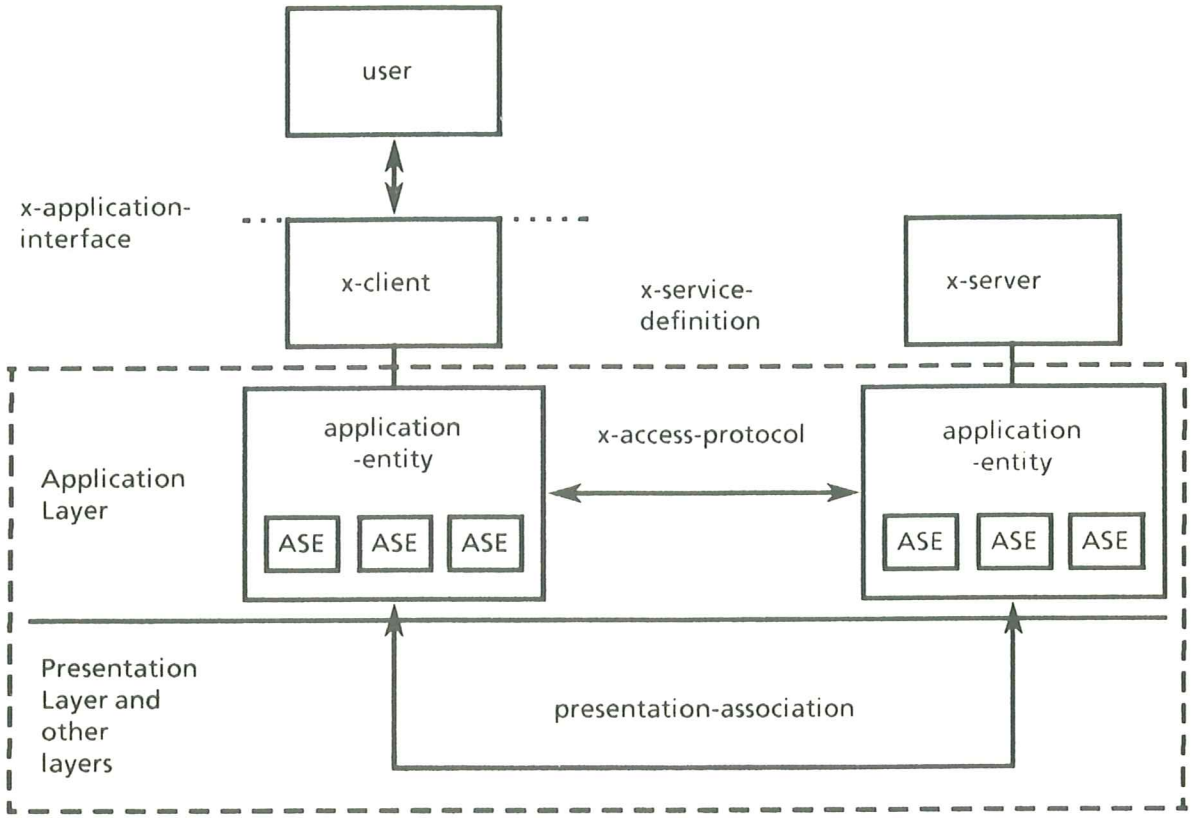


Figure 3 - Distributed-office-application with OSI Communication

In Figure 3, the dashed box is expanded to show a more detailed model of the OSI communication between the x-client and the x-server.

In detail, interactions take place between application entity invocations. Discrete sets of interactions between the same pair of application process invocations, if required, are performed between discrete pairs of application entity invocations.

However, for most practical purposes, it is not necessary to refer to the above detailed structure. Instead, the model of x-client/x-server communication by means of an x-access-protocol is generally sufficient for discussing the structure of distributed-office-applications.

Additional details about the client/server communication are specified in clause 3.4.

3.2.4 Object Model of a Distributed-office-application

The interworking of the application-processes of a distributed-office-application requires a shared conceptual schema describing the shared universe of discourse.

The typical universe of discourse is perceived of objects and relationships between them and may provide a classification of objects.

The client/server model is considered as an application oriented tool to develop conceptual schema. The object model has broader view and is more abstract. The components of a distributed-office-application (e.g., mailbox, filing cabinet, etc.) are all addressed as objects.

The tools which were developed within the object model are used here for the specification of the conceptual schema. The conceptual schema is the basis for the service-definition.

It is worth noting that another kind of object type is used in the context of distributed-office-applications. These are data objects (e.g. ASN.1 data types, message contents, Interpersonal Message body parts, private document formats). The abstract syntax of these data objects is defined independently of x-access-protocols and x-system-protocols.

3.3 Functional Model

3.3.1 Multiple Applications in an Integrated System

An integrated office system is formed by a set of office applications (e.g. Message Transfer, Document Filing and Retrieval and Printing). The integration of the office applications is performed by the user. The user interacts with the human user and, on behalf of the human user with the set of office applications. Interactions between the user and an x-server are performed via the x-client. Interaction between clients is performed via the user. This is depicted in Figure 4.

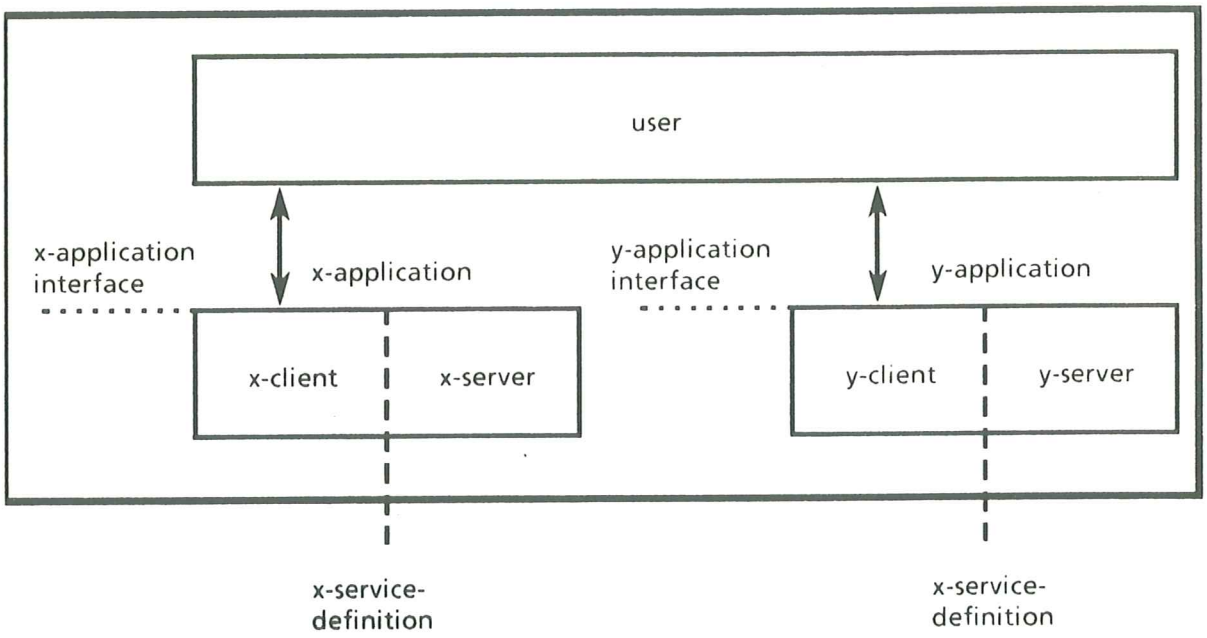


Figure 4 - Multiple Non-distributed Office Applications

3.3.2 Multiple Distributed-office-applications

The user and the clients are in one application-process which is termed the user-application-process (see Figure 5). Several user-application-processes may coexists on one user node, but these are kept separate within the framework.

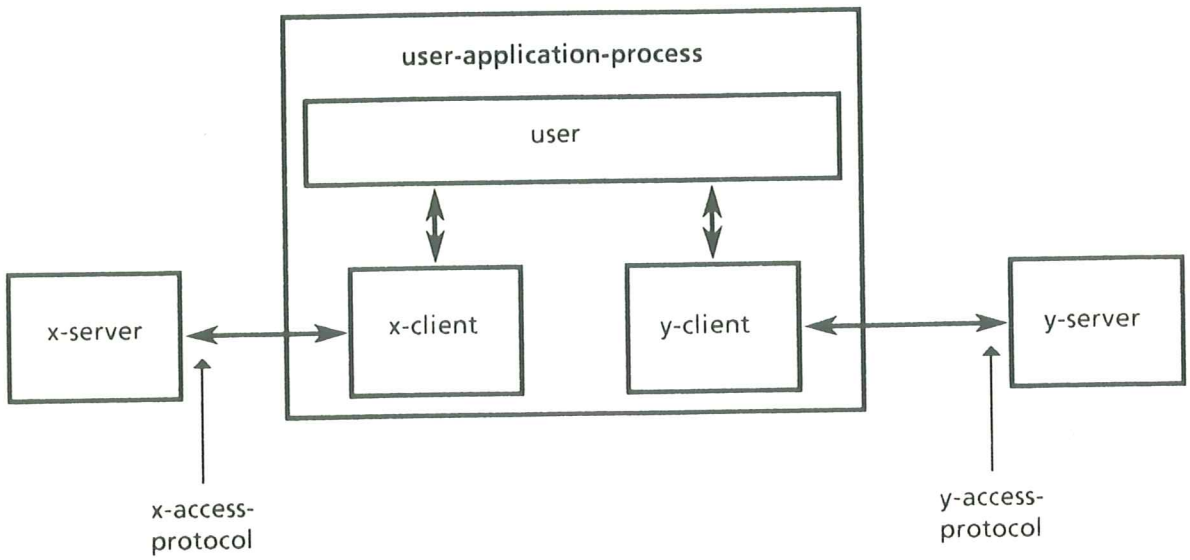


Figure 5 - Multiple Distributed-office-application

3.3.3 Organization of Servers

3.3.3.1 System of Servers of the Same Type

There can be a second distribution step, distributing the functionality of the server part of an x-application to several x-servers on several nodes. The set of x-servers is termed an x-system. Each x-server is functionally equivalent in that it supports the same x-access-protocol. Each x-server is within one node.

An x-system may consist of:

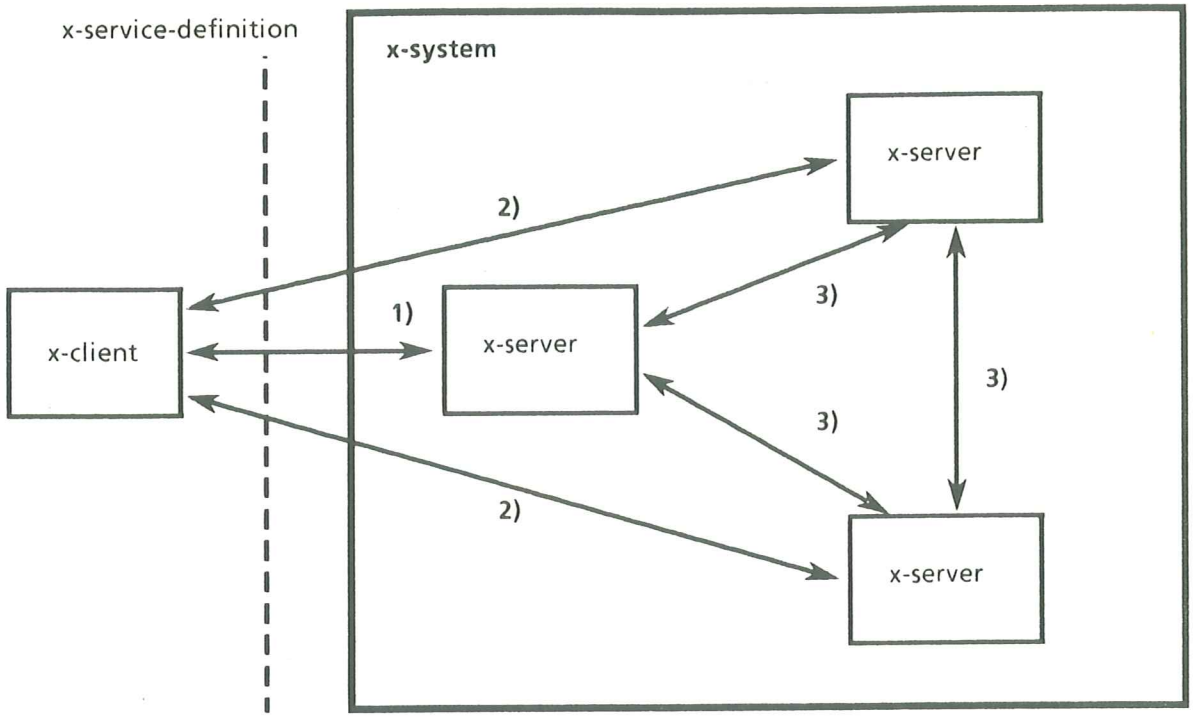
1) a single x-server;

2) several non-interacting x-servers;

3) several interacting x-servers.

The interaction between an x-client and an x-server is governed by an x-access-protocol (see Figure 6). These protocols are subject to specific x-application standardization and are not addressed in detail herein.

Several x-server connected through a network may interact to form the total x-system. In that case, they cooperate by means of an x-system-protocol. These protocols are subject to specific x-application standardization and are not addressed in detail herein.



Legend: 1) x-access-protocol
2) x-access-protocol (optional)
3) x-system-protocol (if required)

Figure 6 - X-system

The subset of the x-access-protocol available between the x-client and a particular x-server depends on the x-service-definition and the partitioning of the x-server. For x-servers which are maintaining a distributed information base, for example, the x-access-protocol may contain a hint mechanism (which means that the contacted x-server returns a hint to the x-client as to which other x-server(s) to contact in order to find a particular piece of information), if an x-system-protocol does not exist, or if the x-server is not not able, or unwilling to obtain this in a way transparent to the x-client.

Note that an x-client is not introduced for x-system-protocols, although the same OSI concepts of application-process and application-entity, etc., are applied. The inherent asymmetry of the client/server model may not be useful in the design of some system protocols.

3.3.3.2 A Server as a User of Another Server

Sometimes one system (x-system) will use another system (y-system). This is modeled by describing the x-server that needs to use the y-system as having the role of a y-user of the y-application. In this case, a y-client exists within the application-process containing the x-server.

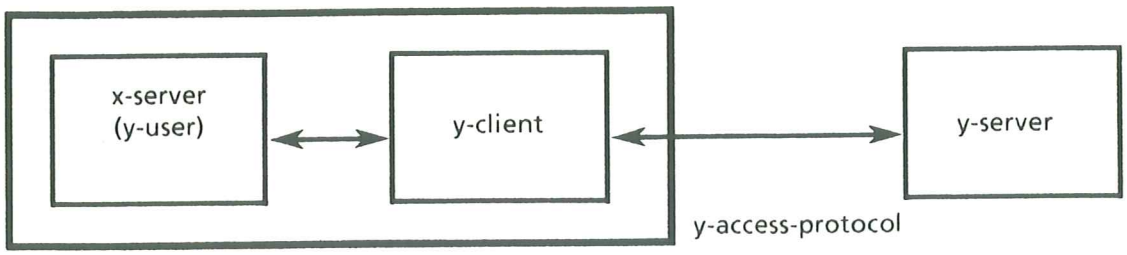


Figure 7 - A Server as a User of Another Server

This model can also be used when the two servers are of the same type,i.e., when an x-server is using another x-server.

3.3.3.4 Third-party-transfer

For some data object types (see clause 3.2.4) an x-server acts as a source and a y-server acts as a sink of data values (e.g. a Mailbox server may be a source and a Printing server may be a sink of printable documents such as Interpersonal Message Body Parts). In general, a server may act as both a source or sink of data (e.g. Document Filing and Retrieval Server, Mailbox Server).

If an x-client and a y-client are co-located within one application-process and an object value has to be transferred from an x-server to a y-server, it may be inefficient to transfer this object value via the x-access-protocol from the x-server to the x-client, and then via the y-access-protocol from the y-client to the y-server (see Figure 8). In this case it is more efficient to transfer only a reference to the object value in the access protocols. The referenced object value itself is transferred directly from the source x-server to the sink y-server.

This transfer is called "third-party-transfer" because there are three entities involved. In the case of a two party transfer the object value is transferred within the x-access-protocol between the x-client and the x-server.

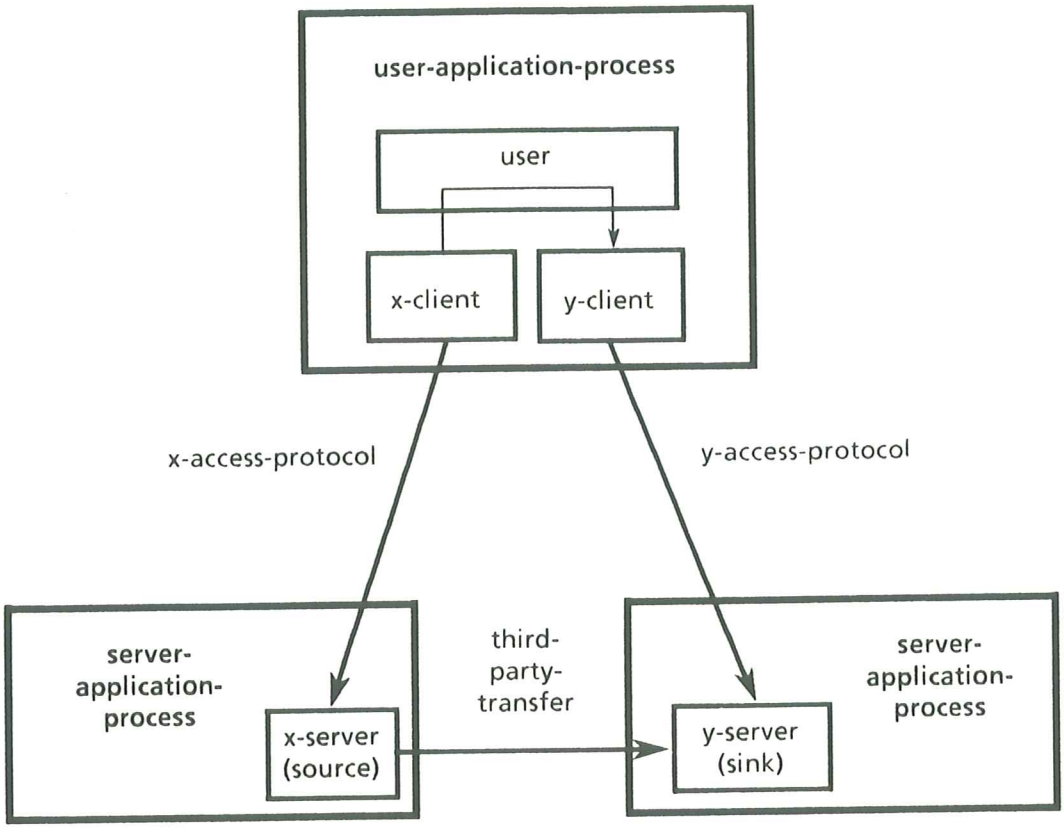


Figure 8 - Third-party-transfer

3.4 Client-Server Communication Model

This clause specifies some additional details about the communication between an x-client and an x-server, on the basis of the model introduced in clause 3.2.3 and Figure 3.

Figures 9 to 12 show various examples of configurations that require different sets of application-service-elements. The Association Control Service Element (ACSE) is required in every set of application service elements. Furthermore, the Remote Operations Service Element (ROSE) and/or the Reliable Transfer Service Element (RTSE) may be required in every set. Which additional application service elements are required in a given set depends on:

- the nature of the distributed-office-application concerned;
- whether the set is associated with a client or a server;
- whether the set implements the access protocol or the system protocol.

The data-object-format-specification represents a basis of cooperation between the user and the x-server, or between users.

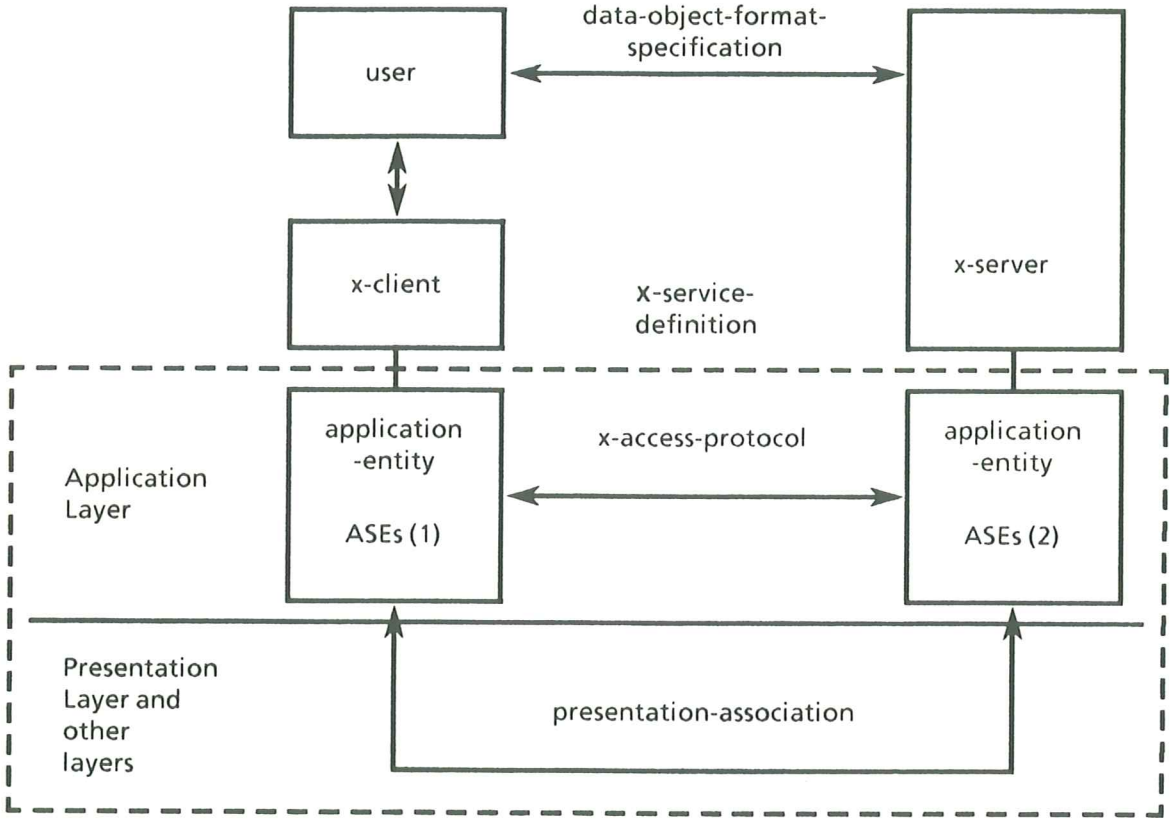


Figure 9 - OSI Communication between an X-client and an X-server

Legend for Figure 9:

(1) This set of application-service-elements implements the functions required by an x-client for communication with an x-server, using the x-access-protocol.

(2) This set of application-service-elements implements the functions required by an x-server for communication with an x-client, using the x-access-protocol.

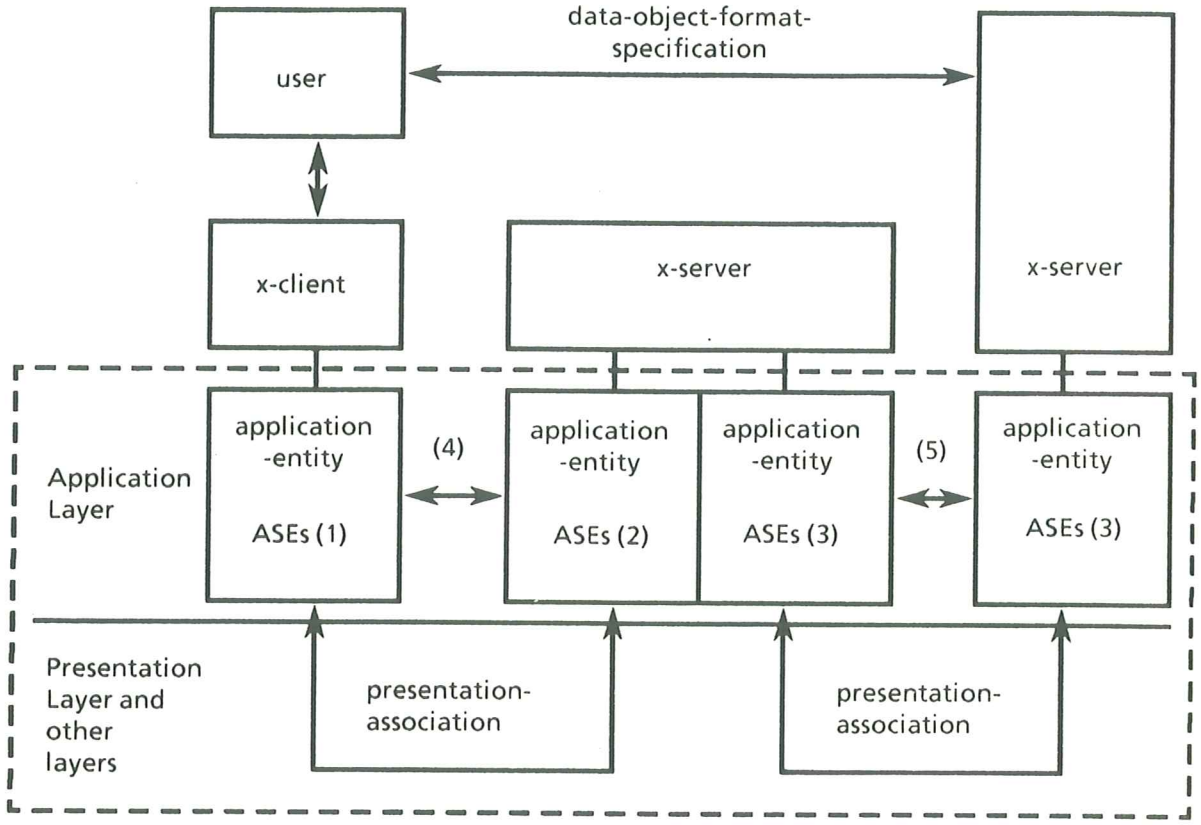


Figure 10 - OSI Communication between an X-client an X-server and between two X-servers

Legend for Figure 10:

(1) This set of application-service-elements implements the functions required by an x-client for communication with an x-server, using the x-access-protocol.

(2) This set of application-service-elements implements the functions required by an x-server for communication with an x-client, using the x-access-protocol.

(3) This set of application-service-elements implements the functions required by an x-server for communication with another x-server using the x-system-protocol.

(4) This is the x-access-protocol.

(5) This is the x-system-protocol.

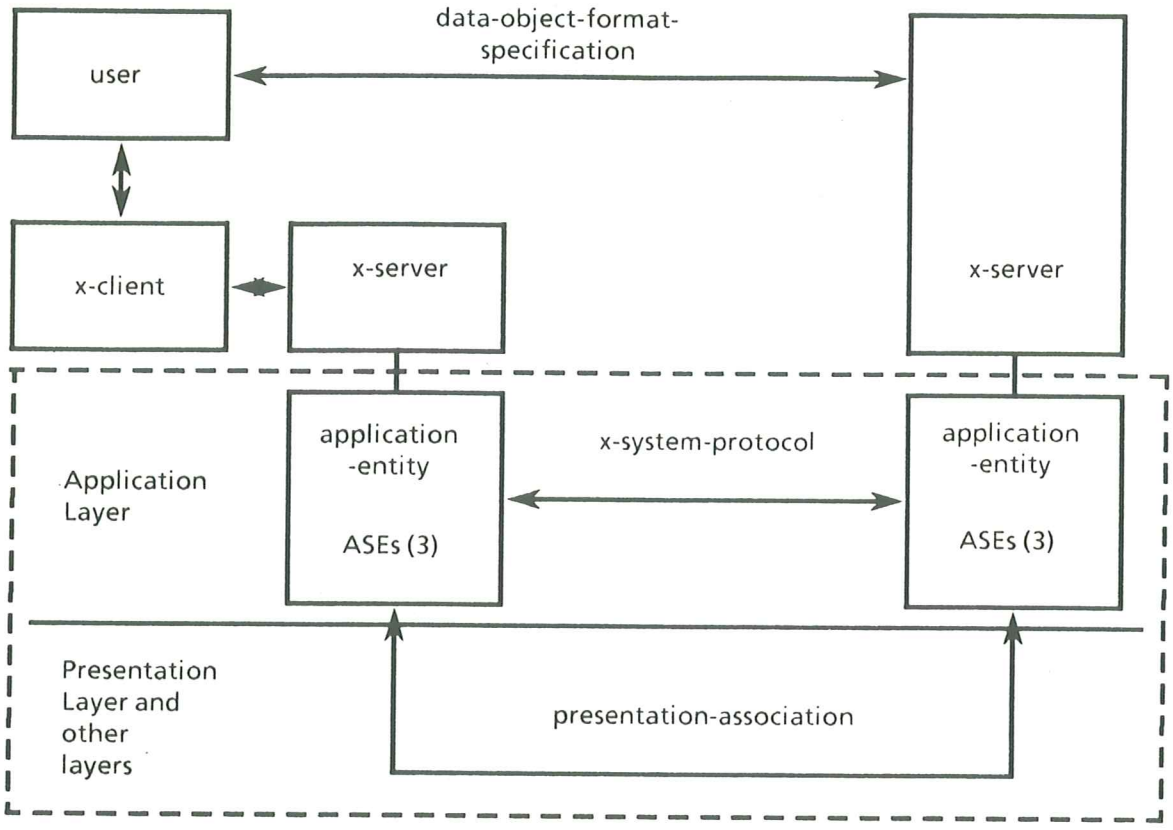


Figure 11 - OSI Communication between an X-client with a co-located X-server and another X-server

Legend for Figure 11:

(3) This set of application service elements implements the functions required by an x-server for communication with another x-server using the x-system-protocol.

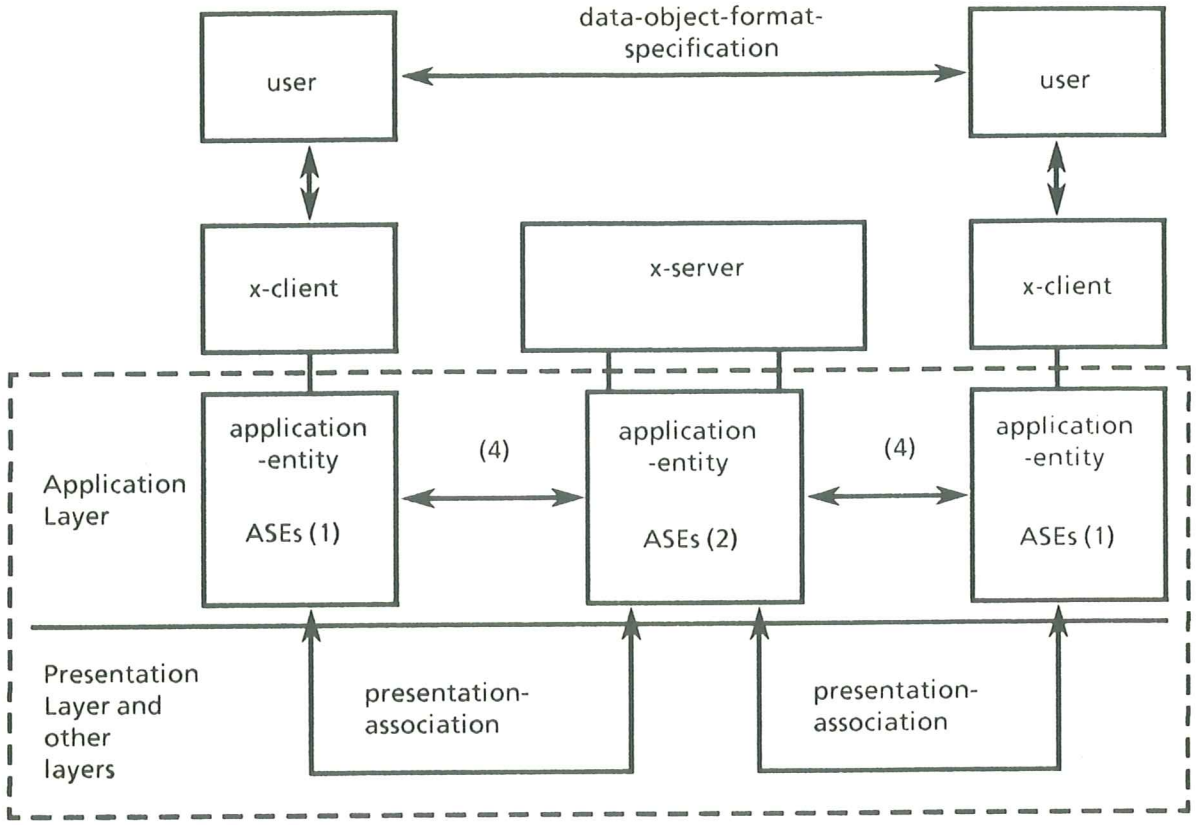


Figure 12 - OSI Communication between two X-clients and an X-server and between two Users

Legend for Figure 12:

(1) This set of application-service-elements implements the functions required by an x-client for communication with an x-server, using the x-access-protocol.

(2) This set of application-service-elements implements the functions required by an x-server for communication with an x-client, using the x-access-protocol.

(4) This is the x-access-protocol.

In Figure 13 the x-server is used as a store-and-forward system between the two users (e.g. in Document Filing and Retrieval or Message Transfer). In this case the data-object-format-specification represents the basis for cooperation between two users.

3.5 Functional Categories

3.5.1 Productive and Supportive Applications and Facilities

A distinction is made between supportive applications and productive applications and between supportive roles and productive roles of an application.

It is worth noting that the categorization of applications as "productive" or "supportive" is somewhat imprecise, though nonetheless useful. For example, the basic Message Transfer application could potentially be used as a supportive application for other applications, such as for distributing directory updates between directory servers. Likewise, although the Directory application is generally viewed as a supportive application, it could also be considered as a productive application when used for responding to human user queries for information. Rather than an intrinsic property of the application, the distinction is based on whether or not an application directly provides facilities of interest to human users.

Applications with supportive roles collaborate in order to provide users with a stable, "high-level" environment. Just as a programming environment consists of a number of programs (many of which are general utilities), the network operating environment for the productive distributed-applications consists of several supportive applications. These supportive applications, which are built using the same framework concepts as the productive applications, constitute the general operating support that can be assumed by the productive applications, and provide the users of the OSI distributed-office-applications with a "high-level" view of their environment, such as allowing them to be de-coupled from the location and physical addressing of various devices and resources.

In other words, these supportive applications constitute the high level support environment for the productive applications and for the users of these applications.

The productive applications are visible to the human user, are seen as useful, and are specifically used by him. For the general office worker the productive applications include remote printing, document filing and retrieval, mail, some uses of directory etc.

3.5.2 Operating Support

The general operating support that can be assumed by the productive applications includes, for example:

- time base;

- authentication and attribute facility;

- some directory functions (e.g. name to address mapping).

This list is not exclusive.

3.5.3 Management

Some management functions are particularly important, e.g. those functions recording operational behavior of the distributed-office-applications environment.

Other management functions are seen as distinct applications, and are seen by the end-user as productive applications. This is particularly so for the analysis and presentation of management information.

Further study of the management proposals is needed.

3.5.4 Guidelines for Applications

Some supportive applications, for example, authentication, have a major impact on other access protocols. This is both in terms of the information carried by the protocols and in terms of the sequence in which operations are performed. The framework will specify in due course, for instance, the permitted sequences required for authenticating, requesting and gaining access to a server. The area of authentication is for further study.

Other functionality, for example logging and accounting, have an impact on application design and specification. The framework will state guidelines, for example, on what is logged, when it is logged and how it is logged. The areas of security logging and accounting are for further study.

Administrators of a distributed office application system will choose policies for some of these aspects; so productive applications will need to be adaptable to changes in these policies.

Clause 8 of this Report gives an initial set of guidelines, which can be used at present, until certain aspects e.g. the security aspects, have been studied more in depth.

3.6 Types of Interactions Between Applications

This clause describes and classifies different types of interactions as they relate to the categories of supportive applications and productive applications introduced in clause 3.5.

These types of interactions will be used in later clauses of this report.

The interactions between a user and a productive application is described in clause 3 and not repeated here.

The interactions between a user and a supportive application may occur in relation to an interaction between the user and a productive application. This is shown in Figure 13 as a Type 1 interaction between the user and the (supportive) y-application. Information obtained in this Type 1 interaction is used by the user in the interaction with the (productive) y-application. The interactions uses the x- and the y-access-protocols respectively.

The interactions between two servers (either of which may be productive or supportive) is shown as a Type 2 interaction in Figure 13. The x-server uses the co-located y-client to access the y-server using the y-access-protocol.

Finally a set of coordinated interactions exists that is known as a third-party-transfer. Here the user or an entity acting as x-user and y-user using the x- and y-access-protocols, instructs the x-server and y-server to perform an information transfer.

This information transfer - the third-party-transfer - is subsequently carried out by the servers using the (application independent) third-party-transfer protocol. This protocol is executed by the third-party-transfer application-entities which are part of the server-application-processes.

This is shown as a Type 3 interaction in Figure 13. The Type 3 interactions thus contain two coordinated actions. Firstly, the action, internal to the user, to coordinate the set-up for the third-party-transfer with the x-server and the y-server and, secondly the third-party-transfer itself.

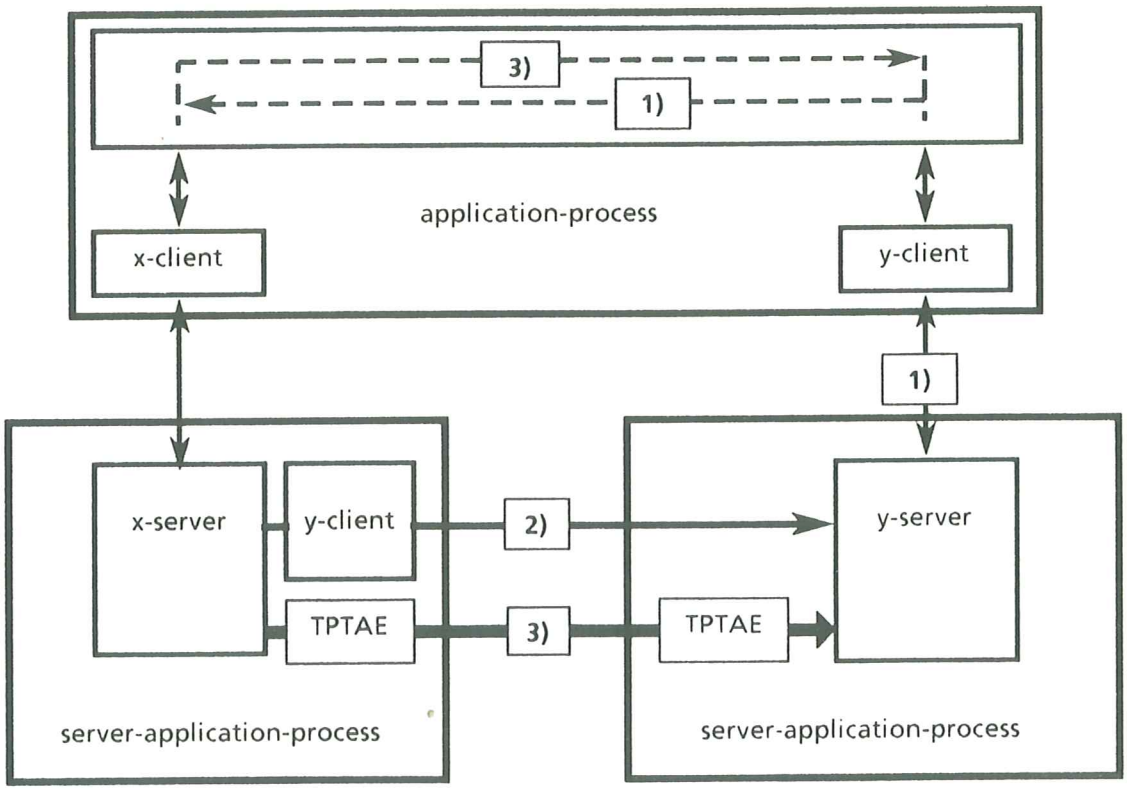


Figure 13 - Types of Interaction

Figure 13 Types of interactions:

1) Type 1 interactions between a user or a server acting in an x-user role and a y-server which results in information obtained from the y-server being used in interaction with the x-application.

2) Type 2 interactions between an x-server and a y-server using the y-client and the y-access-protocol.

3) Type 3 interaction between two servers acting upon instructions issued by the user or an entity acting in the role of x-user and y-user (using the x-access-protocol and the y-access-protocol). The information transfer from the x-server to the y-server uses the third-party-transfer.

3.7 Overview of Application Interactions

Figure 14 suggests possible interactions and their types according to the classification developed in the clause 3.6. The possibilities described are neither mandatory nor exhaustive; the text is tutorial.

The protocols (P1) and (P3) are already standardized protocols. The system-protocol P1 is the message transfer protocol (ECMA-93) used for communication between Message Transfer Agents of a Message Transfer System. P3 is the Submission and Delivery Protocol (CCITT Rec. X.411) enabling access to the Message Transfer System.

x-application	y-application						
	Directory	Authenti-cation	Time Base	Mailbox	Filing & Retrieval	Printing	Message Transfer
Directory	X	1, 2	1, 2				
Authenti-cation	1, 2	X	1, 2				
Time Base			X				
Mailbox	1, 2	1, 2	1, 2	X	3	3	2 (P3)
Filing & Retrieval	1, 2	1, 2	1, 2	3	2, 3	3	
Printing	1, 2	1, 2	1, 2			X	
Message Transfer	1, 2	1, 2	1, 2				(P1)

Figure 14 - Example of Application Interaction Matrix

4. NAMING CONSIDERATIONS FOR DISTRIBUTED-OFFICE-APPLICATIONS

4.1 General Requirements

The distributed-office-applications environment is characterized by the geographical and logical dispersion of entities. These entities, frequently referred to individually as applications, nodes, objects, clients and server, must all be made to work together in a coherent interconnection in order for a distributed-office-application to effectively carry out its tasks.

An important tool in developing this coherent interconnection is the utilization of the concept called "Naming". A "Name" is a linguistic concept that identifies a particular entity from among the set of all entities. An entity, e.g. a server, would therefore have its own name distinguishing it from other servers. This distinguished name would be found within the Directory application.

Some system entities which will have distinguished names are listed below.

- Human user;
- Group of human users;
- Node (user node and server node);
- Server;
- Object (e.g. a Mailbox or a File Cabinet).

In addition to identifying the separate entities, the naming concept assists in providing other functional capabilities necessary for distributed-office-applications. User nodes and server nodes need, for example, to be able to find the names of:

- a Filing and Retrieval server contains a specific filing cabinet;
- a server that contains a Mailbox for a specific user;
- a server that holds the master copy of data (e.g., that one who is allowed to update a part of the directory);
- Print servers in an organization that can offer Elite typefaces and wide carriages.

Users can accomplish these actions because the Directory offers the following functional capabilities:

- Name-to-attribute binding: This capability binds a name to a piece of information related to the entity (object) to which the name refers. Name-to-address binding is a special case of name-to-attribute binding. The large number of entities that may reside in a distributed-office-application system, make this capability essential. This function is analogous to a "white pages" directory.
- Attribute-to-set-of-names binding: This capability lists the name or names of entities (objects) which have a given attribute or attributes. One example of this is a "yellow pages" directory which can be used, for example, to find the names of all remote Print servers in an organization which have "Elite" typefaces and have wide carriages available to them.
- Alternative names: Different names for the same entity (object), i.e. aliases, are useful in allowing users more flexibility to access the numerous entities of the distributed-office-application environment. In order to provide different spellings of the same object, it may be convenient to introduce an alias (e.g. MUNICH / MÜNCHEN). Similarly, in an inter-enterprise environment, a server may be referenced externally by a name different to its internally used one, to hide any organizational details.

4.2 Name Types

ISO 7493 Addendum 3 sets forth some principles for naming entities within OSI. In particular, it defines two kind of names, primitive and descriptive, both of which are structured.

4.2.1 Primitive Names

A primitive name is an ordered sequence of "sub-names", each assigned by a naming authority which is denoted by the lexically previous sub-name. The distinguished names in the Directory application are, for all practical purposes, primitive names.

4.2.2 Descriptive Names

A descriptive name is an unordered set of assertions about an object's properties. Many names recognized by the Directory application are essentially descriptive in nature.

4.3 Guidelines for the Naming of Entities

The ECMA TR/32 allows an entity's name to consist of a number of Name Parts. There are no restrictions on which Name Part Types and the number of Name Parts which constitute a system element name. This is left to be determined by the applications using the Directory application. For details see ECMA TR/32 OSI Directory Access Service and Protocol.

The following restrictions apply to the Distinguished Name as defined in ECMA-TR/32:

- All entity names are built up using a common sequence of Name Part Types.

- A particular Name Part Type is only allowed to occur once in an entity name.

The exact Name Part Types to be used for distributed-office-applications is for further study.

Users of the Directory Application can use descriptive names when referring to an object in the directory information tree, i.e. the object is referenced by making assertions about it Name Parts and Properties. Nevertheless, it is advisable to order the Name parts partially or fully in order to guide the Directory server in locating the object's directory entry.

A user request to the Directory server may consist of a Name Part list that purports to be a name. The process by which these Name Parts are verified to be a name, is the name verification process. The name verification process determines, within its capability, whether the purported name is a valid name, or ambiguous, or erroneous.

It should be reiterated that all object names and Name Parts used in application bindings must be stored in the Directory.

4.4 Registration of Identifiers

A number of objects will have to be given standardized identifiers, either as part of the respective standards or by some registration authority. A few examples are given below. Identifiers are realized in the protocols in the form of OBJECT IDENTIFIERs as defined in ISO DIS 8824.2 ASN.1.

4.4.1 Protocol Types

The Remote Operation Service Element (ECMA TR/31) requires identification of Application Protocols.

4.4.2 Message Content Types

The message content types identify the different types of content that can be carried by the P1 protocol in the Message Interchange Application. Content type is an example of the data-object-format-specification (clause 3.4). For details see ECMA-93 MIDA.

4.4.3 Message Body Part Types

Message body part types identify the different types of format/encoding which can be found as part of a User Message (see clause 3.4 and ECMA-93 MIDA).These message body part types are also data-object-format-specifications.

4.4.4 Third-party-transfer Object Types

There are no third-party-transfer object types defined yet, but the need for such types has been identified.

4.4.5 Attribute (or Property) Types

There are no directory attribute (or property) types defined yet, but the need for such types has been identified.

Attribute (or property) types will be defined for any distributed-office-application, using the concept of "Attributes" (see clause 8.2.6). If an attribute type identifier has been allocated for a particular type of information in one application, this type can (and should) be used by other applications needing the same attribute type.

4.4.6 Interpretation Types

Interpretation types will be defined for any distributed-office-application, using the concept of "Interpretations" (see clause 8.2.6). If an interpretation type identifier has been allocated for a particular type of information in one application, this type can (and should) be used by other applications needing the same interpretation type.

5. SECURITY CONCEPTS

5.1 Introduction

This clause is tutorial in nature.

5.1.1 A Definition of "Security"

For the purposes of this document "security" will refer to characteristics of office systems that give resistance to accidents, failure and misuse, intentional or otherwise.Thus, security refers to a complex of procedural, logical and physical measures aimed at prevention, detection and correction of certain kinds of accident, failure and misuse together with tools to administer and manage these measures.

Given this definition, security does not only address intentional misuses, e.g. threats to a system, but it also addresses accidents such as the misrouting of a message and pinpointing the cause of the misrouting so that the responsible party can be identified.

In this view, security improves the integrity of doing business in addition to addressing threats to the organization itself.

5.1.2 Scope of Security

Many different security needs imply a common set of secure functions to be provided independently of office applications. These common, secure functions will become visible in the interactions between users and productive applications, between productive applications and supportive applications but also in the installation, maintenance and management of applications and of the underlying system. These functions, their interactions and their management constitute the scope of security in this Technical Report.

5.1.3 Security-policies

To be effective, security measures need to be coherent. Therefore, an organization will define its security measures and methods of administration and management of those measures in a security-policy. The responsibility for executing the security-policy and for maintaining its effectiveness is exercised by a security-administrator.

Below are examples of security measures to be addressed by a security-policy. Which measures to implement as part of a given security-policy depends on the environment of the organization:

- integrity of information contained in and/or processed by a system;
- confidentiality of (selected) information contained in and/or processed by a system;
- integrity of services and functions provided by a system;
- confidentiality of services and functions provided by a system.

- means to obtain third party guarantees for certain operations. In other words, verification of the integrity of processes and information by third parties is needed;
- means to authenticate individual users or groups of users according to defined rules;
- control of access to servers, functions and information available on or through a system;
- control of the flow of information within and between systems.

In the general case the organization will require interaction with other organizations. Organizations will select their own policies; each security-policy can be said to apply to a given security-domain which is under the control of a single security-administrator. Doing business requires, by implication, interaction between security-domains. This too needs to be addressed by a security-policy.

In some cases, two security-domains may interact directly, in other cases they may interact through a third party. Also, the degree of trust between security-domains may vary.

5.2 Security Requirements for Distributed-office-applications

5.2.1 General Security Requirements

This sub-clause introduces general security requirements as these occur in a distributed-office-application environment. These requirements reflect both implied requirements - for example: no system can be secure without some form of control of access - as well as specific requirements for security functions from a user point of view - for example: authentication of data origin.

5.2.1.1 Protection of Access

Access control provides the means to confine access to certain known users as well as to control access by these users to specific resources for specific operations. Thus, control of access has two major components: authentication of users and authorization of access by authenticated users. Access control will be exercised according to an access-control-policy which applies to a security-domain.

AUTHENTICATION

Users gaining access to a distributed-office-application system will be authenticated before being allowed access to any particular application that is subject to access-control-policy. Users may also require that the servers accessed are authentic. Users may access an x-server from a node belonging to the same security-domain as the x-server or from a node belonging to another security-domain. In either case, a commonly agreed procedure of exchanging authentication information must be used.

Authentication may be time bound; repeated authentication throughout an occasion of communication may be required by certain security-policies.

ACCESS AUTHORIZATION

Nodes in a distributed-office-application environment may require the use of access authorization to protect the confidentiality and integrity of security-objects and the integrity of the server node. Authorization methods may use a variety of mechanisms such as access-control-lists, capabilities and other security-attributes, singly or in combination.

Users will be granted access to x-servers and to security-objects within x-servers based on their privilege-attributes under the prevailing security-policy of the security-domain(s) involved.

Whenever users access servers or security-objects not belonging to their security-domain, authorization information as required by the serving security-domain, will be passed in a secure fashion.

5.2.1.2 Protection of Data Information

Security-policies may require data interchanged with or stored on a distributed office system to be protected from external attack. "External" in this context is used to indicate other than by the normal system access route, (e.g. line tapping, theft of media.)

Data protection covers both confidentiality (keeping secret) and integrity (protecting against change).

Within a distributed-office-application environment the following requirements with regard to data protection apply:

- Protection of data in storage (even on removable media);
- Protection in interchange, e.g. access control information, messages, electronic documents and files exchanged between systems.

Protection refers to preventing leakage of sensitive information as well as preventing contamination of trusted information with untrusted information.

Unless physical protection is to be depended upon, confidentiality requires the use of encryption, integrity requires the use of crypto-sealing.

Encryption techniques require the use of cryptographic keys. Systems supporting encryption must provide a secure method of managing keys both within a security-domain and between security-domains.

5.2.1.3 Protection of Usage of Resources

Security-policies may require protection of usage of resources. This protection takes two forms: keeping usage secret (confidentiality of usage) and preventing denial of service.

5.2.1.4 Accountability of Usage of Resources

Security-policies may require means to assure accountability of usage of resources. Accountability includes the selective logging of an audit trail of operations (both attempted and completed) as well as non-repudiation of data origin and of receipt.

Non-repudiation is proof, a posteriori, to a third party, of the identity of an entity that sent or received for example a given message. It is closely allied to data integrity and is usually combined with it.

5.2.2 Security Management Requirements

5.2.2.1 General Considerations

Systems supporting secure distributed-office-applications should provide the operating organization with the tools to manage the security-facilities of these systems. Examples of these tools are secure software installation and audit facilities for auditing the operation of the security-facilities.

Users of a distributed-office-application trust the integrity of system components to perform the expected functions and no other.

5.2.2.2 Aspects of Security Management

For every kind of security function defined for the distributed-office-application environment, four aspects of misuse or breach of security should be addressed that together define the management requirements of these functions. These aspects are: prevention, detection, recovery and administration. Depending on the level of security desired, some or all of these aspects become visible in actual implementations.

PREVENTION is based upon rules for managing a security function or application. An example of such a rule is changing passwords every 3 months.

DETECTION is based upon auditing of the security operations of a system.

Auditing of a security related operation provides security-administrators with feedback concerning the use and effectiveness of the security functions of the system.

Auditing has three components:

- audit trail generation and collection;

- audit trail analysis; and

- audit trail archiving.

The latter belongs to the aspect of administration.

In a distributed-office-applications environment an application may be distributed over multiple security-domains. Where such is the case, commonly agreed audit techniques may be needed to facilitate inter-domain cooperation.

RECOVERY of a security breach - real or suspected - may require changes in security procedures and information available at the different nodes of a distributed system. Therefore, protocols and procedures are needed to support the implementation of recovery measures.

ADMINISTRATION has two aspects related to the life of the system:

- gathering information from the system;

- creating information for the system.

The first aspect concerns reports made from information logged in the protected data base. Special filters must be provided so that a security-administrator can adapt the report to get only the kind of information he needs. The second aspect deals with the creation or the deletion of security-subjects and security-objects and with the definition of keys, rights and passwords (at least the initial password).

5.3 Secure Systems Model

5.3.1 Overview

In a secure distributed system, a number of activities must be engaged to provide that security.

The secure systems model divides these activities into elements, each one having a single, coherent role to play in the provision of the total security picture. These abstract elements are intended as reasoning tools rather than as real implementations of security functions. These elements are referred to as security-facilities.

Having identified the security-facilities and the communications between them, it can be shown how they might combine together to form supportive security applications or how they become trusted components of user-application-processes and server-application-processes, and standard protocols defined where appropriate for their interactions with each other and with their elements of the distributed-office-application environment.

In terms of the OSI model, the level of view addressed here is above the Application Layer. The supportive security applications described communicate using services of sufficient security to satisfy their needs. These needs take the form of guarantees, to some acceptable level, that communications between them and with their untrusted peers are confidential and unmodified, and that each communication is with a known and identified peer entity.

The model for the provision of these guarantees at lower OSI layers is a matter for further study.

There are two fundamentally different levels at which the security requirements of a distributed system need to be addressed:

- application independent level, to control access to distributed system security-objects as user-application-processes, server-application-processes, workstations, communication resources, etc.;

- application specific level, to control access to specific security-objects within an application (such as a document).

These two levels of view have quite different requirements reflected in different security-policy subsets tailored to the different kinds of protected security-objects involved and the different components that are responsible for their support. Something that is considered as a protected security-object at one level can become an accessing security-subject at another.

5.3.2 Security-facilities

At this stage of description, the reader should make no assumptions about the degree of distribution of the facilities; this might vary from being a single security server to being an aspect of every distributed supportive or productive application. Neither it is suggested that all of these facilities need be available on every node of a distributed system. They should be viewed as a list of building blocks from which a choice can be made appropriate to the security-policy and level of security required for the distributed system. However, by identifying the full list, the model causes omissions to be made evident and any resulting security weaknesses other than accidental. Clauses 5.3.3 and 6.2 show some of these security-facilities combined into three supportive security applications.

This clause identifies the following security-facilities:

5.3.2.1 User Sponsor Facility

The only entity in a distributed system that is aware (independently of any services that may be being used) of an individual security-subject's current access to protected security-objects. The security-subject involved will usually be a human user, but under policies where servers access to other servers is policed, the security-subject can be an x-server. Its responsibility include:

- passing credentials for authentication
- initiating of service selection
- timing out inactive users.

5.3.2.2 Authentication Facility

Accepts and checks security-subject credentials, communicating its conclusions to other security-facilities. The security-subject will either be a human user via his user sponsor, a non-security application acting as security-subject (i.e. an x-server using a y-server), or a non-security application coming on-line and making itself available.

5.3.2.3 Security-attribute Facility

Provides appropriate subject-related access privilege-attributes and access control-attributes, to be used to authorize or deny requested access by security subjects to security-objects.

5.3.2.4 Authorization Facility

Uses access-context, (security-subject) privilege-attributes and (security-object) control-attributes to authorize or deny requested access by security-subjects to security-objects.

5.3.2.5 Association Management Facility

This facility ensures:

- secure underlying communications, including assurance of the identity of the communicating entities.
- authorization, via an authorization facility, for the two entities to communicate on behalf of the controlling user.

How the upper layer architecture functions relate to or may be used to support the Association Management Facility is for further study.

5.3.2.6 Security State Facility

Maintains the current dynamic state of authenticated security-subjects and security-objects in the distributed system, their associations and the privilege-attributes carried by those associations.

5.3.2.7 Security Audit Facility

Receives event information from other security-facilities for recording and immediate or later analysis.

5.3.2.8 Security Recovery Facility

Acts upon event information from the Security Audit Facility according to a set of rules defined by a security-administrator.

5.3.2.9 Inter-domain Facility

Controls and maps one security-domain's interpretation of security-subject identity, security-object identity, authentication and authorization data into another security-domain's interpretation. Helps Association Management form associations between entities in different security-domains.

5.3.2.10 Cryptographic Support Facility

Provides cryptographic functions used both by other security-facilities and applications to secure data in storage and transit in the following specific ways:

- data confidentiality
- communications confidentiality
- communications integrity
- data origin authentication
- non-repudiation of origin
- non-repudiation of receipt.

5.3.3 Supportive Security Applications

For the purpose of this framework, the following three security applications are identified:

- Authentication and Security-attributes application. This combines the Authentication Facility and the Security-attribute Facility.
- Inter-domain application. This is the Inter-domain Facility.
- Security Audit application. This is the Security Audit Facility.

In addition other supportive security applications may be defined that implement other combinations of the security-facilities given in clause 5.3.2. Note that the presence of security-facilities, e.g. Association Management, whether as separate entities or as part of user-application-processes and server-application-processes will need additional protocol elements for distributed-office-applications.

5.3.4 Proxy

In some cases an x-server may be accessed by a y-server rather than directly by a user. There are two situations:

- the initiating x-server is acting on its own behalf; or
- the x-server is acting on behalf of of another security-subject (e.g. a human user).

The first situation may be used for example to restrict access to security-objects held on one server (say File server) to those coming via another (say Database server). It is entirely appropriate for the Database server to act with respect to the File server as a security-subject with its own identity and access privileges.

On the other hand, it might be appropriate for the initiating x-server to act on behalf of the user (by proxy) and assume some or all of his security-attributes. The proxy could either imply trust by the user for a single specific access request or it can imply wider powers. The proxy may either contain access request details or it may contain a reference to those details. This is for further study.

In this way access can be controlled in terms of the route used.

6. OPERATION OF SUPPORTIVE APPLICATIONS AND FACILITIES

6.1 Time Base Facility

The various components of any distributed system must be able to obtain the current time. This time could be used by other applications, e.g. to timestamp files, to timestamp messages, to enable authentication to be carried out.

With the current proliferation of international organizations, a world-wide means of providing a reliable and unambiguous time base is required. This will become even more important in the future, as large numbers of nodes are connected to world-wide networks which themselves are interconnected to other networks.

The methods by which various hosts obtain and maintain the correct time is outside the scope of this Report.

All protocols in the Distributed Office Application environment will express the time using the data type, "UTC Time", as defined in ISO DIS 8824 and CCITT Rec. X.409. For a formal definition, see clause 8.2.4.

This facility provides for a generalized international time containing day, month and year in the Christian era (with optional seconds), which can be specified to a precision of one second or one minute.

Synchronization does not have to be exact but should maintain time to be within a reasonable spread (e.g. something of the order of 10 minutes) over the whole of the distributed system. The accuracy is set by the administrators.

More precise timing if required, (e.g for timestamps) is by use of local time facilities within a node. Values of local time may need to be qualified by location information indicating the source of the time value.

6.2 Supportive Security Applications

Note 1:

This clause and its Sub-clauses are provided for explanatory purposes only and should not be understood as definitive for the specification of Supportive Security Applications. This subject is to be treated more fully in the Framework for Security in Open Systems, now under preparation in ECMA TC32-TG9.

This clause describes the supportive security applications which together provides support for:

- application-independent level security to control access to the distributed system security-objects such as clients, servers, workstations, communication resources;
- application-specific level security to control access to specific security-objects within an application (such as a document).

The following supportive security applications are identified and described in subsequent clauses:

- Authentication and Security-attributes application (clause 6.2.1)
- Inter-domain application (clause 6.2.2)
- Security Audit application (clause 6.2.3).

These security applications are not responsible for:

- the establishment of secure communication between a security application client and a security application server (this could be achieved by the installation of cryptographic master keys as part of the system establishment);

- the authentication of nodes as genuine and authorized members of the distributed system.

These matters are the responsibility of Security Management during installation, configuration and reconfiguration; management mechanisms are for further study.

It should be noted that these security applications are independent of the office applications they support and service.

6.2.1 Authentication and Security-attributes Application

This application provides for the authentication and the handling of the security-attributes of human user and of applications level security-objects e.g x-servers.

In general, authentication relies on a human user, user or x-server proving its identity by proving that it is in possession of some piece of information that is held (secret) by that human user, user or x-server. This takes three basic forms

a) The human user, user or x-server produces a copy of a piece of information that is kept secret by that human user, user or x-server that the system associates with that human user, user or x-server by virtue of some mapping table (the classical password approach). The checking of the validity of this information may be achieved by the use of another y-server (e.g. check simple credentials operations in the ISO Directory proposals).

b) The human user, user or x-server uses some piece of information without passing it explicitly such that the system (which also knows it) can prove to its satisfaction that the sender was in possession of the information. Again by internal mapping, the identity of the sender is proven. (Cryptographic techniques based on conventional key schemes).

c) The human user, user or x-server uses some piece of information without passing it explicitly such that the system (which does not know it) can prove to its satisfaction by the use of some shared piece of information that the sender was in possession of the information. By either internal mapping, or explicitly from the information passed, the identity of the sender is corroborated. (Cryptographic techniques based on public key schemes).

Various protocols for exchanging authentication may be used e.g a challenge/response protocol that offers resistance to replay of an authentication sequence.

In order to provide a flexible approach, users and human users are associated with particular security-attribute values. This application provides for the handling and storage of subject-related privilege-attributes and object-related control-attributes used to authorize or deny requested access to application level security-objects e.g. an x-server. Tests on these two sets of security-attribute values are carried out by the Authorization Facility; these tests may be complex and take external factors into account.

The result of authentication and the values of security-attributes are not valid indefinitely Under some security-policies, for example, human users will be required to re-authenticate themselves and re-establish security-attributes from time to time.

A user of the Authentication and Security-attributes application is, under most security-policies, logged in the Security Audit application.

6.2.2 Inter Domain Application

Human users and/or the components of a distributed system may be in different security-domains.This impacts the security aspects of interactions between them. For instance, security-objects freely accessible to human users in one security-domain may be accessible to only a few human users outside the security-domain.

It is important that certain "secret" security-objects can be protected from access by any user from other security-domains. However, it may be necessary to allow certain users from the other security-domains access to other "non-secret" security-objects. Even so, in cases of breaches in the security of remote security-domains, the "secret" security-objects in the local security-domain should be protected.

The choice of which security-objects should be inaccessible between security-domains may be different depending on the relationship between the security-domains. Nor is this choice purely hierarchical. Because of a working relationship between two security-domains a particular set of security-objects may be accessible to some users from one security-domain and inaccessible to all users from other security-domains whilst other security-objects inaccessible from this security-domain may be accessible to some users from other security-domains. In other words one security-domain may "trust" another for accesses to a restricted set of security-objects. This set will depend on the relationships between the security-domains of the security-subject's and security-object's.

Given trust for access to a set of security-objects between security-domains, a further grain of control may be necessary within the security-object's security-domain based on the identity and attributes of the security-subject. The security-object's security-domain will have to trust the security-subject's security-domain to the security-subject's identity and may also make use of security-subject attributes supplied by the security-subject's security-domain.

Relationships between security-domains will be bi-lateral (and reflect underlying business arrangements). Therefore, there will be a logically distinct Inter Domain application for any two interacting security-domains.

The Inter Domain application maps one security-domain's interpretation of:

- security-subject identity (of human users and users);

- security-object identity; and

- security attributes

into another security-domain's interpretation.

The Inter Domain application interacts with the Authentication and Security-attribute application to achieve this. Under any security-policy, the Inter Domain application logs activity with the Security Audit application.

6.2.3 Security Audit Application

An audit trail is an essential requirement for any effective monitoring of security.

The security-policy specifies the events or occurrences that must be recorded by productive applications and by supportive applications. They record the events or occurrences by using the Security Audit application.

The security audit trail is held securely for subsequent analysis by human users that have the appropriate security-attributes. (Some events or occurrences may cause immediate reports and alarms to be made to nominated human users or to nominated components of a distributed system.)

6.3 Directory Application

Distributed-office-applications shall use the Directory application specified in ECMA TR/32: "OSI Directory Service Access and Protocol".

The Directory application plays a key role in providing users with a stable "high level" view of the distributed-office-applications environment in spite of the inherently changeable network and physical facilities at the lower layers.

The most basic function of the Directory application is the resolution of names to presentation addresses. A user references an entity on the network by name, and the application, via the

Directory server, resolves this to a presentation address that usually bears a close relationship to "physical location". This also allows a user friendly interface to the system, since entities can be referred to by an easily remembered name.

The Directory application also provides a means of managing groups (lists) of entities (i.e., adding and deleting members, recursive expansion and membership verification). This has many applications, for example in electronic mail.

In addition, the Directory application provides limited but generally useful facilities for locating entities by referring to some properties of the entities (e.g. finding, within a specified domain, the set of entities that are x-servers of a certain x-application; or binding to the first available instance of an x-server).

6.4 Third-party-transfer Facility

Several office applications will act as a source or sink of objects whose values are of comparatively large quantities, for example files, documents or P2 body parts.

The transfer of a large data object value conceptually involves three parties: a user which requests and orchestrates the transfer, a source which produces the data object value, and a sink which consumes the data object value.

In a two party transfer, the user is either the source or the sink and the second party the sink or source respectively. In some cases the user may be an intermediate sink and afterwards the intermediate source of the same data object value, i.e the data object value is transferred from the source to the user and afterwards from the user to the sink. In this case it is more efficient if the user orchestrates a (direct) third-party-transfer from the source to the sink.

In analogy to a "high level programming language"the remote operations of the sink and source services use the "argument or result by value" technique in the case of the two party transfer and the "argument or result by reference" technique in the case of third-party-transfer.

6.4.1 Characteristics of the Third-party-transfer Facility

The third-party-transfer facility has the following characteristics:

- provide a single mechanism to perform the third-party-transfer between an open ended list of servers;
- the sink should not need to know the server type of the source server and vice versa;
- the object transferred is "data" and the user is responsible that the data type produced by the source server matches the data type expected by the sink server;
- the third-party-transfer should not constrain the specification of the productive applications;
- the third-party-transfer should consider the security requirements.

More details about the third-party-transfer facility is given in clause 8.2.7 and below.

6.4.2 Third-party-transfer vs. Other Types of Interactions between Servers

The third-party-transfer facility does not replace all other types of interaction between servers. In particular, when x-servers of the same x-application need to have some interactions other than a simple data transfer, a specific x-system-protocol may be defined for the given application.

When servers of different applications need to have some interactions other than a simple data transfer they use access-protocols, as described in clauses 3.3.3.2, 3.6 and 3.7.

6.4.3 Security of Third-party-transfer

Given a proxy mechanism that allows security-subjects (human user or authorized programs) to pass their authorization, or part thereof for a given set of operations, to other security-subjects such as productive or supportive applications any interaction between

multiple applications on behalf of an authorized security-subject can be securely implemented (see also clause 5.3.4).

The example below describes a secure third-party-transfer operation and uses the concept of a Service Request to denote a request for an operation from the user to the applications involved.

The proxy mechanism used here is a Service Request sealed by the Authentication and Security-attribute application defined in clause 6.2.1. The sealed request contains (references to) the identities of the server involved.

Assumptions:

- A request for third-party-transfer is considered to have two components. an Initiator Request Component and a Responder Request Component. Both components are sealed cryptographically.
- The description given here assumes that the user generates two request components and arranges their transmission.
- It is assumed that trusted paths exists in the sense that if an entity receives a Request or Token it can be assured of the identity of the sender.
- The Initiator (sink) is considered to act on behalf of the user.

Process Description:

1) The user creates a Service Request for the transfer of a file to the printer for printing.
2) The user sends the Responder Request Component to the Responder (source) server. The latter checks it and if acceptable returns an appropriate indication - a Token - to the user. A copy of the responder Request Component and of the Token is stored in Security State for further reference.
3) The user sends the Initiator Request Component and the Token to the Initiator (sink) server. The latter may return an indication that the operation can be carried out.
4) When the initiator (sink) is ready to start the operation it sends the copy of the Token to the Responder (source).
5) The Responder (source) can validate the received Token and allow the requested operation to proceed. Upon completion of the operation the copy of the Token maintained by Security State is updated e.g. by the Responder. In case the proxy was valid only for one operation the Token is inactivated; otherwise it may remain active for other operations.

The above description postulates interactions between the user and the Authentication and Security-attribute application as well as between the former and Security State. The latter is needed for proxies that cover multiple operations.

This description is only an example that serves to illustrate interactions required to secure such operations. Other are possible as well as a more general mapping to multiple application interactions. This subject requires further study.

6.5 Interactions between Supportive Applications

Every supportive application uses the Time Base. These interactions are depicted by dotted arrows in Figure 15. The means by which time is obtained are outside the scope of the Framework. When Time is available locally, it is accessed by local means; the only requirement is one of synchronization with the Time Base, as explained in clause 6.1.

The Directory application uses the Authentication and Security-attributes application in order for the Directory to authenticate its users. This is a type 2 interaction.

If the Authentication and Security-attributes application is distributed among a number of servers, then any of its servers may interact as a user with the Directory application to find the presentation address of another server of this application (type 2 interaction).

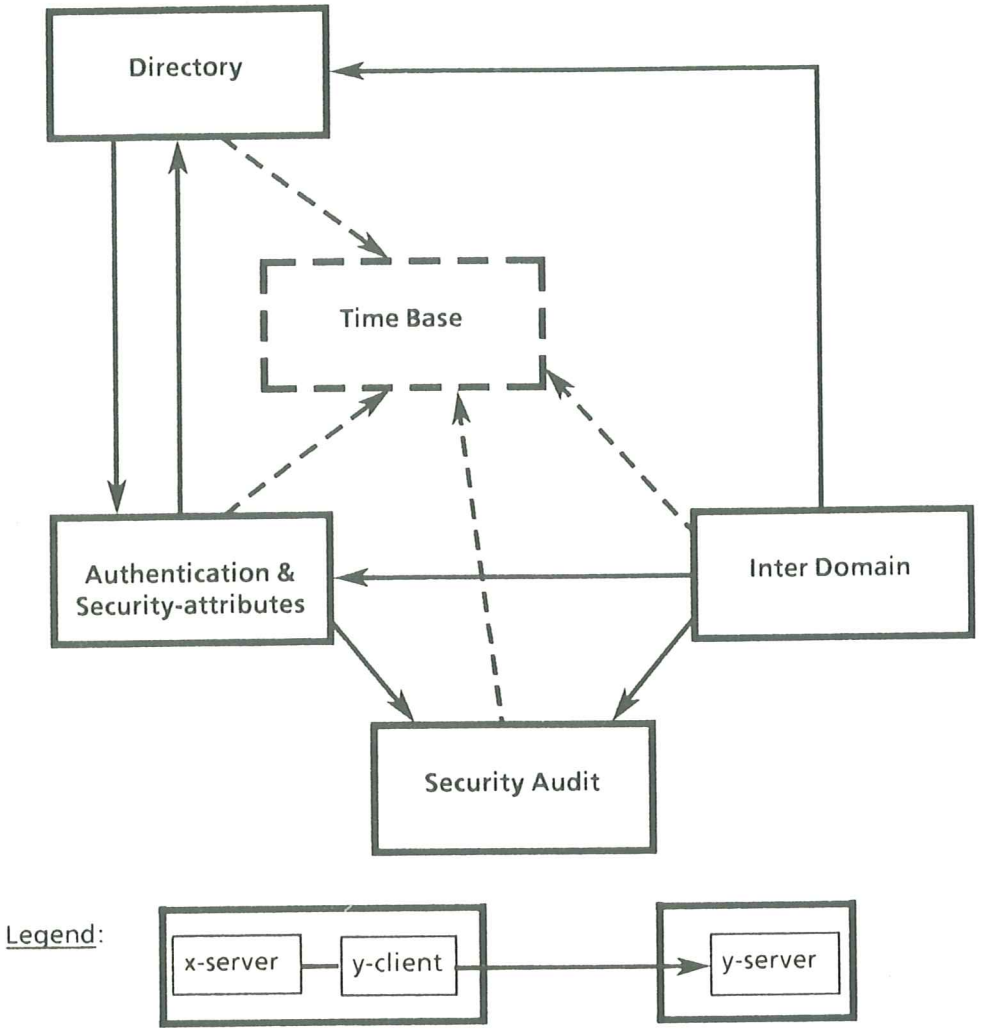


Figure 15 - Interactions between the Supportive Applications

6.6 Interactions between Users and Supportive Applications

There are interactions between the user of a certain supportive application and another supportive application; i.e. the user of the Directory application may access the Time Base and Authentication and Security-attributes application, the user of the Authentication and Security-attributes application may access the Time Base and the Directory application . An example is shown in figure 16. All the interactions depicted by solid arrows in the figure are of type 1 as defined in clause 3.6. The dotted arrows represent steps of obtaining time; the means by which the client does it, are outside the scope of this framework. However, the interaction with Time Base are emphasized as the clients require a common notion of time (Time Base) in order to interact with the distributed-office-applications.

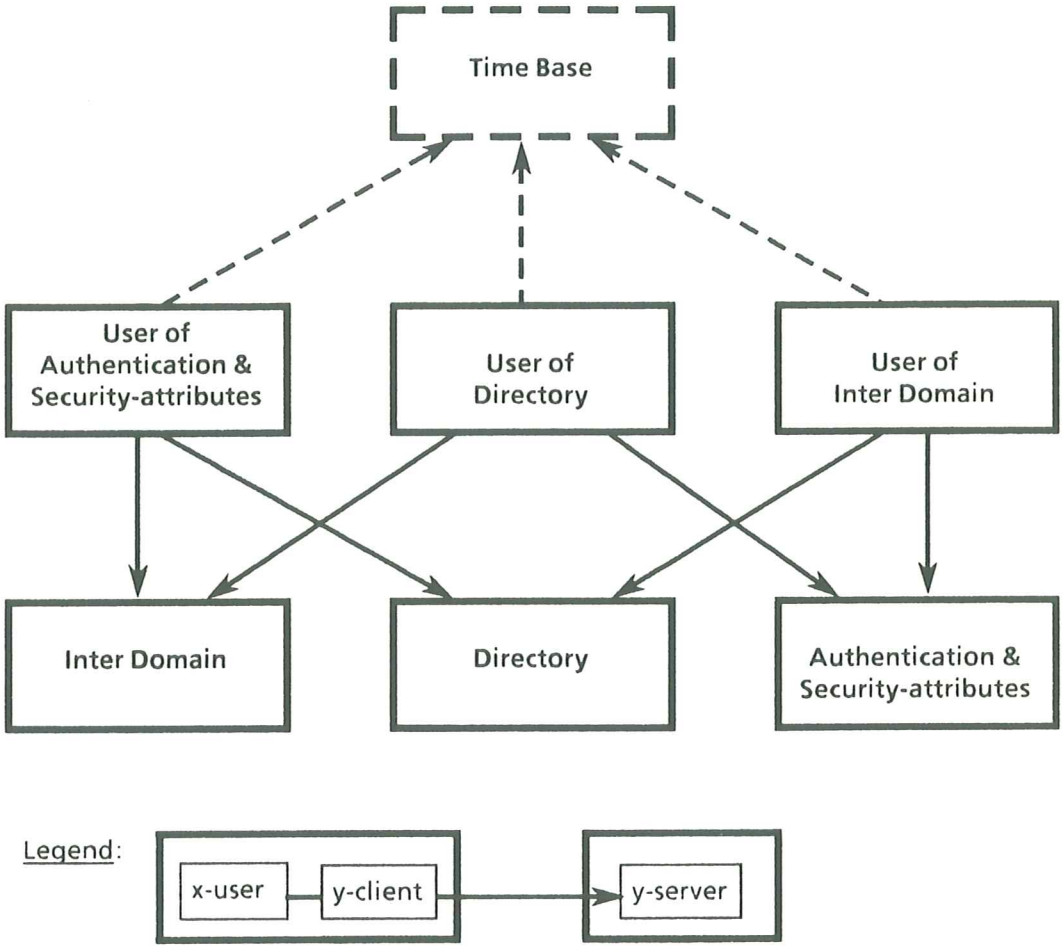


Figure 16 - Example of Interactions between a User of a Supportive Application and Another Supportive Application

7. SUPPORT FOR THE PRODUCTIVE APPLICATIONS

7.1 Introduction

This clause describes in a tutorial manner how the productive applications use the supportive applications. The examples include Message Transfer, Mailbox, Document Filing and Retrieval, and the Printing application.

The supportive interactions described herein are generally invisible to the human user of these productive applications, i.e. there is no requirement for human users to have to explicitly specify these interactions.

The interaction types mentioned in the following subsections, are those defined in clause 3.6.

7.2 Support for Message Transfer Application

7.2.1 Description of the Message Transfer Application

The Message Transfer application enables its users to send and receive messages of various lengths and content. The Message Transfer application is directly accessible to all users of a distributed-office-application system, allowing interchange of messages between users separated by large or small physical distances. The transfer of messages from sender to recipient is done in a store and forward manner.

An example of such a Message Transfer application is defined in Standard ECMA 93 - Distributed Application for Message Interchange.

A Message Transfer application may provide specific facilities, such as the support of distribution lists to group a number of recipients together under one name.

7.2.2 Operation of the Message Transfer Application

The Message Transfer application requires support from the Time Base and from the Authentication and Security-attribute application and Directory application.

The following interactions may be involved in a typical use of the Message Transfer application:

1) The Message Transfer application user accesses the Time Base.

2) The Message Transfer application user accesses the Directory application to obtain the presentation address of a Message Transfer server (interaction type 1).

3) The Message Transfer application user accesses the Authentication and Security-attribute application to obtain security-attributes for accessing the Message Transfer server (interaction type 1).

4) The Message Transfer application user submits a message to the Message Transfer server.

5) The Message Transfer server accesses the Authentication and Security-attribute application to authenticate the Message Transfer application user (interaction type 2).

6) The Message Transfer server accesses the Time Base to generate a time stamp for each message.

8) The Message Transfer server accesses the Directory application to expand any "recipients" which are on distribution lists (interaction type 2).

9) The Message Transfer server accesses the Directory application to obtain the presentation address of each recipient's Mailbox server (interaction type 2).

7.3 Support for Mailbox Application

7.3.1 Description of the Mailbox Application

The Mailbox application is closely tied to the Message Transfer application. The Message Transfer application actually delivers messages to a "mailbox", which is associated with a user, and the Mailbox application allows this user to obtain mail. An example of a Mailbox application is the Standard ECMA- .

7.3.2 Operation of the Mailbox Application

The Mailbox application requires support from the Time Base and from the Authentication and Security-attribute and Directory applications.

The following interactions may be involved in a typical use of the Mailbox application:

1) The Mailbox application user accesses the Time Base.

2) The Mailbox application user accesses the Directory application to obtain the presentation address of the Mailbox server (interaction type 1).

3) The Mailbox application user accesses the Authentication and Security-attribute application to obtain security attributes for accessing the Mailbox server (interaction type 1).

4) The Mailbox application user requests mail messages from the Mailbox application.

5) The Mailbox server might access the Authentication and Security-attribute application to authenticate the Mailbox application user (interaction type 2).

6) The Mailbox server uses an Authorization Facility to check if access is permitted.

7) The Mailbox application returns messages to the user.

7.4 Support for Document Filing and Retrieval Application

7.4.1 Description of the Document Filing and Retrieval Application

The Document Filing and Retrieval application provides the capability for large capacity document storage filing and retrieval to multiple users in a distributed system.

The Document Filing and Retrieval application also provides control of access for the documents held.

7.4.2 Operation of the Document Filing and Retrieval Application

Within a security-domain the Document Filing and Retrieval application requires support from the Time Base and from the Authentication and Security-attribute and Directory applications.

The following interactions involved in a use of the Document Filing and Retrieval application (in this example retrieval) are:

1) The Document Filing and Retrieval application user accesses the Time Base.

2) The Document Filing and Retrieval application user accesses the Directory application to obtain the presentation address of the Document Filing and Retrieval server it requires (interaction type 1).

3) The Document Filing and Retrieval application user accesses the Authentication and Security-attribute application to obtain security-attributes for for accessing the Document Filing and Retrieval server (interaction type 1).

4) The Document Filing and Retrieval application user requests the document from the Document Filing and Retrieval application.

6) The Document Filing and Retrieval server might access the Authentication and Security-attribute application to authenticate the Document Filing and Retrieval application user (interaction type 2).

7) The Document Filing and Retrieval application returns the document to the Document Filing and Retrieval application user.

7.5 Support for Print Application

7.5.1 Description of the Print Application

The Print application provides the capability for multiple users in a distributed system to share expensive, high facility imaging equipment.

7.5.2 Operation of the Print Application

Within a security-domain the Print application requires support from the Time Base and from the Authentication and Security-attribute and Directory applications.

The following interactions may be involved in a use of the Print application (in this example printing a document):

1) The Print application user accesses the Time Base.

2) The Print application user accesses the Directory application to obtain the presentation address of the Print server it requires (interaction type 1).

3) The Print application user accesses the Authentication and Security-attribute application to obtain security-attributes for using the Print server (interaction type 1).

4) The Print server accesses the Authentication and Security-attribute application to authenticate the Print application user (interaction type 2).

6) The Print application user sends the document to the Print server.

7) The Print server queues the document for printing.

8) The Print application notifies the Printing application user of the completion of the request.

8. GUIDELINES FOR THE DESIGN OF PROTOCOLS

Note 2:

The content of clauses 8.1 and 8.2 are based on agreed and stable ECMA documents. It is expected that current work in ISO and CCITT will replace ECMA TR/31 and ECMA/TR 32 as reference, once that work is stabilized.

All protocols for both productive and supportive applications shall conform to the Remote Operations as specified in ECMA TR/31. In more detail the access protocols shall use the notation and concepts of that report and shall allow any mappings defined in clause 6 of that Report. The following clauses give a short introduction to these concepts and notation in the context of an access protocol taking into consideration the application rules of clause 8.2.

8.1 Concepts

8.1.1 Object Model

Both clients and server are considered to contain objects.

All the externally visible behaviour of an object is described in an object-type specification. Objects described by the same object-type specification are said to be instances of that object type. The object-type specification is a set of type-operations, each of which is distinct and logically complete. Each type-operation is specified at a level of abstraction that defines what happens, not how it happens.

The effect of a type-operation may depend on the state of the object; the state results from a subset of the preceding type-operations performed on it. Change of sate of the object and their effects on type-operations are described in the object type specification.

An x-server is modeled by a collection of one or more objects of a variety of object types. These objects are called x-server objects. The externally visible behaviour of the x-server is described by the object-type specifications of the x-server objects in that collection of objects. Similarly, the externally visible behaviour of the x-client may be described by a number of object-type specifications. These objects are called x-client objects. All interactions between a client and a server are described by the x-service-definition which is derived wholly from the object-type specifications of the server and the client.

An occasion of communication between a client invocation and a server invocation is established by a bind-operation. The bind-operation confines the series of interactions:

- to an application context which defines a subset of the type-operations available in the x-service-definition and sequencing rules stated in the x-service-definition;

- to a particular set of x-server object instances and optionally x-client object instances.

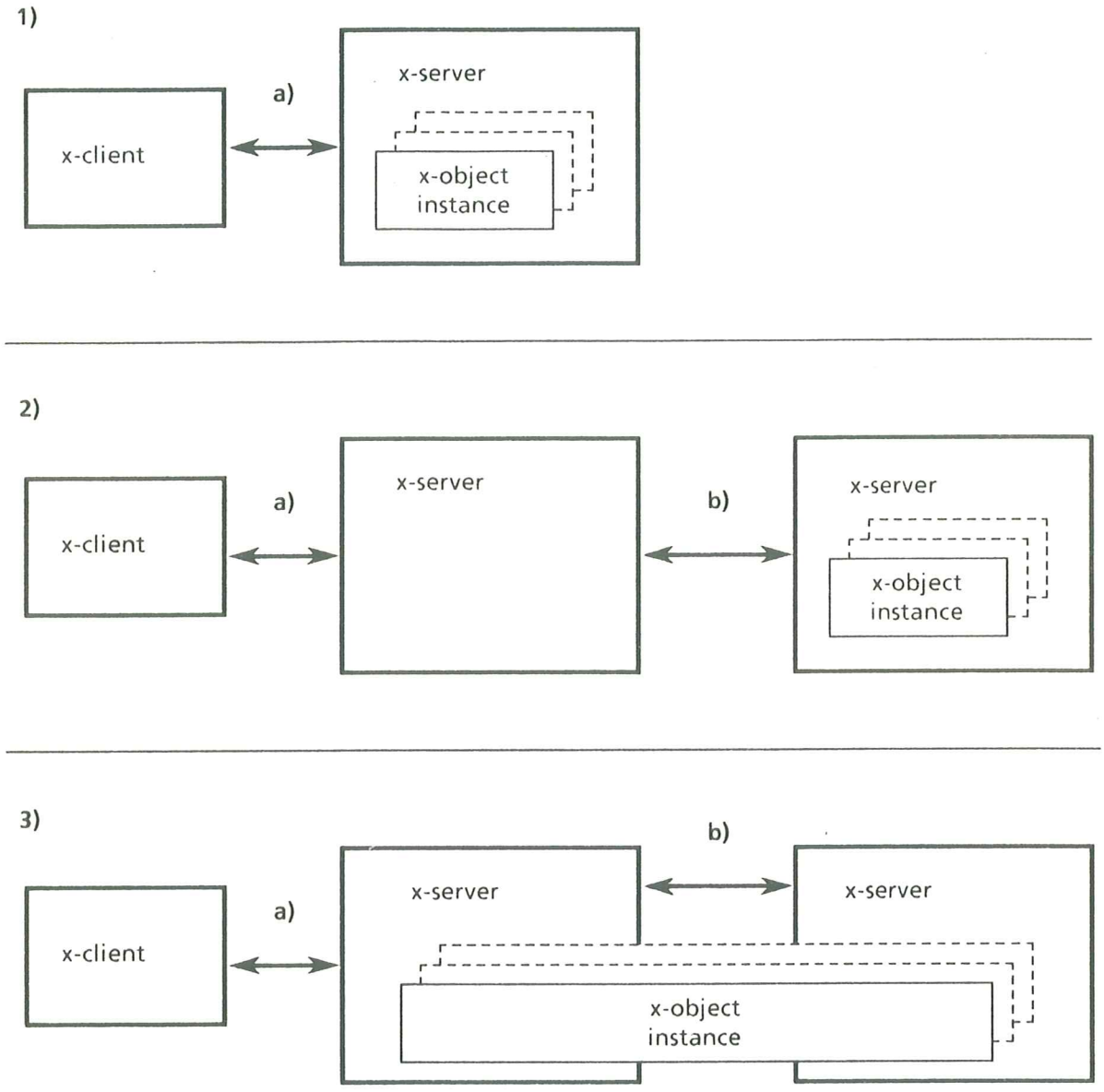
Within the series of interactions, within the confinement established by the bind-operation, there may be further confinement, relaxation and change in the set of object instances of interest. These are always within the confinement established by the bind-operation.

The bind-operation and its converse unbind-operation are described in the object-type specifications of the x-server and are part of the x-service-definition.

An x-client object or x-server object may, on closer inspection, consist of several subordinate objects. These subordinate objects may have associated type-operations. A sub series of interactions may be confined to a specific set of these subordinate objects. Identification of these objects and type-operations on these objects are expressed in parameters to the type-operations of the object-type specifications.

8.1.2 Relations between X-server and Object Instances

The server objects of an x-system are hereafter referred to as x-objects. Depending on the nature of x-objects, one or more x-server may be involved in performing a type-operation on an x-object instance on behalf of an x-client.



Legend: a) x-access-protocol
b) x-system-protocol or x-client and x-access-protocol

Figure 17 - Server and Object Instances

Figure 17 illustrates three different situationd of x-object instances in relation to x-server:

1) one or more instances of an x-object type are held completely by the x-server that the x-client communicates with (e.g mailbox instances held by a mailbox server);

2) one or more instances of an x-object type are held completely by an x-server other than the one that the x-client communicates with, (e.g. portion of directory information in the information base of a directory server). Access to the x-object instances between the x-servers is performed either by means of an x-system-protocol, or an x-client and the x-access-protocol;

3) one or more instances of an x-object type are held by several x-servers (e.g. the whole or part of a distributed data base). Accesses to the objects are coordinated by the sharing x-servers, using either the x-system-protocol, or an x-client and the x-access-protocol.

From the point of view of the x-client, there is no distinction between these three situations. The x-client simply specifies the type-operations concerned to the x-server with which the x-client communicates, using the x-access-protocol. Not all three situations described above are necessarily supported by every distributed-office-application.

8.1.3 Object Model and Remote Operations

The Remote Operations defined in ECMA TR/31 provide a notation and a protocol specification for bind-operations, unbind-operations and operations (called type-operations in the object model).

Figure 18 shows an example of an x-access-protocol using the Remote Operations Notation.

```
XServiceAccess  DEFINITIONS  ::=

BEGIN

--Use of macros imported from module RemoteOperations see ECMA/TR 31

BIND            ::=  RemoteOperations.BIND
OPERATION       ::=  RemoteOperations.OPERATION
UNBIND          ::=  RemoteOperations.UNBIND
ERROR           ::=  RemoteOperations.ERROR

-- Definition of bind operation
xBind           BIND
                ARGUMENT    ArgumentOfxBind
                RESULT      ResultOfxBind
                BIND-ERROR  ErrorOfxBind
                ::= 1

-- Definition of unbind operation
xUnBind         UNBIND
                ::= 2


-- Definition of other operations
aOperation      OPERATION
                ARGUMENT    ArgumentOfaOperation
                RESULT      ResultOfaOperation
                ERRORS      { cError, dError }
                ::= 3

bOperation      OPERATION
                ARGUMENT    ArgumentOfbOperation
                RESULT      ResultOfbOperation
                ERRORS      { cError }
                ::= 4

-- Definition of errors
cError          ERROR
                PARAMETER   ParameterOfcError
                ::= 1

dError          ERROR
                ::= 2

END
```

Figure 18 - Example of an X-access-protocol Using Remote Operations

8.2 Application Rules

To fulfill the requirement of clause 2.2.3 in the specific context of a distributed office system, the following rules have been established in order to simplify the management of shared resources among a number of applications.

8.2.1 Concurrency and Resource Sharing

8.2.1.1 Concurrency

There are established techniques in centralized systems to control concurrent access and to preserve the integrity of data. For distributed systems there are no economic general solutions to the general case of distributed data.

Applications should avoid the general case. Until a stronger requirement arises and a solution is worked out for a specific application, the guideline is to use a managed weak consistency - that is:

- allow inconsistent data;
- have one master copy for each item of data, and have one specific server responsible for updating it;
- have one sequence of propagating changes to copies of that data item and changes to related data items;
- minimize relationships between data items in different servers;
- provide administrative controls to regulate how long it takes for a change to propagate;
- design applications to be tolerant of or resilient to out of date data.

With this guidelines, concurrency controls can be limited within a single x-server, or at most within an x-system. The impact on protocols is limited to the impact of resource sharing.

8.2.1.2 Resource Sharing

Shared resources within a server are managed by the server (which in turn relies on the underlying operating system of the node).

The impact on protocols is limited to managing the consequences of a server not being able to respond to an interaction for a time. This can be manifested by a refusal to accept an interaction, a response indicating a delay or a deferred response. The entity acting as a user may impose time-out disciplines.

If required, an x-system may provide for management of resources shared between its x-servers. This will require to be expressed in the x-system-protocol, but will not impact the x-access-protocol except as described above.

8.2.2 Network Transparency

As a corollary of the "distribution of applications" the actual network configuration of a given application should be as transparent as possible to the user. This is made possible by associating to the user-application-process the name of the attached servers, and possibly the name of alternate servers as a backup. This may be achieved by using the Directory application as a bootstrap.

Even a Directory server that provides only the "hint mechanism" satisfies this requirement to some extent. If the "local" Directory server is unable to fulfill the request, it supplies to the requestor the information necessary to contact another Directory server, which can answer the request. Therefore it is not necessary for the the user-application-process to have knowledge about the actual Directory server address and location in the network.

8.2.3 Object Identifier Allocation

The concept of OBJECT IDENTIFIERs as described in ISO DIS 8824 on ASN.1 allows classes of objects to be given identifiers in the protocols, making these objects globally unique over any number of OSI applications using the ASN.1 encoding scheme.

The allocation of OBJECT IDENTIFIER values is built on a hierarchical naming structure. On the top level in this hierarchy are:

- CCITT;

- ISO;

- CCITT and ISO jointly.

Further details are given in Annexes D, E and F of ISO DIS 8824.

8.2.4 Common Definition of Time

All protocols in the Distributed Office Application environment will express the time using the data type, "UTC Time" as defined in ASN.1 (see ISO DIS 882).

Time ::= UTCTime

8.2.5 Common Definition of Name

Note 3:

This is based on ECMA TR/32. It is expected that the definitions will be updated later to reflect the current work on Directory in ISO/CCITT, once that work has been stabilized.

All protocols in the distributed-office-application environment will express names using the data type "Name". A common perception of names and a common definition is needed as explained in clause 4 of this Report.

The ASN.1 encoding of names used by applications adhering to this Framework are as defined in ECMA TR/32.

8.2.6 Use of Attribute Concept

The need for a set of attributes associated with objects has been identified for a number of distributed-office-applications. Examples are "message attributes" associated with messages in a Mailbox server and "document attributes" associated with documents in a a Document Filing and Retrieval server.

This common usage of attributes has led to the concept of having generically defined attributes, each of which represents one piece of information about the associated object.

An Attribute consists of an Attribute Type, an Attribute Interpretation and an Attribute Value. An Attribute Type is a class of information (e.g. a message's priority) which has a meaning to the user of the relevant application. An Attribute Interpretation is a class of information (e.g. INTEGER or String) which has a meaning to the application itself, enabling it to intelligently compare two attributes. An Attribute Value is the information itself (e.g. ABC Corporation). Attribute Interpretation plus Attribute Value may recur within an Attribute.

Attribute Types defined for one application may be reused by another application as long as the definitions and the semantics are the same. The OBJECT IDENTIFIER concept described in clause 8.2.3 is used as a tool to achieve this.

Appendix A gives macro definitions for Attributes and Interpretations.

8.2.7 Third-party-transfer Facility

The access protocols which involve third-party-transfer are based on the Remote Operations, as described in ECMA TR/31. The servers are sinks or sources of large data object values. This clause describes the application of data object type concepts in an access protocol. The

third-party-transfer protocol defines a data type to be used within the access protocol data object occurrence.

Any place in an argument or result of an operation where large data object types may occur, hereafter is referred to as data object occurrence. To enable both two party and third party transfer, a data object occurrence contains either the data object value (in the two party case) or the reference (in the third party case).

If for a specific operation the server is the source, this operation is referred to generically as the produce operation. In this case an individual data object type is conceptually part of the result of that operation. The user (client) has to specify the sink in the argument of the produce operation. The sink might be specified to be: 1) the initiator, 2) a third party or 3) selectable. If the sink is specified to be selectable, the initiator leaves the decision, for initiator or third party to the source. The source may decide for initiator if the amount of data is comparatively small or else for third party, depending on a local rule. The data object occurrence is part of the result of the produce operation. If the sink is specified to be the initiator, the data object occurrence contains the data object value. If the sink is specified to be third party, the data object occurrence contains the reference.

If for a specific operation the server is the sink, this operation is referred to generically as the consume operation. In this case an individual data object type is conceptually part of the argument of that operation, i.e the data object occurrence is part of the argument of the consume operation.

If for a specific operation the server is both sink and source, this operation is referred to generically as a produce-consume operation. In this case an individual data object type is conceptually either part of the argument or part of the result of that operation, i.e data object occurrences are part of both argument and result.

The third-party-transfer can only work together with other applications that have included the facility to produce and/or consume third-party-transfer data objects in their protocols.

In order to have a common definition of how to specify third-party-transfer data object occurrences in access protocols the TPT-DATA macro is defined in Annex A.

8.2.8 Global References of Data Objects

In clause 8.2.7 the concept of a global reference to a data object has been introduced in a particular case of third-party-transfer facility. Sometimes the use of global references is justified outside the scope of third-party-transfer. This generalized "Third Party Cooperation" may use different kinds of protocols to get access to referenced data object values: access protocols or system protocols.

The general situation can be described in the following term. The user asks an x-server (source) for a global reference to an object held by this server, and then passes this reference to another y-server (sink), which in turn uses this reference to get access to the referenced data object ("reference resolution") for executing some operation on it. The protocol between the sink and the source server, used for resolution of the given reference, may be, according to the type of operation to be performed on the referenced object, the third-party-transfer protocol, or the x-access-protocol, or else the x-system-protocol (when both source server and sink server are x-server of the same x-application). In the case when several protocols are defined between the sink and the source servers (e.g. third-party-transfer protocol and x-access-protocol), the sink server needs to know which protocol to use for the resolution of the given reference. Normally, the protocol to be used will be defined implicitly by the specific context of the given occurrence of the reference (specific occurrence in the argument of a specific operation). If however for some operation the choice of a protocol still exists, the desired protocol must be explicitly named (as part of the operation's argument).

If some protocol, defined for the source server's application is not implemented on the sink server, then any request for the use of this protocol for the resolution of a global reference will be rejected by the sink server. It is desirable for the user to know in advance which protocols can be used between the given pair of servers; the best moment to obtain this

information is the bind-operation. When the user establishes an association with a server, the latter will specify, in the bind-operation's result, all protocols it supports. Given this information for both source and sink servers, it is the user's responsibility to choose the appropriate protocol in each specific case.

In this way, any protocol may accommodate the same concept of a global reference, and any protocol specification may use, when appropriate, the same related macros to specify data objects and data object occurrences..

To be universally recognizable, any global reference must contain, at least, two different parts: a local reference, which uniquely identifies the referenced object within the scope of a given (source) server, and the addressing part, which uniquely identifies this source server. The former has, a priori, no meaning either for the user, nor for the sink server, while the latter has globally recognizable value.

APPENDIX A

ASN.1 MACROS

Figure A.1 of this Annex defines some macros which may be used in protocols for distributed-office-applications. It is expected that current work on several standards will replace this Annex A as reference, once that work is stabilized.

```
ECMA-TR-DOA-Framework  DEFINITIONS   ::=

BEGIN

--Macros for Attributes

INTERPRETATION MACRO ::=
BEGIN
   TYPENOTATION       ::=    "REPRESENTED BY" type
   VALUENOTATION      ::=    value (VALUE OBJECT IDENTIFIER)
END

ATTRIBUTE MACRO ::=
BEGIN
   TYPENOTATION       ::=    Occurrence Interpretation |
                             Occurrence Representation
   VALUENOTATION      ::=    value (VALUE OBJECT IDENTIFIER)

   Occurrence         ::=    empty | "MULTIVALUED"
   Interpretation     ::=    "INTERPRETATION IS" value (VALUE OBJECT IDENTIFIER)
   Representation     ::=    "REPRESENTED BY" type
END

--Macro for third-party-transfer data object occurrence

TPT-DATA MACRO ::=
BEGIN
   TYPENOTATION       ::=    "{" type (Data-object-type) "}"
   VALUENOTATION      ::=    value (VALUE Data-Object-Occurrence)

   <Data-Object-Occurrence   ::=   CHOICE
                                   {   byValue       [0] Data-object-type,
                                       byReference   [1] Reference }
     -- Reference for further study
   >
END

END
```

Figure A.1 - Macro Definitions