ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-137

DOCUMENT FILING AND RETRIEVAL (DFR)

Part 1 - Abstract-Service Definition and Procedures

Part 2 - Protocol Specification

January 1990

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

Phone: +41 22 735 36 34 Fax: +41 22 786 52 31

ECMA
EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-137

DOCUMENT FILING AND RETRIEVAL (DFR)

Part 1 - Abstract-Service Definition and Procedures

Part 2 - Protocol Specification

January 1990

BRIEF HISTORY

The Document Filing and Retrieval Application is one of a series of standards defining applications needed in the area of office automation, as described in the Distributed Office Application Model [ISO/IEC 10031]. This ECMA standard provides the functionality of document filing and retrieval which directly supports the user in an office environment. Thus Document Filing and Retrieval is not a general standardization of all types of filestores as they may exist in computing systems. Rather it concentrates on the filing and retrieval of documents, as related to the task of office work. Document Filing and Retrieval aims only at standardizing the model of such document stores and the associated services and protocols defining the principles of how clients can access such document store servers, where clients and servers reside on different nodes of a distributed office system.

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 microprocessors and large or medium sized mainframes interconnected through local area network or wide area 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.

This ECMA Standard has been developd by ECMA-TG32 in close cooperation with ISO/IEC SC18 WG4. A similar standard is under development in WG4 (see ISO/IEC DIS 10166).

Adopted as an ECMA Standard by the General Assembly of 14th December 1989.

Table of Contents

Part 1

Abstract-Service Definition and Procedures

SECTION ONE - INTRODUCTION

1. SCOPE AND FIELD OF APPLICATION

1.1 Scope

This Standard ECMA-137 consists of two parts:

Part 1: Document Filing and Retrieval - Abstract-Service Definition and Procedures

Part 2: Document Filing and Retrieval - Protocol Specification

The Document Filing and Retrieval Application (DFR) provides the capability for large capacity non-volatile document storage to multiple users in a distributed office system. This facility is particularly useful in an environment where a large population of desktop workstations that have limited storage capacity require access to large expensive storage devices.

Documents have associated attributes, to facilitate and control retrieval. Use of these attributes according to given algorithms will enable documents in the document storage to be browsed, retrieved, managed and deleted in a variety of ways. Access control protects documents from unauthorized operations. Documents can be stored in nested groups. References to documents and groups can be created and also stored in nested groups. With specific attributes a document can be designated a version of another document. Single documents, references or groups can be moved from one group into another group. Enumeration of groups, identification by other attributes besides names, identification by conditions over attributes, search for documents meeting search criteria, concurrent access to the same document, reference or group of documents are further functions provided by this standard for the user requirements in an office environment.

The Document Filing and Retrieval Application is a Distributed Application located in the Application Layer of the Reference Model for Open Systems Interconnection (see ISO 7498).

It should be noted that a Document Filing and Retrieval Application will provide storage for an open-ended set of document types. The content of the documents stored is transparent to the Document Filing and Retrieval Server.

NOTE 1
This Standard deals with individual Document Filing and Retrieval Servers, it defines the Document Filing and Retrieval (DFR) Protocol. The Standard governs the interactions of a Document Filing and Retrieval Client and a single Document Filing and Retrieval Server. Future standardization will consider the facilities of a Distributed Filing and Retrieval Server System and the need for inter-server protocols and a DFR Administration Protocol. It is intended that the results of the initial standardization work be extendible and support this future work.

NOTE 2
This Standard does not presently include administration aspects of the Document Filing and Retrieval abstract-service. For the time being these aspects are left to local implementation, although they are candidates for future standardization.

1.2 Field of Application

This Part 1 of this ECMA Standard specifies the Document Filing and Retrieval Abstract-service that enables a User to communicate with a remote Document Filing and Retrieval Server in order to access a remote document store.

This Part 1 of this ECMA Standard:

- defines a Client-Server type model in accordance with the Distributed Office Application Model;
- defines functions and services provided by Document Filing and Retrieval Servers;

- defines a specific Document Filing and Retrieval model for managing Documents and Groups of Documents;
- defines the Document Filing and Retrieval Abstract Service using the principles established by the Abstract Service Definition Conventions [ISO 10021-3];
- defines the usage of other Services.

This ECMA Standard serves the following important fields of application:

- capability for large capacity document storage for use by multiple users in a distributed system;
- ordered filing and multi-key retrieval of documents;
- structured organization of groups of documents;
- storage of an open-ended number of different document types;
- referencing documents and groups ;
- filing and referencing of documents outside of the document storage (for example, non-electronic hard copy documents);
- adjoining attributes to documents, groups and references, independent of the content;
- capabilities to store, retrieve and delete documents of the document store whatever their content;
- capabilities to search for, order, retrieve, and delete single documents or groups of documents using document attributes;
- capabilities to maintain different versions of a document, including such concepts as "previous version", "next version" and "last version";
- protection against unauthorized storage and retrieval of documents;
- capabilities to control concurrent access to DFR objects.

2. REFERENCES

2.1 Reference Model and DOAM References

ECMA-131	Referenced Data Transfer (ISO/IEC DP 10031-2)
ECMA TR/42	Framework for Distributed-Office-Applications (ISO/IEC DP 10031-1)
ISO 7498	Information Processing Systems - Open Systems Interconnection - Basic Reference Model
ISO/IEC 10021-3	Information Processing Systems - Text Communication - MOTIS - Part 3: Abstract Service Definition Conventions

2.2 Presentation References

ISO 8824	Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)
ISO 8824 DAD1	Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1): Draft Addendum 1 on ASN.1 Extensions

ISO 8825 Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)

ISO 8825 DAD1 Information Processing Systems - Open Systems Interconnection - Specification for ASN.1: Draft Addendum 1: ASN.1 Extensions

2.3 Association Control References

ISO 8649 Information Processing Systems - Open Systems Interconnection - Service Definition for the Association Control Service Element

ISO 8650 Information Processing Systems - Open Systems Interconnection - Protocol Specification for the Association Control Service Element

2.4 Reliable Transfer References

ISO/IEC 9066-1 Information Processing Systems - Text Communication - Reliable Transfer - Part 1: Model and Service Definition

ISO/IEC 9066-2 Information Processing Systems - Text Communication - Reliable Transfer - Part 2: Protocol Specification

2.5 Remote Operations References

ISO/IEC 9072-1 Information processing systems - Text Communication - Remote Operations Part 1: Model, Notation and Service Definition

ISO/IEC 9072-2 Information Processing Systems - Text Communication - Remote Operations Part 2: Protocol Specification

2.6 Directory References

ISO/IEC 9594-1 Information Processing Systems - The Directory - Overview of Concepts, Models and Services

ISO/IEC 9594-2 Information Processing Systems - The Directory - Models

ISO/IEC 9594-3 Information Processing Systems - The Directory - Abstract Service Definition

2.7 Office Document Architecture References

ECMA-101 Open Document Architecture (ODA) and Interchange Format
Part 1: Introduction and General principles (ISO/IEC 8613-1)
Part 4: Document Profile (ISO/IEC 8613-4)
Part 5: Office Document Interchange Format (ISO/IEC 8613-5)

2.8 Message Handling References

ISO/IEC 10021-5 Information Processing Systems - Text Communication - MOTIS - Part 5: Message Store: Abstract Service Definition

3. DEFINITIONS

3.1 Ancestor

The *Parent* of a *DFR-Object* and, recursively, any Ancestor of the former, including the *DFR-Root-Group*.

3.2 Attribute-Type

That component of an attribute which indicates the type of information given by the *Attribute Value*.

3.3 Attribute-Value

A particular instance of that class of information indicated by an *Attribute-Type*.

3.4 Attribute-Value-Assertion

A proposition, which may be true, false, or undefined, concerning the values of *Attributes* in a *DFR-Entry*.

3.5 Conceptual-Document

A set of *DFR-Documents*, considered to be "different *Versions* of the same document".

3.6 Control-Attribute-Package

A collection of attributes used to control access to a *DFR-Object*.

3.7 Descendant

For a given *DFR-Group*, any of the *DFR-Group Members*, and recursively, any Descendant thereof.

3.8 DFR-Attribute

A distinctive characteristic of a *DFR-Object*.

3.9 DFR-Basic-Attribute-Set

The set of *DFR-Attributes*, that will be mandatorily supported by every *DFR-Server*.

3.10 DFR-Content

The prime information content of a *DFR-Object*. The nature of the DFR-Content depends on the DFR-Object-class of the *DFR-Object*.

3.11 DFR-Document

A structured amount of information that can be filed, retrieved, and interchanged consisting of a *DFR-Document-Content* and associated *DFR-Attributes*.

3.12 DFR-Document-Content

A body of information actually contained within the document, e.g an office document, and not interpreted by DFR.

3.13 DFR-Document-Store

A named collection of *DFR-Objects* which is logically arranged in a hierarchical structure.

3.14 DFR-Entry

A *DFR-Object* together with additional *DFR-Attributes* describing its hierarchical place in the DFR-Document-Store.

3.15 DFR-Extension-Attribute-Set

The set of *DFR-Attributes* (beyond the *DFR-Basic-Attribute-Set*) defined in this Standard and optionally supported by some *DFR-Servers*.

3.16 DFR-External-Reference

A *DFR-Reference* whose *Referent* is in another *DFR-Document-Store*.

3.17 DFR-Group

A collection of *DFR-Objects* in a *DFR-Document-Store* which are called *DFR-Group Members* of the *DFR-Group*. A DFR-Group consists of *DFR-Attributes* which are associated with the DFR-Group as a whole and a *DFR-Group-Content*.

3.18 DFR-Group-Content

A sequence of UPIs, identifying all the *DFR-Group-Members* of the *DFR-Group*.

3.19 DFR-Group-Member

A *DFR-Object* which is identified in the *DFR-Content* of it's parent *DFR-Group*.

3.20 DFR-Internal-Reference

A *DFR-Reference* whose *Referent* is in the same *DFR-Document-Store*.

3.21 DFR-Membership-Criteria

A *DFR-Attribute* of a *DFR-Group* establishing constraints on *DFR-Group* membership based on attribute values.

3.22 DFR-Object

One of a set of information entities managed by a *DFR-Server*. DFR-Objects defined are *DFR-Documents, DFR-Groups, DFR-References* and *DFR-Search-Result-Lists*.

3.23 DFR-Object-Class

A *DFR-Attribute* indicating the class of a *DFR-Object* (*DFR-Document, DFR-Group, DFR-Reference* or *DFR-Search-Result-List*).

3.24 DFR-Object-Tree

The DFR-Object-Tree of a *DFR-Group* is the tree formed by a specific *DFR-Group* and all its *Descendants*.

3.25 DFR-Pathname

A *DFR-Attribute* used to help identify a *DFR-Object* in a *DFR-Document-Store*. The DFR Pathname is formed by a sequence of values of the DFR-Title attribute of all *Ancestors* of the *DFR-Object* to be identified with the DFR-Title of the *DFR-Object* itself being the last in the sequence.

3.26 DFR-Proper-Group

Any *DFR Group* other than the *DFR-Root-Group*.

3.27 DFR-Reference

A *DFR-Object* which acts as a link to another *DFR-Object*, which is called the *Referent* of the DFR-Reference.

3.28 DFR-Reference-Content

The information stored in a *DFR-Reference* for the purpose of identifying the *Referent*.

3.29 DFR-Root-Group

The distinguished *DFR-Group* within a *DFR-Document-Store* having no *Ancestor* and the *DFR-Object-Tree* of which encompasses all *DFR-Objects* in the *DFR-Document Store*.

3.30 DFR-Search-Criteria

A *Filter*.

3.31 DFR-Search-Result-List

A *DFR-Object* which has information about a set of *DFR-Objects* satisfying specified search criteria.

3.32 DFR-Search-Result-List-Content

Information about the result of a DFR Search abstract operation.

3.33 DFR-Server

That part of the DFR which supplies Document Filing and Retrieval services.

3.34 DFR-Unique-Permanent-Identifier

A *DFR-Attribute* assigned to every *DFR-Object* by the *DFR-Server* to identify unambiguously a *DFR-Object* within the *DFR-Document-Store*.

3.35 DFR-User

The consumer of services supplied by a *DFR-Server*. At any time it is acting for a security subject and takes on the security privileges of that security subject.

3.36 Filter

A construct specifying assertions about the presence or value of *DFR-Attributes*, it is intended to be the same as in Directory (ISO/IEC 9594) and Message Store (ISO/IEC 10021-5).

3.37 Member

See *DFR-Group-Member*.

3.38 Owner

A Security subject, with Owner Access right to a specific DFR-Object.

3.39 Parent

Each *DFR-Object*, except the *DFR-Root-Group*, is a *DFR-Group-Member* of a *DFR-Group*, which is called its Parent

3.40 Privilege-Attribute-Certificate

A certified set of access privileges that can be presented by a *DFR-User* to establish access rights.

3.41 Referent

That *DFR-Object* to which a *DFR-Reference* refers.

3.42 Version

DFR-Document specified by the user as a derivation of one or more other *DFR-Documents* by means of specific *DFR-Attributes*.

4. ACRONYMS

AE	Application Entity
ASN.1	Abstract Syntax Notation One
CAP	Control-Attribute-Package
DFR	Document Filing and Retrieval Application
DOAM	Distributed Office Application Model
DS	DFR-Document-Store
PAC	Privilege-Attribute-Certificate
QoS	Quality of Service
RDT	Referenced Data Transfer
UPI	DFR-Unique-Permanent-Identifier

5. CONVENTIONS

This Part 1 of this ECMA Standard uses the description conventions listed in the following clauses.

5.1 Conventions for Abstract-services

This Part 1 of this ECMA Standard uses the following ASN.1-based descriptive conventions for the indicated purposes:

1) ASN.1 itself, to specify the abstract-syntax of information-objects and their components, common data-types, and state-variables.

2) The ASN.1 OBJECT and PORT macros and associated abstract-service definition conventions of ISO/IEC 10021-3, to specify the DFR Port.

3) The ASN.1 ABSTRACT-BIND, ABSTRACT-UNBIND, ABSTRACT-OPERATION, and ABSTRACT-ERROR macros of ISO/IEC 10021-3, to specify the DFR abstract-service.

4) The ASN.1 ATTRIBUTE MACRO and ATTRIBUTE SYNTAX MACRO from ISO/IEC 9594-2, to specify attributes and attribute syntaxes.

NOTE 3
ASN.1 specifications in this Standard make full use of the ISO 8824 Addendum 1 features, especially such syntactical constructs as "WITH COMPONENTS" (sub-typing of sequences, sets and choices). All specifications are written using "IMPLICIT TAGS" convention, which means systematic omission, at the time of ASN.1 encoding, of all unnecessary "nested" tags, especially those "recovered" by context-specific ones.

The DFR as any ROSE-based standard does not exploit the Presentation Layer facilities for coping with difference between the local encoding and local syntaxes of each open system (see the use of EXTERNAL in 6.3.2.1).

5.2 Conventions for Text in General

For the terms used in this Part 1 of this ECMA Standard the following rules apply:

1) Single terms beginning with a capital letter and compound terms (chained by hyphens and each word also beginning with a capital letter) are defined terms. For the definitions see 3, for the Attributes see also 9. Exceptions are the titles of the International Standards, which also begin with capital letters, see clause 2.

2) Single terms and compound terms (written together without hyphens) which are rendered in **bold** are either ASN.1-specified data-type names or their component identifiers. For the definitions refer to the Appendices or to the corresponding clauses in the main text.

The following characters are used in this Standard to indicate whether a parameter, an attribute or other items described are mandatory, optional or conditional. That is:

M (Mandatory) stands for the condition that an item shall be present in any case (must be supported by DFR);

O (Optional) stands for the condition that an item shall be present at the discretion of a DFR implementer;

C (Conditional) stands for the condition that an item shall be present under some circumstances defined in this Standard.

SECTION TWO - DFR ABSTRACT-SERVICE DEFINITION

6. DFR ABSTRACT MODEL

This clause provides an abstract functional model of Document Filing and Retrieval. For an introduction and description of the abstract-service concept and its definition conventions, see ISO/IEC 10021-3.

The Document Filing and Retrieval environment comprises two atomic objects, the Document Filing and Retrieval Server (DFR-Server) and the Document Filing and Retrieval User (DFR-User). A DFR-Server is modelled as an atomic object, which acts as a provider of services to the DFR-User. The DFR-Server is described using an abstract model in order to define the service provided by the DFR-Server - the Document Filing and Retrieval abstract service. Figure 1 shows the DFR model.

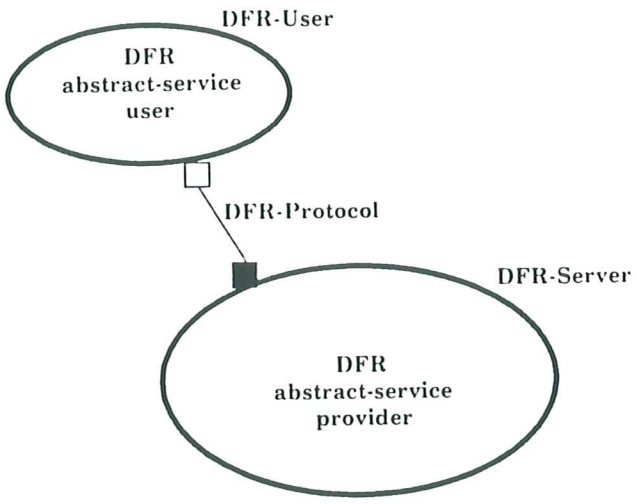

Fig. 1 - Document Filing and Retrieval Abstract-service

6.1 Objects in DFR Environment

The DFR-Server is modelled as an atomic object. It supplies the DFR Port abstract-services to the DFR -User.

The formal definition of the DFR-Server object is as follows:

```
dfr-server      OBJECT
                PORTS { dfr-port [S]}
                ::= id-dfr-server
```

The DFR-User is modeled as a separate object. The DFR-User consumes the DFR Port abstract-services supplied by the DFR-Server.

```
dfr-user        OBJECT
                PORTS { dfr-port [C] }
                ::= id-dfr-user
```

The DFR-Server Port is the concern of this ECMA Standard for operations on DFR-Objects.

6.2 DFR Port

A DFR-User is joined to, and interacts with, a DFR-Server by means of the DFR Port. The collection of capabilities provided by this port forms the DFR-Server Abstract-service. These capabilities include obtaining information on, fetching, and deleting documents residing in the DFR-Server.

By means of the abstract-bind-operation, the DFR-Server authenticates a user by checking the user's credentials or certified identity and access privileges (PAC) before providing it with any of the filing and retrieval capabilities.

This asymmetric abstract port is defined for any DFR-User (in the consumer role) and for any DFR-Server (in the supplier role). Such a pair of abstract ports makes it possible for any DFR-User to communicate with a DFR-Server with a view to executing ordinary DFR-operations. All these operations are also asymmetrical. Each of them is invoked by the consumer (the DFR-User) and performed by the supplier (the DFR-Server).

The DFR Port is defined as follows:

```
Dfr  PORT
     CONSUMER INVOKES {
                                Create,
                                Delete,
                                Copy,
                                Move,
                                Read,
                                Modify,
                                List,
                                Search,
                                Reserve,
                                Abandon }
     SUPPLIER INVOKES { }
     ::= id-pt-dfr
```

6.3 Information Model

A DFR-Server offers its users operations on DFR-Objects in its DFR-Document-Store. The object classes of DFR-Objects are DFR-Documents, DFR-Groups, DFR-References, and DFR-Search-Result-Lists. A DFR-Group is either a DFR-Root-Group or a DFR-Proper-Group. A DFR-Reference is either internal or external. This gives the following specification for all DFR-Object classes.

```
DfrObjectClass ::= ENUMERATED {
                        dfr-document            (0),
                        dfr-root-group          (1),
                        dfr-proper-group        (2),
                        dfr-internal-reference  (3),
                        dfr-external-reference  (4),
                        dfr-search-result-list  (5) }
```

A DFR-Document-Store is managed and accessed by the DFR-Server. A DFR-Document-Store is assigned to one DFR-Server only. A DFR-Server manages one DFR-Document-Store. The assignment of DFR-Servers to DFR-Document-Stores, that is which DFR-Server manages which

DFR-Document-Store, may be noted in the OSI Directory [ISO/IEC 9594]. A DFR-Document-Store is a named collection of DFR-Objects which is logically arranged in a hierarchical structure. An example of how different DFR-Objects in a DFR-Document-Store are related to each other is illustrated in Figure 2 according to the definitions below. Group membership is depicted in the figure by solid lines.

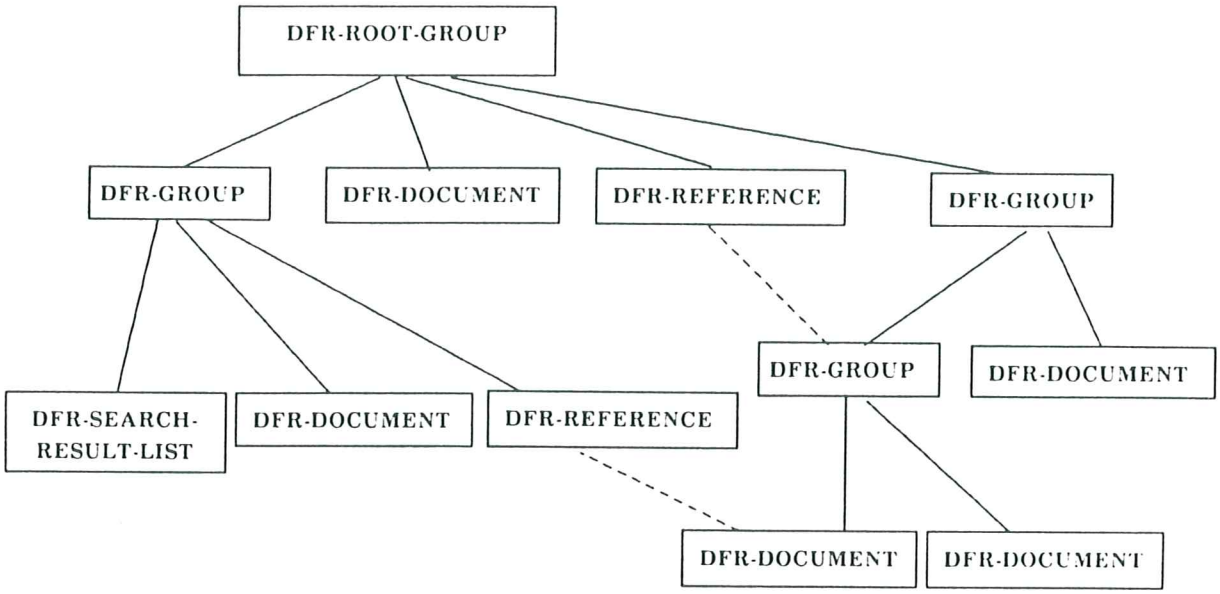

Fig. 2 - Example of DFR Document Store Structure

A DFR-Object is a member of one and only one DFR-Group (parent group), an exception is the DFR-Root-Group that has no parent group. A DFR-Object may participate indirectly in more than one DFR-Group by means of DFR-References. A DFR-Reference refers to one and only one DFR-Object. A reference to a DFR-Reference is not permitted. A DFR-Object may be referenced from more than one DFR-Reference. Each DFR-Group represents a DFR-Object-Tree which is formed by all descendants of that DFR-Group. A DFR-Search-Result-List contains information about a set of DFR-Objects satisfying some selection criteria.

A DFR-Object consists of DFR-Attributes and DFR-Content. A DFR-Object is introduced into a DS by creating a DFR-Entry for it. A DFR-Entry is formed by a DFR-Object together with additional DFR-Attributes describing its hierarchical place in the DFR-Document-Store. The immediately containing DFR-Group is the parent of the DFR-Object, and the DFR-Object is a member of that DFR-Group.

```
DfrEntry ::= SEQUENCE {
        attributes [0] DfrEntryAttributes,
        content    [1] DfrObjectContent }
```

The attributes of a DFR-Entry are those of the correspondent DFR-Object plus two supplementary DFR-Attributes, DFR-Parent-Identification and DFR-Pathname.

The attributes forming the **DfrEntryAttributes** are defined either in this Standard, or may be defined externally. This is modelled by attribute sets, a basic attribute set and extension attribute sets. Two attribute sets are defined in this Standard, the mandatory DFR-Basic-Attribute-Set and the optional DFR-Extension-Attribute-Set. Other optional extension attribute sets possibly including Security attributes may be defined elsewhere. The DFR-Basic-Attribute-Set is

contained in the abstract syntax of the DFR access protocol. Each extension attribute set requires at least one supplementary abstract syntax. The abstract syntax of the DFR-Extension-Attribute-Set is defined in Appendix E.

NOTE 4
Negotiation of extension attribute sets is performed by means of negotiation of the corresponding abstract syntax during the Bind abstract operation.

DfrEntryAttributes :: = SET OF Attribute

The **DfrObjectContent** is the actual information stored with the DFR-Object. The nature of this information depends on the **DfrObjectClass**. The DFR-Content of a DFR-Group is a sequence of UPIs of all its Members. The DFR-Content of a DFR-Document is a body of information, e.g. an office document. The DFR-Content of a DFR-Reference is a logical pointer to some other DFR-Object (DFR-Group or DFR-Document or DFR-Search-Result-List), called the Referent.

DfrObjectContent :: = CHOICE {

document-content	[0] DfrDocumentContent,
root-group-content	[1] DfrGroupContent,
proper-group-content	[2] DfrGroupContent,
internal-reference-content	[3] DfrInternalReferenceContent,
external-reference-content	[4] DfrExternalReferenceContent,
search-result-list-content	[5] DfrSearchResultListContent }

Each DFR-Object has a unique identification within the DFR-Document-Store. This identification is given by a Unique-Permanent-Identifier (UPI). A UPI is assigned to each DFR-Object by the DFR-Server. Once assigned by a DFR-Server, the value of each UPI will not be changed within the life of a DFR-Object. In addition, this UPI value will differ from that of all DFR-Objects which once existed in the same DFR-Server (including all existing DFR-Objects and all deleted DFR-Objects).

DfrUniquePermanentIdentifier :: = OCTET STRING

NOTE 5
UPI is also modelled as a DFR-Attribute, see 9.2.1.

6.3.1 DFR-Document-Store

A DFR-Document-Store is a collection of DFR-Objects logically arranged in a hierarchical structure.

All DFR-Objects in a DFR-Document-Store are accessible via the DFR-Server.

Any DFR-Object within a DS can be accessed via its UPI or via the DFR-Pathname. The UPI is assigned to a DFR-Object by the DFR-Server when the object is stored in a DFR-Document-Store in order to unambiguously identify the DFR-Object. The DFR-Pathname is formed by a sequence of values of the DFR-Title attribute from all Ancestors of the DFR-Object to be identified with the DFR-Title of this DFR-Object being the last in this sequence. Since the DFR-Title attribute is only unique if such a restriction is defined for the DFR-Document-Store, access via the DFR-Pathname may fail.

6.3.2 DFR-Documents

A DFR-Document is a structured amount of information that can be interchanged, stored, and retrieved. A DFR-Document consists of a DFR-Document-Content along with DFR-Attributes which are associated with the DFR-Content. A DFR-Document is contained in one DFR-Document-Store.

NOTE 6
Maintenance of consistency between DFR-Documents and their copies in different DFR-Document-Stores are outside the scope of this standard, and left to the responsibility of the DFR-User.

DFR-Attributes are additional data items used to identify and to describe properties of a DFR-Document. The DFR-Content is the actual information stored in the DFR-Document, and is not interpreted by the DFR-Server.

6.3.2.1 DFR-Document-Content

The DFR-Document-Content is a body of information that has been provided to a DFR-Server for the purpose of storage.

```
DfrDocumentContent :: = EXTERNAL (WITH COMPONENTS {
                              ... ,  direct-reference   PRESENT,
                                     indirect-reference ABSENT,
                                     encoding (WITH COMPONENTS {
                                                  ... , arbitrary ABSENT })})
```

The direct-reference component is an OBJECT IDENTIFIER whose value is the same as the value of the DFR-Document-Type attribute of the DFR-Document (see 9.2.3).

Upon request a DFR-Server transfers the content of a DFR-Document to the DFR-User. A DFR-Server never interprets the content of the documents it stores.

NOTE 7
The cooperation of DFR with a document content access protocol is for further study.

6.3.2.2 DFR-Document Attributes

DFR-Document attributes are data items that identify a DFR-Document, describe its Content, help control access to it, or are in some other way associated with the document.

DFR-Attributes serve to qualify or enhance a DFR-Document's content, or to define additional characteristics of the Document or its intended use. The values of certain DFR-Attributes have a particular meaning to the DFR-Server and elicit defined behavior while others are DFR-User defined and controlled.

6.3.3 DFR-References

A DFR-Reference allows a DFR-Object to participate in more than one DFR-Group without requiring distinct copies of that DFR-Object to be created. A DFR-Reference consists of a DFR-Content containing a "pointer" to the referenced DFR-Object (the Referent) and of DFR-Attributes (see 6.3.3.2).

The Referent may be a DFR-Document, a DFR-Group or a DFR-Search-Result-List; it may not be another DFR-Reference. The DFR-Object-Class of the Referent is noted in the DFR-Reference-Content. In the case of a DFR-Reference to a DFR-Document, the content type of the latter is also noted in this DFR-Reference in the Attribute DFR-Document-Type (in the form of an OBJECT IDENTIFIER).

A DFR-Reference refers either to a DFR-Object within the same DFR-Document-Store (DFR-Internal-Reference) or to a DFR-Object in a different DFR-Document-Store (DFR-External-Reference).

If the DFR-Server detects at any time, that the Referent has been deleted, it does not delete the DFR-Reference, but sets to true the DFR-Reference-Deleted attribute of the DFR-Reference.

The content of a DFR-Reference is structured like a Referenced Data Transfer [ECMA-131] reference to simplify its use in some cases in an RDT Transfer-operation. The RDT-reference information stored in a DFR-Reference-Content is modelled in the Document Filing and Retrieval context as follows:

a) Use of an RDT-reference as the content of a DFR-External-Reference: An RDT-reference to an entire DFR-Object may be produced at the DFR-User's request by the DFR-Server of the Referent. This RDT-reference may then be stored in the content of a DFR-External-Reference in another DFR-Document-Store at the DFR-User's request (see ASN.1 specification below). This DFR-External-Reference may later be used in an Referenced Data Transfer [ECMA-131] transfer operation. The DFR-User may also later read the content of this DFR-External-Reference (the RDT-reference) and transfer it to another application for subsequent RDT transfer of the referenced DFR-Object.

b) Use of an RDT-reference as the content of a DFR-Internal-Reference: A DFR-Internal-Reference may be created by the DFR-Server internally as the result of a DFR-User request. The content of a DFR-Internal-Reference is an RDT-reference without ae-identifier and application sub-components (see ASN.1 specification below). It is not possible to use this DFR-Internal-Reference in an RDT transfer operation.

c) Use of an RDT-reference for immediate RDT transfer: an RDT-reference to an entire DFR-Object, to the attributes of a DFR-Object, or to the content of a DFR-Object may be produced at the DFR-User's request by the DFR-Server of the Referent, to be used for immediate transfer of the referenced data to some other (sink) DFR-Server or to another application using the standard RDT mechanism. If some DFR data are to be transferred via RDT to another application, it is necessary that the latter be able to interpret these DFR data (for example the Print Application might only accept a DFR-Document-Content).

6.3.3.1 DFR-Reference Content

The content of a DFR-Reference contains a logical pointer to the Referent and the DFR-Object-Class attribute of the Referent (to indicate whether the Referent is a DFR-Document, a DFR-Group or a DFR-Search-Result-List).

An internal DFR-Reference refers to its Referent by means of the UPI of the latter. An external DFR-Reference refers to its Referent by means of the name of the DFR-Document-Store containing the Referent together with the UPI of the Referent. In both cases of DFR-References, external or internal, a Quality of Service parameter is defined (see 6.3.3.3).

The content of a DFR-Reference is an RDT-reference, with some restrictions on the presence or absence of its optional components. The ASN.1 specification of a DFR-Reference-Content is as follows:

```
DfrInternalReferenceContent :: = RDT-reference
        (WITH COMPONENTS {
            ae-identifier  ABSENT,
            local-reference (WITH COMPONENTS { specific-reference } ),
            data-object-type (DfrObjectClassID (
                                id-dfr-document |
                                id-dfr-proper-group |
                                id-dfr-search-result-list ) ),
            quality-of-service (WITH COMPONENTS {
                qos-level (WITH COMPONENTS { ... , level-1 ABSENT } ),
                single-use-of-reference (FALSE)} ),
            token ABSENT } )

DfrExternalReferenceContent :: = RDT-reference
        (WITH COMPONENTS {
            ae-identifier,
            local-reference ,
            data-object-type (DfrObjectClassID (
                        59      id-dfr-document |
                                id-dfr-root-group |
                                id-dfr-proper-group |
                                id-dfr-search-result-list ) ),
            quality-of-service (WITH COMPONENTS {
                qos-level (WITH COMPONENTS { ... , level-1 ABSENT } ),
                single-use-of-reference (FALSE)} ),
            token ABSENT } )

DfrObjectClassID :: = OBJECT IDENTIFIER (
                    id-dfr-document |
                    id-dfr-root-group |
                    id-dfr-proper-group |
                    id-dfr-internal-reference |
                    id-dfr-external-reference |
                    id-dfr-search-result-list )
```

The semantics of different components of an RDT-reference in the context of a DFR-Reference-Content is as follows:

ae-identifier
is specified for DFR-External-References only, and identifies the Application Entity of that DFR-Server which actually manages the DS of the Referent. It is a sequence of three alternative forms of AE identification; at least one of them shall be present:

locational-identifier
specifies the AE directly, in terms of its presentation address, application-entity title and application-context name;

logical-identifier
is the distinguished name of the AE in the OSI Directory (ISO/IEC 9594);

indirect-logical-identifier
is the Distinguished-Name of the DS of the Referent (the corresponding OSI Directory (ISO/IEC 9594) entry shall specify the AE of the DFR-Server managing this DFR-Document-Store).

NOTE 8
An RDT-reference to a DFR-Object used as the content of a DFR-External-Reference would normally have both the "locational-identifier" and the "indirect-logical-identifier" components present; however, only the latter component is mandatory.

local-reference
specifies the Referent locally in its DS, as well as the DS itself (the latter in the case of a DFR-External-Reference only); it consist of optional application component and mandatory specific-reference component:

application
(for DFR-External-References only) specifies the Referent's DS locally in the Application Process which currently manages it.

NOTE 9
Such an identification may be useful, first, to distinguish between the DFR-Server and other Servers eventually co-located with it and using the same RDT Service Element; and second, to distinguish (locally) between the Referent's DS and some other DS which may have been later moved to this location. In this latter case, this other DS shall not respond to the same "application" identification as the Referent's DS. On the other side, as this DS identification remains strictly local, it ceases to be valid if the Referent's DS has been moved; so, the actual value of this component shall be specified in the Directory entry for this DS, to be read together with the new DS location information.

specific-reference
gives the UPI of the Referent in its store.

data-object-type
contains an OBJECT IDENTIFIER which corresponds to the value of the DFR-Object-Class attribute of the Referent. This OBJECT IDENTIFIER is one of six possible values of the data-type **DfrObjectClassID** (see specification above), which are in one-to one correspondence with the values of the **DfrObjectClass** enumerated type. Not all six values may be used, because a DFR-Reference cannot refer to another DFR-Reference (in the case of an DFR-Internal-Reference it also cannot refer to its own DFR-Root-Group).

quality-of-service
has two components, both are mandatory for DFR-External-References as well as for DFR-Internal-References:

qos-levelq
specifies whether a "fidelity time" is defined for this DFR-Reference (see details in the next paragraph);

single-use-of-reference
is a boolean value, always set to FALSE (i.e. multiple use of reference) in both DFR-External-Reference and DFR-Internal-Reference cases.

6.3.3.2 DFR-Reference Attributes

DFR-Reference attributes are data items associated with a DFR-Reference.

The DFR-Reference attributes are explicitly associated with the DFR-Reference; they may differ from attributes of the Referent. The semantics of some DFR-Reference attributes

however, will normally, at the users discretion, be related to the Referent; in this way they provide additional information about the latter. Some other DFR-Reference attributes are solely related to the DFR-Reference itself. The classification of all DFR-Reference attributes among these two categories is shown in the tables in chapter 9.

Certain attributes have a particular meaning to a DFR-Server and elicit defined behaviour while others are user defined and controlled.

6.3.3.3 DFR-Reference Quality of Service Level

A Reference may be either Guaranteed or Unguaranteed. If it is guaranteed the DFR-Server guarantees that a Referent will exist and will not be changed (neither its DFR-Content nor its DFR-Attributes) as long as at least one Guaranteed Reference to it exists. The Quality of Service (**QoSLevel**) for References is the same as in RDT. The Referent as existing at reference creation time may be the same at dereferencing time, might be changed in between, or may not be available at all at dereferencing time. This situation is modelled by three Levels of QoS:

Level 1 unguaranteed reference: the Referent might be changed or might not be available at dereferencing time.

Level 2 unguaranteed reference: the Referent might be changed or might not be available at dereferencing time. However a change which has occurred since the reference creation time is indicated to the DFR-User.

Level 3 guaranteed reference: as Level 2 but now the fidelity of the reference (i.e. the Referent is identical at reference creation time and dereferencing time) is guaranteed for a certain amount of time (fidelity time).

```
QosLevel ::= CHOICE {
        level-1         [0] IMPLICIT NULL,
        level-2         [1] IMPLICIT UTCTime,    -- produce time --
        level-3         [2] IMPLICIT SEQUENCE {
                produce-time        UTCTime,
                fidelity-time       UTCTime }}
```

In a DFR-Reference, in contrast to an RDT-reference, the "level-1" **QoSLevel** is never used, that is the "produce-time" is always included in the reference. The reason for this is that the DFR-Server automatically maintains and updates DFR-Attributes-Modify-Date-And-Time and DFR-Content-Modify-Date-And-Time attributes of each DFR-Object in the DS, and thus can always indicate whether changes have been made to the Referent.

6.3.4 DFR-Groups

A DFR-Group is a collection of DFR-Objects in a DFR-Document-Store which are called DFR-Group-Members of the DFR-Group. A DFR-Group consists of DFR-Attributes which are associated with the DFR-Group as a whole and a DFR-Group-Content which is a sequence of UPIs of all Members of the DFR-Group.

A DFR-Group may be either a DFR-Root-Group or a DFR-Proper-Group. A DFR-Proper-Group itself is always a Member of some other DFR-Group (Parent). A DFR-Root-Group is not a Member of any other DFR-Group; each DFR-Document-Store contains only one DFR-Root-Group. Any DFR-Object of a given DS can be reached from the DFR-Root-Group of that DS. The creation (deletion) of the DFR-Root-Group can not be performed by the user of the Document Filing and Retrieval Port.

NOTE 10
The creation and deletion of the DFR-Root-Group including the first registration of Owner(s) and other security subjects in the DFR-Access-List (see 6.8) is performed by some administration authority. This administration authority is not specified in this Standard but left to local implementations.

6.3.4.1 DFR-Group-Content

The DFR-Group-Content is a sequence of UPIs of all members of the DFR-Group.

Members of a DFR-Group may be DFR-Documents, DFR-Groups, DFR-References and DFR-Search-Result-Lists.

The ordering of the DFR-Group-members is as defined in the DFR-Ordering attribute (see 9.2.11) of the DFR-Group.

DfrGroupContent :: = SEQUENCE OF DfrUniquePermanentIdentifier

A DFR-Group may be assigned a DFR-Membership-Criteria attribute. The membership criteria govern membership in the group; new DFR-Members are required to satisfy the membership criteria before being added. The attributes of the new DFR-Member are verified against the membership criteria and the addition is denied if the verification fails. A DFR-Group having no specified membership criteria permits any new direct DFR-Member to be added.

When a request to change the membership criteria of a DFR-Group will cause any conflict with the membership for one or more current members of the group, such a request will be rejected.

The membership criteria for a Group apply only to members. That means, if a DFR-Reference is a member of a given DFR-Group, only the attributes of this reference are the criteria to be verified for the membership in this DFR-Group; it is not relevant for the reference's membership in the DFR-Group if the attributes of the corresponding Referent are changed and thus no longer satisfy the membership criteria.

The Members of a DFR-Group may be enumerated by a **List** abstract operation and their attributes examined. When a group is enumerated, the attributes returned for a Reference member are never those of the Referent. The attributes of a Referent can only be obtained by the **Read** abstract operation.

The DFR-Object-Tree of a DFR-Group is formed by the DFR-Group itself and all its Descendants. These Descendants are either Members of this DFR-Group or Members of other Descendants of this DFR-Group, recursively.

Some DFR operations apply to the entire DFR-Object-Tree of a DFR-Group rather than to its DFR-Group-Members only.

6.3.4.2 DFR-Group Attributes

DFR-Group attributes are data items that identify a Group, describe its Content, help control access to it, or are in some other way associated with the Group.

DFR-Group attributes serve to qualify or enhance a group s content, or to define additional characteristics of the Group or its intended use. Certain DFR-Attributes have a particular meaning to a DFR-Server and elicit defined behavior while others are user defined and controlled.

6.3.5 DFR-Search-Result-List

A DFR-Search-Result-List is a DFR-Object intended exclusively for holding results of a **Search** abstract operation.

A DFR-Search-Result-List may only be created with empty content, which is filled in as result of a **Search** abstract operation. The user can read or list a DFR-Search-Result-List, as well as modify its attributes, but the content may only be modified by another **Search** abstract operation. A **Search** abstract operation may also be done without specifying new search criteria (thus updating the content of an existing DFR-Search-Result-List).

6.3.5.1 DFR-Search-Result-List Content

The DFR-Content of a DFR-Search-Result-List holds a sequence of the DFR-Attributes UPI and DFR-Object-Class from all DFR-Objects satisfying the search expression specified in the Filter of a **Search** abstract operation. It also contains additional information regarding the domain and criteria of the search as well as the time and conditions of the most recent **Search** abstract operation execution.

```
DfrSearchResultListContent ::= CHOICE {  empty                    NULL,
                                         produced   SEQUENCE {
    continuation          [0] ContinuationContext OPTIONAL,
    start-date-and-time   [1] UTCTime,
    end-date-and-time     [2] UTCTime,
    object-list           [3] DfrEntryList,
    ordering              [4] OrderingRule OPTIONAL,
    search-domain         [5] SearchDomain,
    search-criteria       [6] SearchCriteria      } }
```

A **DfrSearchResultListContent** may only be "produced" by a **Search** abstract operation; at the **SearchResultList** creation time it is "empty". There is no possibility to change the DFR-Content of a DFR-Search-Result-List except by a re-execution of the **Search** abstract operation in such a way as either to replace or to append to its existing content. **DfrSearchResultListContent** will always be placed in an existing Dfr-Search-Result-List (produced by a previous **Create** abstract operation) which is either empty, i.e. never used, or not empty, i.e. used in a previous **Search** abstract operation. In the latter case the search-criteria and search-domain stored in the **DfrSearchResultListContent** may either be reused (e.g. for Search continuation) or overwritten by the search-criteria and search-domain used in the **Search** abstract operation.

The ASN.1 types of individual components of **DfrSearchResultListContent** are formally defined in 8.1.6.

6.3.5.2 DFR-Search-Result-List Attributes

DFR-Search-Result-List attributes are data items that identify a Search-Result-List, describe its Content, help control access to it, or are in some other way associated with the Search-Result-List.

DFR-Search-Result-List attributes serve to qualify or enhance a Search-Result-List-Content, or to define additional characteristics of the Search-Result-List or its intended use. Certain Attributes have a particular meaning to a DFR-Server and elicit defined behavior while others are user defined and controlled.

6.3.6 DFR Version Management

The DFR Version management is achieved by a combination of specific DFR-Attributes, managed by the DFR-Server in a predefined way, in order to make it possible for the user to have a very flexible user-defined version structure, and to provide him with DFR-Server assistance while navigating through this structure or attempting to modify it. Briefly, the DFR Version management may be qualified as user-defined and server assisted.

A DFR-Document may have several versions, simultaneously present in a DFR-Document-Store. In fact, each "version of a document" is itself an individual DFR-Document (an individual entry in the DFR-Document-Store). The set of all DFR-Documents considered to be "different versions of the same document", defines a Conceptual-Document. Different DFR-Documents which are versions of the same Conceptual-Document are related by means of two DFR-Server maintained attributes, DFR-Next-Versions and DFR-Previous-Versions. The set of all DFR-Documents which are versions of one Conceptual-Document is structured as a directed graph with regard to these two Attributes. The relationships among different versions of a Conceptual-Document are independent of the group structure of the DFR-Document-Store.

All versions of the same Conceptual-Document are distinguished from all other DFR-Documents by means of a special DFR-Server managed attribute DFR-Version-Root. The value of this attribute is the UPI of the (historically) first version of the Conceptual-Document; this value remains valid and unchanged even if this first version is later deleted (because it is no more considered, in this context, as the UPI of the first version, but is instead identifying the overall Conceptual-Document). Versions of the same document may also have other attribute values in common, but this is not mandatory and is not verified by the DFR-Server.

In contrast, different versions of the same (conceptual) document are distinguished, apart from their UPIs, by another special attribute, called Version-Name. This is a DFR-User-specified attribute, intended for the user's perception; the DFR-Server may only optionally reinforce the uniqueness of its values among all versions of the same document.

When a DFR-User creates a DFR-Reference to a multi-version document, this DFR-Reference always points to a particular version (that is, to one specific DFR-Document in the DFR-Document-Store). By copying the DFR-Version-Root attribute of the Referent in the corresponding attribute of the created DFR-Reference, the DFR-User is always able to find (using a search operation with appropriate search criteria) some version of the same Conceptual-Document even after the referenced version disappears.

In the simplest case, all versions of the same document are linearly ordered: each version except the first has exactly one previous version, and each version except the last has exactly one next version. In this case the latest version of a Conceptual-Document may be reached by navigating through it using the DFR-Next-Versions attribute.

In a more general case, a given version of a Conceptual-Document may have several next versions (tree model). In addition, a version may be declared following more than one previous versions (directed graph model). Loops are not possible even in this case, as versions are created sequentially, and each version may be linked only to existing ones (a DFR-Previous-Versions attribute may not be modified for any Document which has a next version).

It is always the DFR-User's action to define some existing or newly created DFR-Document as a new version of some other DFR-Document or DFR-Documents; the latter are explicitly specified by the user as previous version(s) for the newly defined one. Links from the new version to all its predecessors (i.e. their UPIs) are the values of the attribute DFR-Previous-Versions.

At the same time, links in the opposite direction are created: each time a new version is created as a successor of an existing one, the DFR-Next-Versions attribute of the latter contains the link to the successor as a supplementary value.

When "discarding" an existing version, all links to it from its successors are redirected as to point to each of its predecessors, and all links to it from its predecessors are redirected as to point to each of its successors.

Version management applies only to DFR-Documents; versions of DFR-Groups, or of DFR-References, or of DFR-Search-Result-Lists, are not defined.

6.3.7 Attributes

This subclause is provided to support the reader in understanding DFR. The definitions of **Attribute, ATTRIBUTE MACRO, ATTRIBUTE SYNTAX MACRO** and **Filter** are in ISO/IEC 9594.

Each DFR-Object consists of two parts, a set of Attributes, and a "content" entity. DFR-Attributes are managed independently of the content. Attributes can be read and changed. If an attribute is changed the content of that object is not changed. Attributes characterize an object, that is each attribute provides a piece of information about, or derived from, the object to which it corresponds. Attributes affect storage and retrieval of an object and control access to it.

For an overview of the DFR-Attributes see 9.

Some selected attributes are intended for ordering the objects by the value of these attributes. In the case of textual attributes ordering presupposes the existence of some of collating sequences. The definition of collating sequuences is outside the scope of this standard.

6.3.7.1 Introduction

An attribute consists of an attribute type, which identifies the class of information given by an attribute, and the corresponding attribute value(s), which are particular instances of that class appearing in the entry (see Figure 3).

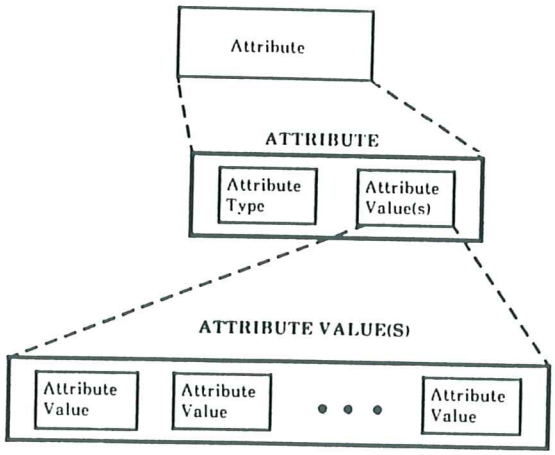

Fig. 3 - Attributes and Attribute Values

```
Attribute ::= SEQUENCE {
  type    AttributeType,
  values  SET OF AttributeValue -- at least one value is required -- }
```

NOTE 11

Thus, for example, in a DFR-Document attribute the attribute-type could be the object

identifier of the DFR-Document-Type attribute, and the attribute-value could be the object identifier for "ODA".

All attributes of a given DFR-Entry must be of distinct attribute types.

6.3.7.2 Attribute-Type

Some attribute types will be internationally standardized. Other attribute types will be defined by national administrative authorities and private organizations. Some externally defined attribute types will not be specific to DFR, for example security attributes. This implies that a number of separate authorities will be responsible for assigning types in a way that ensures that each is distinct from all other assigned types. This is accomplished by identifying each attribute type with an object identifier when the type is defined .

AttributeType :: = OBJECT IDENTIFIER

Attribute-types defined in this Standard are specified in 9.

6.3.7.3 Attribute-values

Defining an attribute type also involves specifying the syntax, and hence data type to which every value in such attributes must conform. This could be any data type.

AttributeValue :: = ANY -- *DEFINED BY type* --

6.3.7.4 Attribute-type Definition and the ATTRIBUTE Macro

The definition of an attribute-type involves :

a) assigning an object identifier to the attribute type;

b) indicating or defining the attribute syntax for the attribute type;

c) indicating whether an attribute of this type may have only one or more than one value (secured).

NOTE 12
A filter may always test for the presence or absence of an attribute of a particular attribute-type.

The DFR ensures that the indicated attribute syntax is used for every attribute of this type. The DFR also ensures that attributes of this type will have one and only one value in entries if attributes of this type are defined to have only one value.

The following ASN.1 macro is used to define an attribute-type:

```
ATTRIBUTE MACRO      ::=
BEGIN
   TYPE NOTATION     ::= AttributeSyntax Multivalued | empty
   VALUE NOTATION    ::= value (VALUE OBJECT IDENTIFIER)

   AttributeSyntax   ::= "WITH ATTRIBUTE-SYNTAX" SyntaxChoice
   Multivalued       ::= "SINGLE VALUE" | "MULTI VALUE" | empty
   SyntaxChoice      ::= value (ATTRIBUTE-SYNTAX) Constraint | type MatchTypes
   Constraint        ::= "("ConstraintAlternative")" | empty
   ConstraintAlternative ::= StringConstraint | IntegerConstraint
   StringConstraint  ::= "SIZE" "("SizeConstraint")"
```

```
SizeConstraint      ::= SingleValue | Range
SingleValue         ::= value (INTEGER)
Range               ::= value (INTEGER) "..." value (INTEGER)
IntegerConstraint   ::= Range

MatchTypes          ::= "MATCHES FOR" Matches | empty
Matches             ::= Match Matches | Match
Match               ::= "EQUALITY" | "SUBSTRINGS" | "ORDERING"
END
```

The correspondence between the parts of the definition and the various pieces of the notation introduced by the macro is as follows:

a) the object identifier assigned to the attribute type is the value supplied in the value assignment of the macro;

b) the attribute syntax for the attribute type is that identified by **AttributeSyntax** production. This either points to a separately defined attribute syntax or explicitly defines an attribute syntax by giving its ANS.1 type and matching rules. If a separately identified attribute syntax is employed, a size constraint for underlying string types or a value range for an underlying integer type may optionally be indicated;

c) the attribute is single valued if the Multivalued production is "SINGLE VALUE" and may have one or more values if it is "MULTI VALUE" or empty.

6.3.7.5 Attribute Syntax Definition

The definition of an attribute syntax involves:

a) optionally, assigning an object identifier to the attribute syntax;

b) indicating the data type, in ASN.1, of the attribute syntax;

c) defining appropriate rules for matching a presented value with a target attribute value. None, some, or all of the following matching rules may be defined for a particular attribute syntax:

   i) equality. Applicable to any attribute syntax. The presented value must conform to the data type of the attribute syntax.

   ii) substrings. Applicable to any attribute syntax with a string data type. The presented value must be a sequence ('SEQUENCE OF'), each of whose elements conforms to the data type.

   iii) ordering. Applicable to any attribute syntax for which a rule can be defined that will allow a presented value to be described as less than, equal to, or greater than a target value. The presented value must conform to the data type of the attribute syntax.

If no equality matching rule is defined, the DFR-Serverwill not attempt to match presented values against target values of such an attribute type.

If an equality matching rule is defined, the DFR-Server:

a) treats values of attributes of this attribute syntax as having type ANY DEFINED BY the data type indicated for the attribute syntax.

b) will only match according to the matching rules defined for that attribute syntax.

c) will only match a presented value of a suitable data type.

The following ASN.1 macro is used to define attribute syntaxes:

```
ATTRIBUTE SYNTAX MACRO ::=
BEGIN
        TYPE NOTATION     ::= Syntax MatchTypes | empty
        VALUE NOTATION    ::= value (VALUE OBJECT IDENTIFIER)
        Syntax            ::= type
        MatchTypes        ::= "MATCHES FOR" Matches | empty
        Matches           ::=  Match Matches | Match
        Match             ::= "EQUALITY" | "SUBSTRINGS" |"ORDERING"
END
```

The correspondence between the parts of the definition and the various pieces of the notation introduced by the macro is as follows:

a) the object identifier assigned to the attribute syntax is a value supplied in the value assignment of the macro;

b) the data type of the attribute syntax is that identified by the Syntax production, i.e., that following macro name.

c) the defined matching rules are equality, if "EQUALITY" appears in the **MatchTypes** production, substrings if "SUBSTRINGS" appears, and ordering if "ORDERING" appears. If the production is empty, then no matching rules are defined.

Should the "empty" alternative of the notation be selected, the resulting notation ("ATTRIBUTE-SYNTAX") can be used to denote any possible attribute syntax.

NOTE 13
No support is provided in the macro for actually defining the matching rules themselves: this must be done by natural language or by other means.

6.3.7.6 Filters

A Filter parameter applies a test that is either satisfied or not by a particular DFR-Entry or not. The **Filter** is expressed in terms of assertions about the presence or value of certain attributes of the DFR-Entry, and is satisfied if and only if it evaluates to TRUE.

NOTE 14
A filter may be TRUE, FALSE, or undefined.

```
Filter ::= CHOICE {
        item    [0] FilterItem,
        and     [1] SET OF Filter,
        or      [2] SET OF Filter,
        not     [3] Filter}
```

A **Filter** is either a **FilterItem** (see 6.3.7.6.1), or an expression involving simpler **Filters** composed together using the logical operators and, or, and not. The **Filter** is undefined if it is a **FilterItem** which is undefined, or if it involves one or more simpler **Filters**, all of which are undefined.

Otherwise, where the **Filter** is:

a) an **item**, it is TRUE if and only if the corresponding **FilterItem** is TRUE;

b) an **and**, it is TRUE unless any of the nested **Filters** is FALSE;

NOTE 15
Thus, if there are no nested Filters, the and evaluates to TRUE.

c) an **or**, it is FALSE unless any of the nested **Filters** is TRUE;

NOTE 16
Thus, if there are no nested Filters, the or evaluates to FALSE.

d) a **not**, it is TRUE if and only if the nested **Filter** is FALSE.

6.3.7.6.1 Filter-item

A **FilterItem** is an assertion about the presence or value(s) of an attribute of a particular type in the DFR-Entry under test. Each such assertion is either TRUE, FALSE or undefined.

```
FilterItem ::= CHOICE {
    equality          [0] AttributeValueAssertion,
    substrings        [1] SEQUENCE {
                              type      AttributeType,
                              strings   SEQUENCE OF CHOICE {
                                 initial [0] ANY, -- DEFINED BY type --
                                 any     [1] ANY, -- DEFINED BY type --
                                 final   [2] ANY }},-- DEFINED BY type --
    greaterOrEqual    [2] AttributeValueAssertion ,
    lessOrEqual       [3] AttributeValueAssertion,
    present           [4] AttributeType,
    approximateMatch  [5] AttributeValueAssertion}
```

Every **FilterItem** includes an Attribute-Type which identifies the particular attribute concerned.

Any assertion about the value of such an attribute is only defined if the **AttributeType** is known, and the purported **AttributeValue**(s) conform to the attribute syntax defined for that attribute type.

NOTE 17
Where these conditions are not met, the Filter is undefined

NOTE 18
Access Control restrictions may require that the FilterItem be considered undefined.

Assertions about the value of an attribute are evaluated using the matching rules associated with the attribute syntax defined for that attribute type. A matching rule not defined for a particular attribute syntax cannot be used to make assertions about that attribute.

NOTE 19
Where this condition is not met, the FilterItem is undefined.

A **FilterItem** may be undefined (as dscribed abowe). Otherwise, where the **FilterItem** asserts:

a) equality, it is TRUE if and only if there is a value of the attribute which is equal to that asserted;

b) substrings, it is TRUE if and only if there is a value of the attribute in which the specified substrings appear in the given order. The substrings must be non-overlapping, and may (but need not) be separated from the ends of the attribute-value and from one another by zero or more string elements.

If initial is present, the substring shall match the initial substring of the attribute value; if final is present the substring shall match the final substring of the attribute value; if any is present the substring may match any substring in the attribute value;

c) greaterOrEqual, it is TRUE if and only if the relative ordering (as defined by the appropriate ordering algorithm) places the supplied value before any value of the attribute;

d) lessOrEqual, it is TRUE if and only if the relative ordering (as defined by the appropriate ordering algorithm) places the supplied value before any value of the attribute;

e) present, it is TRUE if and only if such an attribute is present in the entry;

f) approximateMatch, it is TRUE if and only if there is a value of the attribute which matches that which is asserted by some locally-defined approximate matching algorithm (e.g. spelling variations, phonetic match, etc). There are no specific guidelines for approximate matching in this version of the DFR-Standard. If approximate matching is not supported, this **FilterItem** should be treated as a match for equality.

NOTE 20
In the case of an attribute whose value is of character type, ordering (greater-or-equal and less-or-equal) is defined by the lexicographic order based upon some collating sequence. This standard does not specify how and when a collating sequence is established.

6.3.7.6.2 Attribute-value-assertion

An **AttributeValueAssertion** is a proposition, which may be true, false, or undefined, concerning the values of a DFR-Entry. It involves an attribute type and an attribute value:

AttributeValueAssertion :: = SEQUENCE { AttributeType, AttributeValue }

and is:

a) undefined, if any of the following holds:

i) the attribute type is unknown,

ii) the attribute syntax for the type has no equality matching rules,

iii) the value does not conform to the data type of the attribute syntax;

b) true, if the entry contains an attribute of that type, one of whose values matches that value;

c) false, otherwise.

6.3.8 Security in DFR

Two specific security mechanisms are addressed here: Authentication and Access authorization.

6.3.8.1 Authentication

At bind time, the security subject represented by the DFR-User is authenticated either by using password(s) or, in the case of a user who has been previously authenticated elsewhere, by checking a certified identity. The authentication mechanism verifies the credentials of the DFR-User requesting access to the DFR-Document-Store. This does not, however, qualify the DFR-User to access all DFR-Objects stored in the DFR-Document-Store. Certified identities can also be presented subsequent to the bind-operation, as arguments to individual abstract operations. This permits the bind to be shared between multiple security subjects, the DFR-User representing a different security subject for each of the abstract operations concerned.

6.3.8.2 Access authorization

Access authorization is controlled by security attributes collectively known as a Control-Attribute-Package (CAP).

The CAP is a collection of security related attributes, assigned to DFR-Objects. In order to provide the design freedom necessary for DFR-Servers to operate under a wide variety of security policies, the DFR implementer should be free to choose control attribute types appropriate to the security regimes to which the implementer is targeted.

In the longer term standards will be defined for access authorization rules and for control attribute types, from which implementers will make their choice. Particular examples of such types might be the DFR object's security classification, integrity level or other access group category.

In the short term these standards are not available, and this standard defines one specific control attribute, the Dfr-Access-List.

If the accessing DFR-User has presented, for a DFR-Object, to determine what access the user may have to it, a Privilege-Attribute-Certificate (PAC), containing privilege attributes, these will be used where appropriate in conjunction with the attributes in the CAP. Privilege attributes are optionally presented either at abstract bind time via a PAC type credentials argument (see 7.1.1) or explicitly at the time of the abstract operation request in the privileges argument (see 8.1.3.4), or both. The DFR-User may also specify a proxy PAC in a specific abstract operation request. This proxy PAC is used by the DFR-Server to access another server on behalf of the requesting DFR-User (see 8.1.3.4).

Each specific DFR realization may make use either of the DFR-Access-List attribute, or of other CAP attributes, according to some defined matching algorithms.

6.3.8.3 DFR-Security Policy

All DFR-Objects in a DFR-Document-Store have one or more Owners. An Owner is a DFR-User with specific privileges with regard to the owned DFR-Object.

Each Owner of a DFR-Object is noted in the DFR-Access-List attribute (of the CAP) of that DFR-Object. An Owner of a DFR-Object can add further Owners or delete existing ones. Attempts to delete the last Owner result in an error.

The DFR-Access-List attribute defines the list of security subjects allowed to access a specific DFR-Object, together with their access rights. A security subject registered in the DFR-Access-List (of the CAP) of a DFR-Object is allowed, according to the registered access rights, to access that DFR-Object. A DFR-Object will not be made visible to the DFR-User unless at least read access was granted by the Owner of this DFR-Object. For example, if a DFR-Object satisfies a filter condition specified in a **Search** operation, it will not be made visible to the DFR-User having no read access permission to this DFR-Object.

The DFR-Access-List is an attribute of the DFR-Object. The contents of the DFR-Access-List attribute can be modified only by an Owner. A DFR-User having only read access right to a DFR-Object can only read the own access rights to that DFR-Object. A DFR-Access-List consists of one or more elements. Each element consists of two parts:

```
DfrAccessListElement ::=
  SEQUENCE {
      accessId        AccessId,
      accessRights    AccessRights }

  AccessId ::= DistinguishedName
```

NOTE 21
In the case of a DFR-User having only read access to the DFR-Object, the DFR-Access-List returned differs from the DFR-Access-List assigned to the DFR-Object, because it contains only the users own AccessId and AccessRights.

```
AccessRights ::= ENUMERATED {
  read                (0),
  extended-read       (1),
  read-modify         (2),
  read-modify-delete  (3),
  owner               (4) }
```

Read access to a DFR-Object allows a DFR-User to access that DFR-Object, using **Read**, **Copy**, **Search**, and **List** abstract operations. These abstract operations are permitted to both the content and attributes of the DFR-Object, except those parts of the DFR-Access-List attribute not related to this DFR-User and the attributes DFR-Created-By, DFR-Modified-By, DFR-Reserved-By. Attempts to read the Access-List attribute returns only the initiating DFR-User's access rights. Even if a DFR-Object satisfies the search criteria specified in the **Search** operation, that DFR-Object is not listed in the resulting DFR-Search-Result-List unless the initiating DFR-User has read access permission to the DFR-Object. The DFR-Object will be enumerated in the result of a **List** abstract operation only if the initiating security subject is specified in its DFR-Access-List attribute.

Extended-read access to a DFR-Object includes all privileges of Read access and also permits the user to read the entire DFR-Access-List attribute of that DFR-Object and all other attributes.

Read-modify access to a DFR-Object allows a DFR-User to reserve or unreserve that DFR-Object and modify its content and/or attributes with the exception of the attribute. Modification of the content of a DFR-Group means new Members may be inserted in the group; operations on existing Members in the DFR-Group depend on the rights noted in their DFR-Access-List attributes. Read-modify access includes extended-read access.

Read-modify-delete access to a DFR-Object allows a DFR-User to delete or move that DFR-Object. In the case of a DFR-Group a DFR-User with the Read-modify-delete access right to this DFR-Group may move or delete the whole Object-Tree of this DFR-Group and also may move or delete any of the descendants of this DFR-Group regardless of the access rights noted for this DFR-User in the DFR-Access-Lists of the Descendants. Read-modify-delete access includes read-modify access.

Owner access to a DFR-Object allows a DFR-User to modify the DFR-Access-List attribute of that DFR-Object. Owner access includes read-modify-delete access. A DFR-User creating

a DFR-Object (by **Create** or **Copy** abstract operations) is automatically included in the DFR-Access-List attribute of the new DFR-Object as the Owner.

If a DFR-User does not have at least read access permission to a DFR-Group, this user cannot access its Descendants via this DFR-Group. Direct access to a Descendant object via the object's UPI may still be granted depending upon the DFR-Access-List attribute in place on the DFR-object. Operations involving access to more than one DFR-Object require the appropriate access right to each of these objects.

6.3.8.4 Other Security Policies

A security policy other than that described in 6.3.8.3 may define another set of security-related attributes for both PAC and CAP, and a specific algorithm to compare them in order to deliver access rights (of a security-subject to a DFR-Object) having the same semantics as the DFR access-rights defined in 6.3.8.3. Applying such a security policy does not modify substantially the operational behaviour of a DFR-Server.

Other security policies may define a set of access-rights different from those defined in 6.3.8.3. Applying such a security policy may interfere more seriously with the operation of a DFR-Server. However, the abstract syntax of all interactions between the DFR-User and the DFR-Server shall still conform to that specified in this Standard, in order to maintain a certain level of compatibility between DFR-Servers as viewed by a DFR-User.

7. ABSTRACT-BIND AND ABSTRACT-UNBIND PARAMETERS

7.1 Abstract-bind Parameters

The bind-operation parameters needed by the DFR-Server Port are defined and described in the present clause.

```
DfrBind :: = ABSTRACT-BIND
  TO {dfr-port[S]}
  BIND
    ARGUMENT    DfrBindArgument
    RESULT      DfrBindResult
    BIND-ERROR  DfrBindError
```

7.1.1 Bind-argument Parameters

The **DfrBindArgument** parameter identifies or authenticates the DFR-User. It also accepts a set of restrictions for entries to be returned as result of a DFR abstract-operation.

The definition is:

```
DfrBindArgument :: = SEQUENCE {
    credentials                [0] Credentials,
    retrieve-restrictions      [1] Restrictions OPTIONAL, -- default is none--
    dfr-configuration-request  [2] BOOLEAN DEFAULT FALSE,
    bind-security              [3] BindSecurity OPTIONAL,
    priority                   [4] Priority DEFAULT medium }
```

1) credentials (M): Credentials may be exchanged between the DFR-User and the DFR-Server. Whether the DFR-User is representing a specific end user or not is of no concern to the DFR-Server. The DFR-Server's view of the accessing subject's identity and access privileges is obtained from the credentials passed . Credentials serve to identify a user, authenticate a user, or certify the identity of a user

previously externally authenticated. In the latter case, access control privileges associated with the user can also be passed. The full syntax and semantics of credentials when used for authentication purposes is out of scope of this standard (these matters are common to all bind operations). Use of certified security attributes is described in outline below. Semantic details are however not defined since this is dependent on the actual security policy formulated and implemented by the organization operating the DFR.

The authentication of the user may have taken place external to the DFR-Server; the resulting access control attributes will then be passed in the abstract bind process to allow the DFR-Server to make subsequent access control decisions. This is the function of the PAC construct. Either the DFR-User or the DFR-Server may abort the abstract bind process if the authentication parameters do not justify successful completion of the abstract bind-operation.

The credentials passed in a bind-operation may be modified by the **Privileges** Parameter of a specific DFR operation (see 8.1.3.4).

```
Credentials ::= CHOICE {
    simple      [0] Creds,      -- used for initial authentication --
    certified   [1] PrivilegeAttributeCertificate } -- used when initial authentication --
    -- has already taken place external to the DFR-Server ---
```

a) A **Creds** contains a password associated with the DFR-User.

```
Creds ::= OCTET STRING
```

b) A **PrivilegeAttributeCertificate** contains attributes associated with the DFR-User, for example the user's name, job title or security clearance. These can be used in making access control decisions (see 6.3.8).

```
PrivilegeAttributeCertificate ::= EXTERNAL
```

NOTE 22
The detailed syntax of the PrivilegeAttributeCertificate is under study elsewhere. The access rights needed to make a BIND may under some Security Policies be different from those needed to be established over the BIND; the possibility of using two PACs, one to establish the rights to make the BIND, the other to apply to access to DFR-Objects in the context of the BIND is also under study elsewhere.

2) retrieve-restrictions (O): This contains the restrictions on objects to be returned as a result of an abstract-operation. The restrictions remain until an abstract-unbind-operation is issued.

In the absence of this argument, the default is that no restrictions need to be performed.

This argument consists of the following components:

```
Restrictions ::= SET {
    allowed-document-types  [0] SET OF OBJECT IDENTIFIER OPTIONAL
            --default is  no restriction--
    maximum-result-length   [1] ResultLength OPTIONAL
            --default is no restriction--}

ResultLength ::= INTEGER
```

a) Allowed-document-types (C): The document-types that the DFR-User is prepared to accept as result of a retrieve abstract-operation. Any document with a document-type other than the ones specified, will not be returned, but result in an error, unless the abstract-operation has explicitly overridden the restriction.

In the absence of this component, the default is that no retrieve-restrictions on document-types need to be performed.

b) Maximum-result-length (C): The maximum-result-length that the DFR-User is prepared to accept as result of an abstract-operation. Any result with a result-length exceeding the one specified will not be returned, but result in an error or in a continuation parameter.

In the absence of this component, the default is that no restrictions on result-length need to be performed.

3) dfr-configuration-request (C): The dfr-configuration-request is specified to obtain information relating to which optional extension attribute sets and constraints the DFR supports.

In the absence of this component, the default is FALSE which indicates that no such request is being made.

4) bind-security (O): This specifies OSI security services required in the bind for example peer entity authentication of the software entities involved, or confidentiality, or integrity protection.

BindSecurity :: = EXTERNAL

5) priority (C): Priority requested for this DFR-User during the association. If there is no other priority specified in the abstract operation (see 8.1.3.3), this priority shall be applied. Medium priority is applied by default.

7.1.2 Bind-result Parameters

The **DfrBindResult** parameter returns any authentication-attributes needed.

The definition is:

```
DfrBindResult :: = SET {
  authentication-attributes          [0] SET OF AuthenticationAttribute,
  available-attribute-types          [1] SET OF AttributeType OPTIONAL,
  constraints-supported              [2] SET OF ConstraintsType OPTIONAL,
  dfr-document-types-supported       [3] SET OF OBJECT IDENTIFIER OPTIONAL,
  function-set-supported             [4] FunctionSetType OPTIONAL
  maximum-result-length-supported    [5] INTEGER OPTIONAL }
```

1) Authentication-attributes (C): Any information returned as confirmation of an authentication check. An information that is unconstrained by this Part 1 of this ECMA standard.

AuthenticationAttribute :: = EXTERNAL

2) Available-attribute-types (C): Specifies the attribute types defined in the optional extension attribute sets supported by the DFR. Only present if a DFR-configuration-request is made.

3) Constraints-supported (C): Specifies the set of all constraints defined for a DFR-Document-Store. Only present if a DFR-configuration-request is made.

For a DFR-Document-Store constraints may be specified:

a) Global unambiguity of DFR-Titles (Each DFR-Entry has a DFR-Title unique within the DS).

b) Local unambiguity of DFR-Titles (each DFR-Entry has a DFR-Title unique within its parent group).

c) Version Name unambiguity (Each Version of a Conceptual Document is uniquely identified by its Version name in the scope of its Conceptual Document).

```
ConstraintsType ::= ENUMERATED {
    global-unambiguity    (0),
    local-unambiguity     (1),
    version-unambiguity   (2) }
```

4) DFR-document-types-supported (C): Specifies a set of object-identifiers that define the document-types that the Document Store has knowledge of. Only present if a DFR-configuration-request is made.

5) Function-Set-supported (C): Specifies which Function Set of the three defined sets (Flat Store Set, Pre-defined Store Structure Set, Full Set) is supported. Only present if a DFR-configuration-request is made.

```
FunctionSetType ::= ENUMERATED {
    flat-store          (1),
    predefined-store    (2),
    full-set            (3) }
```

6) Maximum-result-length-supported (C): Specifies the maximum result length the server supports. This value might be lower or equal to the maximum result length specified by the DFR-User in the Retrieve-restrictions of the bind-operation. This result parameter is only present if a maximum result length was specified in the bind arguments.

7.1.3 Bind-error Parameters

The DFR-Server port may report either of two errors, **SecurityProblem** or **ServiceProblem**. **SecurityProblem** indicates that the identity of the DFR-User cannot be established on the basis of the information the DFR-User supplied, or that this DFR-User is not allowed to access this DS. ServiceProblem reports that the DFR-Server cannot establish the association die to some operational exception. The same errors may occur in DFR abstract operations, see 8.3.9.

```
DfrBindError ::= CHOICE {
    service-error     [0] ServiceProblem,
    security-error    [1] SecurityProblem }
```

7.2 Abstract-unbind Parameters

An abstract-unbind-operation closes the filing and retrieval port. The issuing of an abstract-unbind-operation results in the deletion of any retrieve-restrictions that were specified in the

abstract-bind-operation argument. There are no arguments or errors associated with the abstract-unbind-operation.

```
DfrUnbind :: = ABSTRACT-UNBIND
    FROM {dfr-port[s] }
```

8. ABSTRACT-OPERATIONS

All abstract operations are invoked by the consumers (DFR-Users) and each abstract operation always reports either success or an error.

8.1 Common Data-types used in Abstract-operations

8.1.1 Data-Types used for DFR-Object specification

This subclause identifies, and in some cases defines, a number of data-types which are subsequently used in the definition of Document Filing and Retrieval abstract operations. The data-types concerned are those which are common to more than one abstract operation, or are likely to be so in the future, or which are sufficiently complex or self-contained as to merit being defined separately from the abstract operation which uses them.

Some of these data-types are already defined in 6 of this part of the Standard. They are used for specification of different DFR-Objects contained in a DFR-Document-Store. These data-types are the following:

DfrObjectClass,	(see 6.3)
DfrEntry,	(see 6.3)
DfrEntryAttributes,	(see 6.3)
DfrObjectContent,	(see 6.3)
DfrUniquePermanentIdentifier,	(see 6.3)
DfrDocumentContent,	(see 6.3.2.1)
DfrInternalReferenceContent,	(see 6.3.3.1)
DfrExternalReferenceContent,	(see 6.3.3.1)
DfrGroupContent,	(see 6.3.4.1)
DfrSearchResultListContent.	(see 6.3.5.1)

8.1.2 Imported data-types

Some data-types used in 6 as well as in this clause are actually defined in other standards. These imported data-types are:

From the OSI Directory (ISO/IEC 9594-2):
Attribute,
AttributeType,
AttributeValue,
AttributeValueAssertion,
DistinguishedName.

From the OSI Directory (ISO/IEC 9594-3):
Filter,
FilterItem.

NOTE 23
These two data-types are however included in the complete Formal Definition of the DFR Abstract-service (Appendix C of this Standard), to facilitate their eventual further extensions.

From the DOAM (ISO/IEC 10031-2):
RDT-reference.

8.1.3 Data-types common for most DFR abstract operations

An argument of any DFR abstract operation is a sequence of "parameters" (or "arguments"), some of them being mandatory for the given abstract operation type, while others are optional. Each individual parameter has its own context-specific tag. Parameters eventually used in (almost) any DFR abstract operation have their tags at the end of the [0, 30] integer range, while more specific parameters have their tags at the beginning of this range. The common parameters are specified as follows:

```
CommonArguments ::= SEQUENCE {
        reservation     [27] Reservation OPTIONAL,
        error-handling  [28] ErrorHandlingMode DEFAULT all-or-nothing,
        priority        [29] Priority DEFAULT medium,
        privileges      [30] Privileges OPTIONAL }

CommonResults ::= SEQUENCE {
        warnings        [30] SEQUENCE OF Warning DEFAULT { } }
```

8.1.3.1 Reservation

The semantics of this parameter when it is specified in an abstract operation is the same as defined in the Reserve abstract operation (see 8.2.9). Omission of this parameter means no change in the reservation level of the DFR-Entry concerned. For each DFR abstract operation, the DFR-Entry to which a reservation applies is specified in the description of that abstract operation. If a change to unreserved, or to exclusive write from exclusive-access is requested it is always actioned as the last step in the execution of the abstract operation. If a change to exclusive-access, or to exclusive-write from unreserved is requested it is always actioned as the first step in the execution of the abstract operation. After a DFR-Entry is reserved by some user (either exclusive-write or exclusive-access) reservation requests of other users will be rejected. The reserved DFR-Entry is only re-available for a reservation by another user if it is set to unreserved by user who has reserved it. The DFR-Reserved-By attribute is used to indicate the identity of the DFR-User who reserved the DFR-Entry.

```
Reservation ::= ENUMERATED {
        unreserved      (0),
        exclusive-write (1)
        exclusive-access (2)
```

The reservation of DFR-Objects is handled according to the following rules:

a) exclusive-write

   1) If the reserved DFR-Object is a DFR-Document, a DFR-Reference or a DFR-Search-Result-List and the reservation level is exclusive-write:

      i) other user shall not **Delete** or **Modify** the DFR-Object (neither any attribute nor the content of the object):

ii) other user shall not move the DFR-Object if the **Move** abstract operation is applied directly to the reserved DFR-Object; but the DFR-Object may be implicitly moved, if the **Move** abstract operation is applied to an Ancestor of the DFR-Object;

iii) other user shall not use a reserved DFR-Search-Result-List in a **Search** abstract operation.

2) If the reserved DFR-Object is a DFR-Group and the reservation level is exclusive-write:

i) other user shall not **Delete** or **Modify** the DFR-Group (neither any attribute nor the content of the group);

ii) other user shall not move the DFR-Group if the **Move** abstract operation is applied directly to the reserved DFR-Group; but the DFR-Group may be implicitly moved if the **Move** abstract operation is applied to an unreserved Ancestor of this DFR-Group;

iii) other user shall not
- **Delete** or **Move** any Member from the DFR-Group,
- insert new Members into the DFR-Group by **Create** abstract operation;

vi) if a Member of the DFR-Group is itself a DFR-Group then the Descendants of this member DFR-Group are not reserved.

b) exclusive-access

1) If the reserved DFR-Object is a DFR-Document, a DFR-Reference or a DFR-Search-Result-List and the reservation level is exclusive-access:

i) other user shall not **Delete** or **Modify** the DFR-Object (neither any attribute nor the content of the object);

ii) other user shall not move the DFR-Object if the **Move** abstract operation is applied directly to the reserved DFR-Object; but the DFR-Object may be implicitly moved, if the **Move** abstract operation is applied to an Ancestor of this DFR-Object;

iii) other user shall not **Read** or **Copy** the content of the DFR-Object;

iv) other user shall not use a reserved DFR-Search-Result-List in a **Search** abstract operation;

v) other user shall not **List** the elements of a DFR-Search-Result-List.

2) If the reserved DFR-Object is a DFR-Group and the reservation level is exclusive-access:

i) other user shall not **Delete** or **Modify** the DFR-Group (neither any attribute nor the content of the group);

ii) other user shall not move the DFR-Group if the **Move** abstract operation is applied directly to the reserved DFR-Group; but the DFR-Group may be implicitly moved, if the **Move** abstract operation is applied to an Ancestor of this DFR-Group;

iii) other user shall not
- **Delete** or **Move** any Member from the DFR-Group,
- insert new Members into the DFR-Group by **Create** abstract operations,
- **Modify** a Member of the DFR-Group (neither any attribute nor the content of a Member) if it is a DFR-Document, a DFR-Reference or a DFR-Search-Result-List,
- **Modify** the attributes of a Member if it is a DFR-Group,
- **List** the Members of the DFR-Group,
- **Read** or **Copy** the content of a Member if it is a DFR-Document, a DFR-Reference or a DFR-Search-Result-List,
- **List** the DFR-Entries noted in the elements of a DFR-Search-Result-List which is a Member of the reserved DFR-Group,
- use a DFR-Search-Result-List which is a Member of the reserved DFR-Group in a **Search** abstract operation;

iv) if a Member of the DFR-Group is itself a DFR-Group then the Descendants of this member DFR-Group are not reserved.

8.1.3.2 Error handling mode

```
ErrorHandlingMode ::= CHOICE {
  all-or-nothing          [0] NULL,
  until-first-warning     [1] NULL,
  report-all-warnings     [2] NULL,
  report-n-warnings       [3] INTEGER }
  -- This parameter is only applicable to multiple abstract operations, such as List or Move --
  -- group.--

Warning ::= SEQUENCE {
  entry       [0] DfrEntryName OPTIONAL,
                    -- never reported if AccessProblem is "insufficient-access-rights" --
  access1     [1] AccessProblem } -- see 8.3.3 --
  access2     [2] ReferentAccessProblem
```

When several DFR-Entries need to be accessed and/or updated during the execution of a DFR abstract operation, this may lead to different situations, in relation to the requested access of the requesting DFR-User to these DFR-Entries:

1) none of the DFR-Entries involved may in fact be accessed (updated);

2) all the involved entries may be accessed (updated);

3) some entries may be accessed, while others may not.

The first situation is that of a DFR-error (of access type), and the second that of success (if no other error occurs). The third situation is more complex. For example, when listing a DFR-Group some members of which are accessible to the user while others are not, the DFR-User may however want all accessible members listed, with a warning(s) about inaccessible Members. This is the "report-all-warnings" error-handling mode. The alternative "report-n-warnings" is similar to "report-all-warnings" but it specifies the maximum number of warnings to be returned; zero means continuing processing without any warning reported. Inaccessible Members will not be reported if the reason is "insuffi-

cientAccessRights", but will be reported if the reason is "reservedByAnotherUser". The DFR service definition leaves it to the DFR-User to specify the error-handling mode desired on a per abstract operation basis.

The error handling mode "all-or-nothing" means that the abstract operation will proceed to its conclusion unless any error or warning is detected, in which case the abstract operation will have no effect whatever on the DFR-Document-Store.

Paired with this common argument is the "warnings" common result. It takes the form of a sequence of access-type errors (one per DFR-Entry concerned but not accessible), and may appear in an abstract operation result provided that the abstract operation argument had **ErrorHandlingMode** specified as "until-first-warning" or "report-all-warnings" or "report-n-warnings".

Abstract operations concerned are: **List** and **Copy** when applied to a DFR-Group. Eventual problems related to the modifications parameter of an abstract operation (see 8.1.5.5) are always handled as if the "all-or-nothing" mode were specified.

8.1.3.3 Priority

```
Priority :: = ENUMERATED {
        low      (0),
        medium   (1),
        high     (2) }
```

This common parameter may be specified for any DFR abstract operation. It is useful in the context of a heavily loaded DFR-Server, to allow better service to some privileged requests according to an applied priority policy. A default **Priority** may already have been established at bind time (see 7.1.1); any **Priority** requested here will overwrite the **Priority** given at bind time.

This concept of operational priority is not necessarily related to any communication priority policy. This means that specification of this parameter may, but need not, influence the quality of service of the underlying communication layers.

The DFR-Server does not mandatorily grant the abstract operation with the priority requested. Any numeric characterization of the above three priority levels is server-specific.

8.1.3.4 Privileges

```
Privileges :: = SEQUENCE {
   operation-pac      [0] PrivilegeAttributeCertificate OPTIONAL,
   proxy-pac          [1] PrivilegeAttributeCertificate OPTIONAL }
```

For the definition of **PrivilegeAttributeCertificate** see 7.1.1.

The **Privilege** parameter enables a **PrivilegeAttributeCertificate** to be associated directly with an abstract operation request, modifying that available from the abstract bind within which the request is being made. The way the operation-pac modifies the bind PAC depends on the security policy applied. The resulting privileges are only applied for thisabstract operation.

The operation-pac may be required for either of two reasons:

a) When the access privileges established by the user during the bind are not sufficient to permit the requested abstract operation.

b) When the bind is being multiplexed between a number of users and each abstract operation is potentially under the control of a different user, who must present his own access privileges.

The proxy-pac may be required if the DFR-Server shall make further access to another application on behalf of the user, and itself would have insufficient access rights unless supplemented by those in the proxy-pac. The DFR-Server that receives the proxy-pac itself then uses this as an operation-pac to the other application.

8.1.4 Access names for DFR-Entries

```
DfrEntryName :: = CHOICE {
  Upi                [0] DfrUniquePermanentIdentifier,
  path-name          [1] DfrPathName,
  relative-path-name [2] SEQUENCE {
                       base [0] DfrUniquePermanentIdentifier,
                       path [1] DfrPathName } }

DfrPathName :: = SEQUENCE OF DfrTitle

DfrTitle :: = CharacterData

CharacterData :: = CHOICE { GraphicString, T61String, PrintableString }
```

A DFR-Entry, and thus the DFR-Object it represents in the DS may always be accessed using the DFR-Unique-Permanent-Identifier (UPI) attribute of the object provided the DFR-User has been granted sufficient access permission. This is the primary DFR-Object identification mechanism. Other possibilities are also offered, which are more "semantical", but less universal, in the sense that they may sometimes not lead to the same DFR-Entry.

A path-name is a sequence of values of the DFR-Title attribute of all Ancestors of the DFR-Entry, in descending order. The **DfrPathName** guarantees an unambiguous identification of the DFR-Entry only if the DFR-Title of each Ancestor of the DFR-Entry is locally unambiguous (see 7.1.2).

A relative-path-name is a **DfrPathName** beginning not from the root DFR-Group of the DS, but from some specified intermediate DFR-Group. It may be useful to identify some Descendants of this DFR-Group, the latter specifying some "working domain".

8.1.5 Data-types common for operations on single entries

```
CommonUpdateArguments :: = SEQUENCE {
  object-class       [0] DfrObjectClass OPTIONAL,
  entry              [1] CHOICE { local DfrEntryName,
                                  external [3] ExternalReferenceContent } OPTIONAL,
  destination        [2] DfrEntryName OPTIONAL, --of the parent group--
  position           [3] GroupMemberPosition OPTIONAL, --in the parent group--
  modifications      [4] SEQUENCE OF EntryModification OPTIONAL,
  selection          [5] EntryInformationSelection OPTIONAL }

CommonUpdateResult :: = SEQUENCE {
  Upi                [0] DfrUniquePermanentIdentifier,
  entry-information  [1] EntryInformation OPTIONAL,
  COMPONENTS OF CommonResults }
```

DFR abstract operations concerned are: **Create, Copy, Move, Read,** and **Modify**. Not all of the above parameters may be specified in each of these abstract operations. Details are given hereafter, as well as in the paragraphs defining individual abstract operation types.

8.1.5.1 Object-Class

This common parameter specifies the **DfrObjectClass** of the **DFR-Entry** concerned in the abstract operation.

This parameter shall be present in a **Create** abstract operation, to specify the **DfrObjectClass** of a DFR-Object to be created; it is optional in all other cases, and if specified is used for validation only.

8.1.5.2 Entry

This parameter specifies the **DfrEntryName** of the DFR-Entry to be copied, moved, read or modified. It is mandatorily present in all these cases, and absent in the case of a **Create** abstract operation.

8.1.5.3 Destination

This parameter specifies the **DfrEntryName** of the DFR-Group where the specified object (in a **Create** or **Move** abstract operation) or its copy (in a **Copy** abstract operation) is to be placed. It is mandatorily present in these three abstract operations, and absent in the case of a **Read** or **Modify** abstract operation.

8.1.5.4 Position

```
GroupMemberPosition :: = CHOICE {
        last      [0] NULL,
        first     [1] NULL,
        after     [2] DfrEntryName,
        before    [3] DfrEntryName }
```

This optional parameter may be specified only if the destination parameter is specified (i.e. in a **Create, Copy** or **Move** abstract operation). It specifies the place in the parent DFR-Group where the newly created DFR-Entry is to be put. It cannot be specified if the parent DFR-Group has its DFR-Ordering attribute specified (in which case the DFR-Server automatically puts the new entry at the appropriate logical position, according to the ordering rule specified). This parameter may specify either absolute (first, last) or relative (before, after) position.

8.1.5.5 Modifications

```
EntryModification ::= CHOICE{
    put-attribute           [0] Attribute,
    remove-attribute        [1] AttributeType,
    copy-attributes-from    [2] SEQUENCE {
            source              [0] SourceEntry,
            attribute-selection [1] SET OF AttributeType OPTIONAL },
                --default means all copyable attributes--
    add-values              [3] Attribute,
    remove-values           [4] Attribute,
    add-values-from         [5] SEQUENCE {
            source              [0] SourceEntry,
            attribute-selection [1] SET OF AttributeType OPTIONAL },
                --default means all multivalued attributes--
    put-content             [6] DfrObjectContent,
    remove-content          [7] NULL,
    copy-content-from       [8] SourceEntry }
```

The **EntryModifications** parameter specifies updates to a DFR-Entry as a sequence of **EntryModifications**, to be applied in the order in which they are specified. It is the only means to specify updates in a **Modify** abstract operation (where it is mandatory), or in a **Copy** or **Move** abstract operation (where it is optional). As for the **Create** abstract operation, two methods are available for specifying different items of the newly created DFR-Entry: either by the **EntryModifications** parameter (as a sequence of changes applied to the minimal (server-defined) attribute set and an empty content), or by directly specifying the items in the abstract operation.

An **EntryModification** may apply to an attribute of a specified type, or to the entire content of the DFR-Entry to be updated. An entire attribute may be "put" (possibly deleting the previous values of the attribute), or "removed" (all values are removed, and the attribute becomes absent), or else, "copied" from some other DFR-Entry (**SourceEntry**).

Furthermore, individual values of a multi-valued attribute may be added to its existing values, or removed from the list of the existing values. In the remove case, the values to be removed shall be explicitly specified in the abstract operation; in the add case they may be specified either explicitly in the abstract operation or implicitly (add-values-from") by specifying another **SourceEntry** and Attribute-Types to be taken from it.

Finally, the entire DFR-Content of the DFR-Entry to be updated may be "put" (specified directly in the abstract operation), or "removed", or "copied" from some other specified DFR-Entry (**SourceEntry**). In the latter case, some attributes of the source DFR-Entry, which are closely related to the content, are automatically copied by the DFR-Server (see Table 3). The DFR-Content of a DFR-Search-Result-List may not be modified. This may only be changed by directly using the **Search** abstract operation.

When copying the content of another DFR-Entry, the DFR-Object-Class of the source and of the receiving DFR-Object (the sink) must be the same, with the following unique exception:

When creating or modifying a DFR-Internal-Reference, its content may be specified as being copied from some other **SourceEntry**. If the source DFR-Entry is a DFR-Internal-

Reference, its content is copied in a straightforward manner. But if the **SourceEntry** is a DFR-Document, DFR-Group, or a DFR-Search-Result-List, the DFR-Server creates a DFR-Internal-Reference-Content in the form of an RDT-reference to the source DFR-Entry, which thus becomes the Referent of the DFR-Internal-Reference created/modified.

8.1.5.5.1 SourceEntry

```
SourceEntry ::= CHOICE {
    parent              [0] NULL,
    referent            [1] NULL,    -- only for a DFR-Reference --
    previous-version    [2] NULL,    -- only if unique previous version --
    specified-entry     [3] DfrEntryName,
    reference           [4] RDT-reference }
```

A **SourceEntry**, used in some cases of "modifications" as the source of an attribute and/or the content of the DFR-Object to be created/updated, may be the parent DFR-Group of the latter, or its previous version (only in the case where the updated DFR-Object has a unique previous version), its Referent (for DFR-References only; this may involve the use of the RDT for transfer of attributes or content or both from the Referent to the DFR-Reference in the case of a DFR-External-Reference), or, some explicitly specified source DFR-Entry.

8.1.5.6 Selecting entry information for reading

```
EntryInformationSelection ::= SEQUENCE {
  read-selector             [0] ENUMERATED {
      attributes-only                 (0),
      attributes-and-content          (1),
      content-only                    (2),
      rdt-ref-to-attr-only            (3),
      attr-and-rdt-ref-to-content     (4),
      rdt-ref-to-content-only         (5),
      rdt-ref-to-entire-object        (6),
      attr-and-rdt-ref-to-entire-object  (7) }
        DEFAULT attributes-only,
  attribute-selection       [1] AttributeSelection OPTIONAL }
```

For the **EntryInformationSelection** parameter the following rules apply: by default, all attributes are requested, if read-selector is 0, 1, 4 or 7. For other values of read-selector, attribute-selection shall not be specified. Empty selection shall be specified as {}.

```
AttributeSelection ::= CHOICE {
  all           [0] NULL,
  none          [1] NULL,
  unordered     [2] SET OF AttributeType, --when the delivery order is insignificant--
  ordered       [3] SEQUENCE OF AttributeType, --to specify not only attributes to be --
                -- delivered, but also the order in which they should appear --
  minimum       [4] NULL } -- implicitly selects UPI and DFR-object-class attributes --
                -- (other predefined selections are for further study ) --
```

```
EntryInformation ::= CHOICE {
    attributes-only                 [0] DfrEntryAttributes,
    attributes-and-content          [1] DfrObject,
    content-only                    [2] DfrObjectContent,
    rdt-ref-to-attr-only            [3] RDT-reference,
    attr-and-rdt-ref-to-content     [4] SEQUENCE {
            attributes                  [0] DfrEntryAttributes,
            rdt-ref-to-content          [1] RDT-reference },
    rdt-ref-to-content-only         [5] RDT-reference,
    rdt-ref-to-entire-object        [6] RDT-reference,
    attr-and-rdt-ref-to-entire-object [7] SEQUENCE {
            attributes                  [0] DfrEntryAttributes,
            rdt-ref-to-entire-object    [1] RDT-reference     } }
```

Any of the five abstract operations considered (namely **Create, Move, Copy, Read** and **Modify**) may specify those items from the involved DFR-Entry which are to be read, that is, returned by the DFR-Server in the result of the abstract operation. This is the only purpose of the **Read** abstract operation, but is also interesting in the other four abstract operations as a control "readback" when some modified items have been copied from other entries, and not specified explicitly. Thus, the "selection" parameter is mandatory in the case of the **Read** abstract operation and optional in other cases.

The **EntryInformation** component of the abstract operation's result is present if and only if the selection parameter was specified in the abstract operation's argument, and it consists of exactly those items which were requested in the selection parameter, and which are present in the DFR-Entry to be read.

The **EntryInformationSelection** data-type is a sequence with two components: read-selector and attribute-selection. The read-selector specifies whether an RDT-reference or some values are required and whether the reference or values pertain to the entire DFR-Object; its content or its attributes. This gives a number of combinations which are explicitly named by their ASN.1 identifiers above. The attribute-selection component optionally indicates, where appropriate, which attributes of the DFR-Entry are to be returned. The absence from the DFR-Entry of any attribute selected causes no data to be returned for that attribute nor any error generated.

The RDT-reference produced and returned by the DFR-Server is constructed differently, and is intended for different purposes, according to whether it is an RDT-reference to an entire DFR-Object, only the DFR-Object's attributes, or only the DFR-Object's content (the latter is possible only for a DFR-Document). Details are given in 6.3.3.

8.1.6 Data-types common for operations on multiple entries

Abstract Operations concerned are **List** and **Search**. Also concerned is the data-type **DfrSearchResultListContent**, specified in 6.3.5.1.

```
CommonListSearchArguments ::= SEQUENCE {
    continuation    [1] ContinuationContext OPTIONAL,
    limits          [2] Limits OPTIONAL,
    selection       [3] AttributeSelection OPTIONAL,
    ordering        [4] OrderingRule OPTIONAL }
```

```
CommonListSearchResult ::= SEQUENCE {
    number-of-entries    [0] INTEGER,
    continuation         [1] ContinuationContext OPTIONAL,
    limit-encountered    [2] LimitEncountered OPTIONAL,
    entry-list           [3] DfrEntryList,
    COMPONENTS OF CommonResults }
```

8.1.6.1 Limits and continuation

```
ContinuationContext ::= OCTET STRING -- DFR-Server implementation specific --

Limits ::= SEQUENCE {
    time-limit  [0] INTEGER OPTIONAL,
    count-limit [1] INTEGER OPTIONAL }

LimitEncountered ::= ENUMERATED {
    time-limit      (0),
    count-limit     (1),
    length-exceded  (2) } -- maximum result length as specified during Binding exceeded --
```

When initiating a **List** or a **Search** abstract operation, the DFR-User may specify a maximum number of entries to be found/returned, a maximum execution time (for the **Search** abstract operation only), or both. The result returned in this case contains an indication whether the time-limit, count-limit or maximum result length has been reached. In the case when the abstract operation has successfully finished without encountering any limit, the **LimitEncountered** parameter is absent from the abstract operation result. If, however, a limit has been encountered, then the server communicates to the user a continuation parameter, which cannot be interpreted by the DFR-User, but may be sent to the DFR-Server when resuming "the same" abstract operation (immediately or later), to instruct the server not to start the abstract operation "from the beginning", but to continue it from where it has previously stopped. This **ContinuationContext** also forms a part of a DFR-Search-Result-List-Content. If the result of a **Search** operation is not stored by the DFR-Server in a DFR-Search-Result-List then the option to continue the **Search** is withdrawn at unbind time or on abortion.

8.1.6.2 Information selected and returned

```
DfrEntryList ::= SEQUENCE OF SEQUENCE {
    UniquePermanentIdentifier [0] DfrUniquePermanentIdentifier,
    class                     [1] DfrObjectClass,
    ordering-attribute        [2] Attribute OPTIONAL,
    other-attributes          [3] SEQUENCE OF Attribute OPTIONAL }
```

This is the information about the DFR-Entries listed or found. Only attributes of entries may be returned, and the selection of attributes to be returned for each entry concerned is specified by the selection argument parameter. (The **AttributeSelection** data-type is specified in 8.1.5.6).

For each DFR-Entry listed, the UPI of the corresponding DFR-Object is always returned, regardless of the **AttributeSelection** specified in the abstract operation. The same applies to the DFR-Object-Class of the DFR-Object and to its ordering-attribute (see next paragraph). All other attributes are returned only if their types have been specified in the

AttributeSelection in the abstract operation argument. In contrast, these other-attributes are never stored in a DFR-Search-Result-List-Content.

8.1.6.3 Ordering of entries in a list

```
OrderingRule :: = CHOICE {
   ascending     [0] AttributeType,
   descending    [1] AttributeType }
```

The user may specify in the argument of a **List** or **Search** abstract operation (in the ordering parameter) the order in which different entries are to be listed. An **OrderingRule** specifies the **AttributeType** to be used as an ordering key, and the ordering direction (ascending or descending). If such an ordering rule has been specified, each entry concerned is listed together with the value of its ordering-attribute; if, however, no explicit ordering has been specified, then the corresponding parameter is absent from the abstract operation result. The **OrderingRule** is retained with the **DfrSearchResultList**, if the operation requests the results are to be placed in one. An **OrderingRule** may only be applied for an attribute type having a MATCHES FOR ORDERING clause in its definition.

The lexicographic ordering of attributes is discussed in 6.3.7.

8.1.6.4 Search domain

```
SearchDomain :: = SEQUENCE OF CHOICE {
      previous-result  [0] DfrEntryName, --specifies an entry of "SearchResultList" class--
      scope            [1] SEQUENCE {
            root                    [0] DfrEntryName,
            descent-depth           [1] INTEGER OPTIONAL,
            -- default means the whole subtree --
            dereferencing-depth     [2] INTEGER DEFAULT 0 }
            -- default means no dereferencing --  }
```

The **SearchDomain** parameter is specified in a **Search** abstract operation and is then possibly stored in the corresponding DFR-Search-Result-List-Content, from where it can be then read by the DFR-Server to continue "the same" abstract operation. A search domain is specified as a sequence of "subdomains", each being either a DFR-Search-Result-List (containing results of some previous **Search** abstract operation), or a DFR-Group, designating the starting point of the **Search** abstract operation. In the latter case, the depth of descendants to be searched may be limited (by the descent-depth parameter), as well as the depth of dereferencing (that is, the number of times the server may indirectly proceed from a DFR-Reference to its referent when executing this search), e.g. from a DFR-Reference to its Referent which is a DFR-Group (first step), then from the latter to one of its Members which is another DFR-Reference, then from the latter to its Referent (second step) and so on.

8.1.6.5 SearchCriteria

```
SearchCriteria :: = Filter
```

The **SearchCriteria** parameter is specified in a Search abstract operation and is then possibly stored in the corresponding DFR-Search-Result-List-Content (for the same purpose as the **SearchDomain**). This parameter is specified as a **Filter**; it applies to any DFR-Entry within the **SearchDomain** specified. In the case where the application of the **Filter** to such a DFR-Entry gives TRUE, the DFR-Entry is qualified for inclusion in the **CommonListSearchResult** or in the **DfrSearchResultList** or in both).

8.2 Document Filing and Retrieval Port Operations Definitions

The following abstract operations are available at the Document Filing and Retrieval Port. The abstract operations are performed within a DFR-Document-Store.

Table 1 provides an overview of which parts of the DFR-Entry an abstract operation is related to.

Operations related to -->	DFR Group		DFR Document		DFR Reference		DFR-Search-Result-List	
	Attri-bute	Con-tent	Attri-bute	Con-tent	Attri-bute	Con-tent	Attri-bute	Con-tent
Create	X		X	X	X	X	X	
Delete	X	X	X	X	X	X	X	X
Copy	X	X	X	X	X	X	X	X
Move	X	X	X	X	X	X	X	X
Read	X		X	X	X	X	X	X
Modify	X		X	X	X	X	X	
List		X						X
Search	X		X		X		X	X
Reserve	X	X	X	X	X	X	X	X
Abandon								

Table 1 - Overview of applicability of DFR abstract operations to DFR object classes

To perform a DFR abstract operation the DFR-User is required to have sufficient access rights to all DFR-Entries involved in the abstract operation. This is summarized in Table 2.

Operations Related to -->	Target DFR-Entry	Source DFR-Entry	Target Parent DFR-Entry	Source Parent DFR-Entry	Sources of Modification
Create	(O)	N	RM	N	R
Delete	RMD(1)	N	N	N	N
Copy	(O)	R	RM	N	R
Move	N	RMD(1)	RM	RM	R
Read	R	N	N	N	N
Modify	RM	N	N	N	R
List	R	R	N	N	N
Search	RM	R	N	N	N
Reserve	RM	N	N	N	N
Abandon	N	N	N	N	N

Table 2 - Overview of access rights to DFR-Entries required from a DFR-User to perform a DFR abstract operation

Notes to Table 2

O	owner access right, it is automatically granted to the DFR-User creating this DFR-Object in a **Create** or **Copy** abstract operation.
R	read access right.
RM	read-modify access right.
RMD	read-modify-delete access right.
N	Non applicable.
Target DFR-Entry	The "target" DFR-Entry specified in the abstract operation. In a **List** abstract operation it is the DFR-Group to be listed; in a **Search** abstract operation it is the DFR-Search-Result-List where the result is to be placed.
Source DFR-Entry	The "source" DFR-Entry specified in the abstract operation. In a **List** abstract operation all DFR-Group-Members to be listed; in a **Search** abstract operation all DFR-Entries satisfying the search criteria.
Target Parent DFR-Entry	The DFR-Entry specified in the abstract operation that is the Parent of the "target" DFR-Entry or of the DFR-Entity to be created.
Source Parent DFR-Entry	The Parent of the "source" DFR-Entry.
Sources of Modification	A DFR-Entry specified in the abstract operation from which attributes or the content are to be copied.
(1)	It is also possible to move or delete a DFR-Entry for which the requestor has read-modify-delete access right to an Ancestor, regardless of the requestors access rights to the DFR-Entry to be moved or deleted.

8.2.1 Create

The **Create** abstract operation places a new DFR-Object in a DFR-Group. It creates a new DFR-Entry for this DFR-Object. The class of the new DFR-Object is defined by the object-class argument. When a DFR-Group is created the group membership criteria for future group Members may be defined. Consequential changes to the Parent group attributes are made. The DFR-Content of the DFR-Object is provided in this abstract operation optionally in the case of a DFR-Document, mandatorily in the case of a DFR-Reference, and is not permitted for a DFR-Group or a DFR-Search-Result-List. If a newly created document is declared as a new version (attribute DFR-Previous-Versions supplied), consequential changes are made to the DFR-Next-Versions attribute of each of the Documents specified as previous versions.

Whenever the security policy defined in this standard is applied: the DFR-Access-list attribute for the newly created DFR-Object is specified by the creating DFR-User or taken by default. The default value contains only the creating DFR-User as the Owner. If the DFR-Access-List is specified explicitly but contains no Owner then the DFR-Server adds to it the creating DFR-User as the sole Owner.

```
Create :: = ABSTRACT-OPERATION
  ARGUMENT CreateArgument
  RESULT     CreateResult
  ERRORS {
        Abandoned,
        NameError,
        UpdateError,
        AttributeError,
        VersionManagementError,
        AccessError,
        ReferentAccessError,
        InterServerAccessError,
        SecurityError,
        ServiceError}
```

8.2.1.1 Create-argument

```
CreateArgument :: = SEQUENCE {
  COMPONENTS OF
      CommonUpdateArguments (WITH COMPONENTS { ... ,
        object-class PRESENT,
        destination     PRESENT } ),
  attributes  [7] SET OF Attributes OPTIONAL,
  content     [8] DfrObjectContent OPTIONAL,
  COMPONENTS OF CommonArguments (WITH COMPONENTS {
        ..., error-handling ABSENT })}
```

The components of **CreateArgument** have the following meaning:

a) Common update arguments:
- object-class specifies the **DfrObjectClass** of a DFR-Entry to be created;
- entry optionally gives the **DfrEntryName** of (or an RDT reference to) the DFR-Object which will be copied to the DFR-Entry created;
- destination gives the **DfrEntryName** of the parent DFR-Group where the DFR-Entry is to be placed;
- position optionally specifies at what place the DFR-Entry is to be put in the parent DFR-Group;
- modifications optionally specifies some attributes and/or content of the new entry, especially when the values are not provided in the **Create** argument, but are to be copied from other DFR-Entries;
- selection optionally specifies which information from the created DFR-Entry is to be read back to the requestor (in the **CreateResult**) after creation is done.

b) Specific create arguments:
- attributes is the primary means to provide attributes for the DFR-Entry to be created (alternative means is modifications argument above; both may be used in the same **Create** abstract operation, in which case attributes applies first);
- content is the primary means to provide a content for the DFR-Entry to be created (alternative means is modifications argument; they may not be used simultaneously in the same **Create** abstract operation);

If several alternative parameters are specified then entry is applied first, subsequently followed by attributes, content and modifications.

c) Common arguments:
- reservation, if requested, applies to the newly created DFR-Entry;
- error-handling, always taken by default (all-or-nothig is used);
- priority, see 8.1.3.3;
- privileges, see 8.1.3.4.

8.2.1.2 Create-result

Should the request succeed, the **CreateResult** will be returned.

```
CreateResult :: = CommonUpdateResult
```

The components of the **CreateResult** have the following meaning:

- upi is the **DfrUniquePermanentIdentifier** assigned by the DFR-Server to the newly created DFR-Entry;
- entry-information returns all those items from the new DFR-Entry (attributes and/or content) which have been requested by the selection component of the **CreateArgument**, and which are present in the DFR-Entry;
- warnings are never returned.

8.2.1.3 Create Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3.

8.2.2 Delete

The **Delete** abstract operation deletes a DFR-Entry from the parent DFR-Group and thus from the DFR-Document-Store. Consequential changes to the parent group attributes are made. The UPI of the specified DFR-Object ceases to be valid. If a DFR-Group is deleted the group and all its Descendants are deleted regardless of the requestor's access rights to these Descendants. If a DFR-Reference is deleted the Referent is not affected. If the DFR-Document to be deleted is a version, then consequential changes are made to the related attributes in all its previous and next versions. If the DFR-Object to be deleted, or any of its Descendents, is reserved by another DFR-User then the delete request fails.

NOTE 24
To delete a DFR-Object, the requesting DFR-User must have read-modify-delete access right to this DFR-Object or to one of its Ancestors.

```
Delete :: = ABSTRACT-OPERATION
  ARGUMENT    DeleteArgument
  RESULT      DeleteResult
  ERRORS {
    Abandoned,
    NameError,
    UpdateError,
    AccessError,
    SecurityError,
    ServiceError}
```

8.2.2.1 Delete-argument

```
DeleteArgument :: = SEQUENCE {
  entry          [0] DfrEntryName,
  COMPONENTS OF CommonArguments ( WITH COMPONENTS { ...,
                                    reservation ABSENT,
                                    error-handling ABSENT})}
```

The components of **DeleteArgument** have the following meanings:

a) Specific delete argument:
- entry identifies the DFR-Entry to be deleted.

b) Common arguments:
- error-handling, always taken by default (all-or-nothing is used);
- priority, see 8.1.3.3;
- privileges, see 8.1.3.4.

8.2.2.2 Delete-result

Should the request succeed, the **DeleteResult** will be returned. There are no parameters.

```
DeleteResult :: = NULL
```

8.2.2.3 Delete Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3.

8.2.3 Copy

The **Copy** abstract operation copies a DFR-Object from one DFR-Group to another DFR-Group (destination group). It creates a new DFR-Entry in the destination DFR-Group and copies the original DFR-Object's attributes (subject to changes explicitly specified in the abstract operation). For a DFR-Document, a DFR-Reference or a DFR-Search-Result-List, the DFR-Content is copied from the original to the copy. In the case of a DFR-Group and if an appropriate **ErrorHandlingMode** is specified, all Descendants to which the initiating DFR-User has at least read access permission (a subset of the DFR-Object-Tree) and which are not currently reserved by another DFR-User are copied. Modifications are applied to the newly created DFR-Entry if requested by the DFR-User (see 8.2.3.1). Some attribute modifications are also automatically applied by the DFR-Server (see 9). Consequential changes to the attributes of the destination group are made. The newly created DFR-Objects are assigned new UPIs. All existing DFR-References to the original DFR-Object(s) (and to its Descendants in the case of a DFR-Group) remain valid and do not refer to the new DFR-Object(s) (the Copy). This abstract operation may not be used to copy a DFR-Group from another DFR-document-store.

Whenever the security policy defined in this standard is applied: the DFR-Access-List for the newly created DFR-Object (the copy) and for all the newly created descendants is not taken from the original(s) but supplied explicitly by the requesting DFR-User or defaulted. The default value contains only the requesting DFR-User as the Owner. If the DFR-Access-List is specified explicitly, but contains no Owner, then the DFR-Server adds to it the requesting DFR-User as the sole Owner. When copying a DFR-Group, this resulting DFR-Access-List is applied to the newly created DFR-Group and to all its Descendants.

```
Copy :: = ABSTRACT-OPERATION
  ARGUMENT     CopyArgument
  RESULT       CopyResult
  ERRORS {
    Abandoned,
    NameError,
    UpdateError,
    AttributeError,
    VersionManagementError,
    AccessError,
    ReferentAccessError,
    InterServerAccessError,
    SecurityError,
    ServiceError}
```

8.2.3.1 Copy-argument

```
CopyArgument :: = SEQUENCE {
  COMPONENTS OF
      CommonUpdateArguments (WITH COMPONENTS { ... ,
          entry        PRESENT,
          destination  PRESENT } ),
        COMPONENTS OF CommonArguments }
```

The components of **CopyArgument** have the following meanings:

a) Common update arguments:
- object-class optionally specifies the **DfrObjectClass** of the original DFR-Entry;
- entry gives the **DfrEntryName** of (or an RDT-reference to) the original DFR-Entry;
- destination gives the **DfrEntryName** of the parent DFR-Group where the copy is to be placed;
- position optionally specifies at what place the copied DFR-Entry is to be put in its parent DFR-Group;
- modifications optionally specifies some modifications to the attributes and/or content of the new DFR-Entry (the copy), as compared to the original; no modifications will be applied to any Descendents copied (with the exception of DFR-Access-List);
- selection optionally specifies which information from the created DFR-Entry (the copy) is to be read back to the requestor (in the Copy-result) after copying is done.

b) Common arguments:
- reservation, if requested, applies to the newly created DFR-Entry (but not to its Descendents);
- error-handling, any of four modes may be specified (see 8.1.3.2) when copying a DFR-Group; in this case when specifying until-first-warning mode the result is DFR-Server implementation-dependent. When copying a single DFR-Object, only the all-or-nothing mode applies (taken by default);
- priority, see 8.1.3.3;
- privileges, see 8.1.3.4

8.2.3.2 Copy-result

Should the request succeed, the **CopyResult** will be returned.

CopyResult :: = CommonUpdateResult

The components of the **CopyResult** have the following meaning:

- upi is the **DfrUniquePermanentIdentifier** assigned by the DFR-Server to the newly created DFR-Entry (the copy);
- entry-information returns all those items from the new DFR-Entry (attributes and/or content) which have been requested by the selection component of the **CopyArgument** and which are present in the new DFR-Entry;
- warnings if returned identify those Descendants of the source DFR-Entry which are not copied because of insufficient access rights or reservation.

8.2.3.3 Copy Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3 .

8.2.4 Move

The **Move** abstract operation moves a DFR-Object from a source DFR-Group to a destination DFR-Group. A new DFR-Entry is created for this DFR-Object in the destination DFR-Group. The DFR-Entry in the source group is deleted. This DFR-Object ceases to be a Member of the source group and becomes a Member of the destination group. Modifications are applied to the moved DFR-Object if requested by the DFR-User (see 8.2.3.1). Some attribute modifications are also automatically applied by the DFR-Server (see 9). Consequential changes to the attributes of the source group and to the attributes of the destination group are made. The UPI attribute and all existing references to the moved DFR-Object remain valid. The DFR-Access-List attribute remains unchanged unless explicitly modified. When a DFR-Group is moved its Descendents are moved with it regardless of the requesting DFR-User's access rights to each of them, or of reservations made by other DFR-Users. This abstract operation may not be used to move a DFR-Object from another DFR-Document-Store.

NOTE 25
To move a DFR-Object the requesting DFR-User must have read-modify-delete access rights either to this DFR-Object or to one of its Ancestors.

```
Move :: = ABSTRACT-OPERATION
  ARGUMENT        MoveArgument
  RESULT          MoveResult
  ERRORS {
      Abandoned,
      NameError,
      UpdateError,
      AttributeError,
      VersionManagementError,
      AccessError,
      ReferentAccessError,
      InterServerAccessError,
      SecurityError,
      ServiceError}
```

8.2.4.1 Move-argument

```
MoveArgument :: = SEQUENCE {
        COMPONENTS OF CommonUpdateArguments (WITH COMPONENTS
            {..., entry PRESENT, destination PRESENT }),
              COMPONENTS OF CommonArguments (WITH COMPONENTS
            {..., error-handling ABSENT })}
```

The components of **MoveArgument** have the following meanings:

a) Common update arguments:
- object-class optionally specifies the **DfrObjectClass** of the DFR-Object to be moved;
- entry gives the **DfrEntryName** of the DFR-Object to be moved (external alternative is not applicable);
- destination gives the **DfrEntryName** of the new parent DFR-Group (where the DFR-Object is to be moved);
- position optionally specifies at what place the DFR-Object is to be put in its new parent DFR-Group;
- modifications optionally specifies some modifications to the attributes and/or content of the new DFR-Entry (of the DFR-Object moved), as compared to the old DFR-Entry of this DFR-Object;
- selection optionally specifies which information from the new DFR-Entry is to be read back to the requestor (in the move-result) after moving is done.

b) Common arguments:
- reservation, if requested, applies to the moved object in its new position (by default the previously defined reservation remains in force); reservations applied to any Descendant of the moved object are not changed;
- error-handling, always taken by default (all-or-nothing is used);
- priority, see 8.1.3.3;
- privileges, see 8.1.3.4

8.2.4.2 Move-result

Should the request succeed, the **MoveResult** will be returned.

```
MoveResult :: = CommonUpdateResult
```

The components of the **MoveResult** have the following meaning:

- upi is the **DfrUniquePermanentIdentifier** of the newly created DFR-Entry (which is equal to the UPI of the previous entry of the DFR-Object moved);
- entry-information returns all those items from the new DFR-Entry (attributes and/or content) which have been requested by the selection component of the move-argument, and which are present in the DFR-.Entry;
- warnings are never returned.

8.2.4.3 Move Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3 .

8.2.5 Read

The **Read** abstract operation returns attributes and/or content of a DFR-Entry. Some or all attributes may be read. The content if requested is returned entirely, the content of a DFR-

Group shall not be requested. On request attributes and content of the Referent (DFR-Group or DFR-Document) can be accessed via the DFR-Reference(with the same restriction as above).

```
Read ::= ABSTRACT-OPERATION
  ARGUMENT ReadArgument
  RESULT     ReadResult
  ERRORS {
    Abandoned,
    NameError,
    AccessError,
    ReferentAccessError,
    InterServerAccessError,
    SecurityError,
    ServiceError}
```

8.2.5.1 Read-argument

```
ReadArgument ::= SEQUENCE {
  COMPONENTS OF
      CommonUpdateArguments (WITH COMPONENTS
          { entry, selection } ),
        dereferencing      [7] BOOLEAN DEFAULT FALSE,
        COMPONENTS OF CommonArguments (WITH COMPONENTS
              {..., error-handling ABSENT})}
```

The components of **ReadArgument** have the following meaning:

a) Common update Arguments:
- entry gives the **DfrEntryName** of the DFR-Entry to be read (external alternative is not applicable);
- selection specifies which information from the DFR-Entry is to be read.

b) Specific read argument:
- dereferencing, when TRUE, requests the selected items to be read from the Referent instead of the DFR-Entry specified (applies only when the specified DFR-Entry is a DFR-Reference); when applied to a DFR-External-Reference deferencing implies an RDF-Transfer operation.

c: Common arguments:
- reservation, applies to the DFR-Entry specified in the request
- error-handling, always taken by default (all-or-nothing is used);
- priority, see 8.1.3.3;
- privileges, see 8.1.3.4

8.2.5.2 Read-result

Should the request succeed, the **ReadResult** will be returned.

```
ReadResult ::= CommonUpdateResult
  (WITH COMPONENTS { ... , entryInformation PRESENT } )
```

The components of the **ReadResult** have the following meaning:

- upi is the **DfrUniquePermanentIdentifier** of the DFR-Entry;

- entry-information returns all those items from the DFR-Entry (attributes and/or content) which have been requested by the selection component of the **ReadArgument**, and which are present in the DFR-Entry and accessible to the requesting DFR-User;
- warnings are never returned.

8.2.5.3 Read Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3 .

8.2.6 Modify

The **Modify** abstract operation modifies the attributes and/or content of the specified DFR-Entry. Existing attributes that are not specified in the argument are left unchanged. Attributes of the parent group are not changed. Content may be modified only for a DFR-Document. Modification of a DFR-Document-Content means the complete Content is replaced. Content can be provided explicitly or taken from a specified existing source DFR-Document.

```
Modify ::= ABSTRACT-OPERATION
  ARGUMENT ModifyArgument
  RESULT     ModifyResult
  ERRORS {
              Abandoned,
              NameError,
              UpdateError,
              AttributeError,
              VersionManagementError,
              AccessError,
              ReferentAccessError,
              InterServerAccessError,
              SecurityError,
              ServiceError}
```

8.2.6.1 Modify-argument

```
ModifyArgument ::= SEQUENCE {
  COMPONENTS OF
      CommonUpdateArguments (WITH COMPONENTS { ... ,
        entry          PRESENT,
        destination    ABSENT,
        position       ABSENT,
        modifications  PRESENT } ),
  COMPONENTS OF CommonArguments (WITH COMPONENTS
                {..., error handling ABSENT})}
```

The components of **ModifyArgument** have the following meanings:

a) Common update arguments:
- object-class optionally specifies the **DfrObjectClass** of the DFR-Entry to be modified;
- entry gives the **DfrEntryName** of the DFR-Object to be modified (external alternative is not applicable);

- modifications specifies requested modifications of the attributes and/or content of the DFR-Entry ;
- selection optionally specifies which information from the DFR-Entry modified is to be read back to the requestor (in the modify-result) after modification is done.

b) Common arguments:
- reservation, if requested, applies to the DFR-Entry being modified
- error-handling, always taken by default (all-or-nothing is used);
- priority, see 8.1.3.3;
- privileges, see 8.1.3.4

8.2.6.2 Modify-result

Should the request succeed, the **ModifyResult** will be returned.

```
ModifyResult ::= CommonUpdateResult
```

The components of the **ModifyResult** have the following meaning:

- upi is the **DfrUniquePermanentIdentifier** of the DFR-Entry ;
- entry-information returns all those items from the DFR-Entry modified (attributes and/or content) which have been requested by the selection component of the **ModifyArgument**, and which are present in the DFR-Entry;
- warnings are never returned.

8.2.6.3 Modify Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3.

8.2.7 List

The **List** abstract operation returns the attributes for the Members of a specified DFR-Group or for the elements of a specified DFR-Search-Result-List. Only the Members of the group are listed, not their Descendants. Members of the DFR-Group which are not accessible to the initiating DFR-User are not included in this list (Other Members are however returned provided that an appropriate error handling mode is specified). The number of Members (DFR-Objects) to be listed can be limited by the countLimit argument. If this limit would be exceeded the listing of Members can be continued with the next **List** abstract operation. For each Member only specified attributes are returned. The Members are listed in the specified order.

```
List ::= ABSTRACT-OPERATION
  ARGUMENT ListArgument
  RESULT     ListResult
  ERRORS {
                Abandoned,
                NameError,
                AccessError,
                AttributeError,
                SecurityError,
                ServiceError}
```

8.2.7.1 List-argument

```
ListArgument :: = SEQUENCE {
  entry    [0] DfrEntryName,
  COMPONENTS OF
      CommonListSearchArguments (WITH COMPONENTS { ... ,
        selection PRESENT } ),
  COMPONENTS OF CommonArguments }
```

The components of **ListArgument** have the following meaning:

a) Specific list argument:
- entry identifies the DFR-Group or the DFR-Search-Result-List to be listed.

b) Common list/search arguments:
- continuation optionally specifies the point where the list process has previously terminated, and from where this time it is to be continued;
- limits optionally specifies the maximum number of DFR-Entries to be returned and/or the maximum amount of time to be spent for this abstract operation;
- selection specifies which attributes of each DFR-Entry must be returned in the list-result;
- ordering optionally specifies in which order the DFR-Entries must be put in the list-result (this ordering, if specified, overrides the eventual predefined ordering for the DFR-Group or the DFR-Search-Result-List listed).

c) Common arguments:
- reservation, if requested, applies to the DFR-Entry to be listed;
- error-handling, any of four modes may be specified (see 8.1.3.2); if the until-first-warning mode is specified all the DFR-Group Members (or DFR-Entries referenced from the DFR-Search-Result-List) are listed in the specified order until an inaccessible Member is encountered;
- priority, see 8.1.3.3;
- privileges, see 8.1.3.4

8.2.7.2 List-result

Should the request succeed, the **ListResult** will be returned.

```
ListResult :: = CommonListSearchResult
```

The components of **ListResult** have the following meaning:

Common list/search result:

- number-of-entries gives the number of DFR-Entries returned in this list-result;
- continuation optionally gives the point at which the list process has terminated this time, after one of the 'limits' (specified in the **ListArgument**) has been encountered; it is present only when limit-encountered component (see hereafter) is also present;
- limit-encountered optionally indicates which of two limits (number of entries or time) has been encountered; it is absent if either limits have not been specified in the **ListArgument**, or none of them has been encountered;
- entry-list gives the list of DFR-Entries found in the DFR-Group specified (or referenced from the DFR-Search-Result-List specified), in the order specified; for each DFR-Entry listed, only those attributes are returned which have been requested by the selection component in the **ListArgument**

- warnings, if returned, identify those DFR-Entries which have not been listed.

8.2.7.3 List Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3 .

8.2.8 Search

The **Search** abstract operation searches for all objects in the specified search domain satisfying the specified search criteria. The search domain is defined by one or several (starting) DFR-Groups and/or DFR-Search-Result-Lists. Members of the search domain which are not accessible to the initiating DFR-User are not included in the search result. The search result takes the form of a **DfrEntryList** (see 8.1.6.2); it will be stored in a specified DFR-Search-Result-List and/or returned to the requestor in the result of the **Search** abstract operation. The resulting **DfrEntryList** contains the UPI and DFR-Object-Class for all DFR-Entries satisfying the **Filter** of the **Search** abstract operation. The **Filter** of the **Search** abstract operation is either provided by the user or taken from a specified DFR-Search-Result-List. The number of hits noted in the resulting **DfrEntryList** may be limited. The descent depth of the search process, that is how many levels of Descendants in the DFR-Object-Tree of each starting DFR-Group shall be examined, can be restricted. Members in the DFR-Object-Tree of each starting DFR-Group are examined in the search process to check whether they satisfy the **Filter**. Referents of DFR-References are examined on request. If a Referent is examined the depth argument is also valid for the DFR-Object-Tree of the Referent and will be related to the level on which the Referent exists in the search domain, i.e. if a Referent is a DFR-Group and the DFR-Reference pointing to this Referent exists on level two in the DFR-Object-Tree of a starting DFR-Group to be searched, the counting of the depth for the search process in the DFR-Object-Tree of the Referent starts with two. If descent depth is not specified the entire DFR-Object-Tree of each starting DFR-Group and of each Referent will be examined.

```
Search :: = ABSTRACT-OPERATION
  ARGUMENT SearchArgument
  RESULT     SearchResult
ERRORS {
                Abandoned,
                NameError,
                AccessError,
                AttributeError,
                UpdateError,
                -- concerns the searchResultList to be filled --
                SecurityError,
                ServiceError }
```

8.2.8.1 Search-argument

```
SearchArgument ::= SEQUENCE {
  search-mode     [0] CHOICE {
      continue                [0] DfrEntryName,
      -- Continue the search with all options (search domain, search criteria and --
      -- continuation context) from the search result list specified by the DfrEntryName. --
      -- The result will be added to the present content of this search result list. --
      update                  [1] DfrEntryName,
      -- The present content of the search result list is verified and eventually updated. --
      new-search-stored       [2] DfrEntryName,
      -- All options are supplied by the requestor in the subsequent parameters; they are --
      -- stored in the search result list specified, were the result is then also stored. --
      non-stored-search       [3] NULL }
      -- All options are supplied by the requestor in the subsequent parameters; they are --
      -- not stored, and the result also is not stored but only returned to the requestor. --
  COMPONENTS OF
      CommonListSearchArguments,
      search-domain  [5] SearchDomain OPTIONAL,
      search-criteria  [6] SearchCriteria OPTIONAL,
  COMPONENTS OF CommonArguments ( WITH COMPONENTS
              {..., error-handling ABSENT } ) }
```

The components of **SearchArgument** have the following meanings:

a) Specific search arguments:
- search-mode specifies the execution mode for this **Search** abstract operation;
- search-domain optionally specifies the domain of this **Search** abstract operation;
- search-criteria optionally specifies the criteria for a DFR-Entry to be put in the search-result (both the search-domain and the search-criteria arguments shall be present for the search-mode = 2 or 3, and shall be absent for search-mode = 0 or 1).

b) Common list/search arguments:
- continuation optionally specifies the point where the search process has previously terminated, and from where this time it is to be continued ;
- limits optionally specifies the maximum number of DFR-Entries to be found and/or the maximum amount of time to be spent for this abstract operation;
- selection optionally specifies which attributes of each DFR-Entry found in this **Search** operation must be returned in the entry-list parameter of the result of this operation;
- ordering optionally specifies in which order the DFR-Entries must be put in the search-result ;

c) Common arguments:
- reservation, if requested, applies to the DFR-Search-Result-List if specified in the search-mode parameter;
- error-handling, always taken by default (all-or-nothing is used);
- priority, see 8.1.3.3;
- privilege, see 8.1.3.4.

8.2.8.2 Search-result

Should the request succeed, the **SearchResult** will be returned.

```
SearchResult :: = SEQUENCE {COMPONENTS OF
                                CommonListSearchResult,
                            removed-entries [4] DfrEntryList  OPTIONAL }
```

The components of **ListResult** have the following meaning:

Common list/search result:

- number-of-entries gives the number of DFR-Entries found in this search;
- continuation optionally gives the point at which the search process has terminated this time, after one of the limits (specified in the **SearchArgument**) has been encountered; it is present only when limit-encountered component (see hereafter) is also present;
- limit-encountered optionally indicates which of two limits (number of entries or time) has been encountered; it is absent if either limits have not been specified in the search-argument, or none of them has been encountered;
- entry-list optionally gives the list of DFR-Entries found, in the order specified; for each DFR-Entry listed, only those attributes are returned which have been requested by the selection component in the **SearchArgument**.
- removed-entries gives in the case of a Search in update mode the list of DFR-Entries removed from the previously existing **DfrSearchResultList**, each element comprising only the upi, **DfrObjectClass** and possibly ordering-attribute components;
- warnings are never returned.

8.2.8.3 Search Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3 .

8.2.9 Reserve

The **Reserve** abstract operation is used to modify the reservation level of the DFR-Object specified. There are three levels of reservation defined. Level zero is "unreserved". Level one restricts write access to a DFR-Object's content and attributes (exclusive-write reservation) to the DFR-User who has reserved it. Level two in addition restricts read access to the DFR-Object's content (exclusive-access reservation), allowing access only to the DFR-User who has reserved it (this does not affect read access to the attributes of the DFR-Object, that is, other users may still read the attributes). The reservation level of a DFR-Object may only be changed by a **Reservation** abstract operation or by specifying a reservation parameter in other appropriate abstract operations. **Unbind** does not change the reservation level of a DFR-Entry. If a DFR-Reference is reserved, the Referent is not affected, that is, not reserved. The DFR-Reserved-By attribute is used to indicate the identity of the DFR-User who reserved the object.

```
Reserve :: = ABSTRACT-OPERATION
  ARGUMENT ReserveArgument
  RESULT     ReserveResult
  ERRORS {
                NameError,
                ReservationError,
                SecurityError,
                ServiceError }
```

8.2.9.1 Reserve-argument

```
ReserveArgument :: = SEQUENCE {
  entry              [0] DfrEntryName,
  COMPONENTS OF CommonArguments ( WITH COMPONENTS {
            ..., reservation PRESENT, error-handling ABSENT})}

Reservation :: = ENUMERATED {
  unreserved         (0),
  exclusive-write    (1),
  exclusive-access   (2) }
```

The components of **ReserveArgument** have the following meanings:

a) Specific reserve argument:
- entry gives the **DfrEntryName** of the DFR-Entry to be reserved;

b) Common arguments:
- reservation specifies the level of reservation;
- error-handling, always taken by deafult (all-or-nothing is used);
- priority, see 8.1.3.3;
- privilege, see 8.1.3.4.

8.2.9.2 Reserve-result

Should the request succeed, the **ReserveResult** will be returned. There are no parameters.

```
ReserveResult :: = NULL
```

8.2.9.3 Reserve Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3 .

8.2.10 Abandon

The **Abandon** abstract operation informs the DFR-Server that the user is no longer interested in the execution of a previously started outstanding abstract operation. The DFR-Server may, for example, cease processing the outstanding abstract operation, and may discard any result so far achieved.

```
Abandon :: = ABSTRACT-OPERATION
  ARGUMENT    AbandonArgument
  RESULT      AbandonResult
  ERRORS      { AbandonFailed }
```

8.2.10.1 Abandon-argument

```
AbandonArgument :: = SEQUENCE {
  object  [0] DfrEntryName  }
```

The components of **AbandonArgument** have the following meanings:

- entry specifies the **DfrEntryName** of a DFR-Entry which was specified as an argument in the abstract operation to be abandoned.

8.2.10.2 Abandon-result

Should the request succeed, no **AbandonResult** will be returned. Instead, the abstract operation abandoned reports the **Abandoned** abstract-error.

```
AbandonResult :: = NULL
```

8.2.10.3 Abandon Abstract-errors

Should the request fail, one of the listed abstract-errors will be reported. The circumstances under which the particular abstract-errors will be reported are defined in 8.3.

8.3 Abstract-Errors

The abstract errors that may be reported in response to the invocation of abstract operations at the DFR-Server port are defined and described in this clause.

Any error means that the abstract operation has changed nothing (this does not apply, however, to *warnings*, reported by the DFR-Server as part of a normal result of a DFR abstract operation; see 8.1.3.2). The entry parameter in most DFR Abstract Errors identifies the DFR-Entry to which the abstract operation was being applied when the error occurred. If this DFR-Entry is that specified explicitly in the abstract operation's argument, this parameter takes the same form (i.e. a UPI, or a **DfrPathName**, or a relative pathname) as that used in the argument. If the DFR-Entry causing the problem is not explicitly specified in the abstract operation's argument, then the entry parameter always takes the UPI form.

If several DFR-Entries were specified in the abstract operation's argument (e.g. the original entry and the destination group in a Copy abstract operation), then an abstract error may be reported having the form of a sequence, each member of which specifies a problem related to a given entry. However, only one abstract error type may be reported at once.

8.3.1 Attribute-error

An **AttributeError** reports one or several problems encountered by the DFR-Server while attempting to read or to modify attributes of a DFR-Entry.

```
AttributeError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE {
    entry       [0] DfrEntryName OPTIONAL,
    problems  [1] SEQUENCE OF
        SEQUENCE {
          problem  [0] AttributeProblem,
          type       [1] AttributeType,
          value      [2] AttributeValue OPTIONAL }}
          -- applies also to search criteria and group membership criteria --
```

```
AttributeProblem ::= ENUMERATED {
    no-such-attribute                 (1),
    invalid-attribute-syntax          (2),
    undefined-attribute-type          (3),
    inappropriate-matching            (4),
    constraint-violation              (5),
    attribute-or-value-already-exists (6),
    illegal-modification              (7),
    inconsistent-with-other-attributes (8),
    undefined-for-this-object-class   (9)
    unsupported-document-type         (10) }
```

The entry parameter is absent in an **AttributeError** in the following cases;

- when the abstract operation causing this error was a **Create** abstract operation, and an **AttributeProblem** has been encountered when processing the attributes component of the abstract operation's argument;
- when the abstract operation causing this error was a **Search** abstract operation, and an **AttributeProblem** has been encountered when processing the search-criteria component of the abstract operation's argument.

The problems parameter specifies one or several attribute problems encountered. Each problem (identified below) is accompanied by an indication of the Attribute-Type, and, if necessary to avoid ambiguity, the value, which caused the problem:

a) no-such-attribute: The named entry lacks one of the attributes specified as an argument of the abstract operation;

b) invalid-attribute-syntax: A purported attribute value, specified as an argument of the abstract operation, does not conform to the attribute syntax of the attribute type;

c) undefined-attribute-type: An undefined attribute type was provided as an argument to the abstract operation;

d) inappropriate-matching: An attempt was made, e.g. in a **Filter**, to use a matching rule not defined for the attribute type concerned;

e) constraint-violation: An attribute value supplied (or specified indirectly) in the argument of an abstract operation does not conform to the static constraints imposed by a functional standard or by the attribute definition (e.g. the value exceeds the maximum size allowed).

f) attribute-or-value-already-exists: An attempt was made to add an attribute which already existed in the entry, or a value which already existed in the attribute;

g) illegal-modification: An attempt was made to modify an attribute (i.e. to add or remove the entire attribute or some of its values), which has some specific behavior in DFR, that is, either a DFR-Server assigned attribute (e.g. UPI or Number-Of-Group-Members), or an attribute which, once assigned by the DFR-User, may not be modified in the way specified in the abstract operation (the rules are specified in 9);

h) inconsistent-with-other-attributes: An attempt was made to modify an attribute in a way inconsistent with other attributes of the same DFR-Object (e.g. if a new version of some Conceptual-Document is specified with the DFR-Version-Root attribute identifying this Conceptual-Document, and the DFR-Previous-Versions attribute

pointing to a version of some other Conceptual-Document). An inconsistency with some existent attribute is not reported if it is eliminated by further modifications specified in the same abstract operation. If two attributes enter in conflict, the DFR-Server may report an **AttributeProblem** for either of them, or for both.

i) undefined-for-this-object-class: An Attribute-Type was specified which is not defined for the DFR-Object-Class of the DFR-Entry concerned (e.g. the Number-Of-Group-Members for a DFR-Document). This does not apply to DFR-Entries examined during a **List** or a **Search** abstract operation;

j) unsupported-document-type: An abstract operation has attempted to use a DFR-Document-Type which was not among those agreed at bind time.

8.3.2 Name-error

A **NameError** reports a problem related to a name of a DFR-Entry specified in the argument of an abstract operation. The **DfrEntryName** which caused a problem is reported as it was specified, accompanied by an indication of the problem encountered.

```
NameError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE OF SEQUENCE {
    entry      [0] DfrEntryName,
    problem    [1] NameProblem }

NameProblem :: = ENUMERATED {
  invalid-upi                (1),
  invalid-path-name          (2),
  ambiguous-path-name        (3)
  inappropriate-object-class (4) }
```

A **NameProblem** may be one of the following:

a) invalid-upi: The UPI provided in the abstract operation's argument does not refer to any DFR-Object existing in the DFR-Document-Store (either this UPI has never been assigned, or the related DFR-Object has been deleted from the store);

b) invalid-path-name: The **DfrPathName** (either absolute or relative) provided in the abstract operation's argument does not correspond to any existing DFR-Entry in the store;

c) ambiguous-path-name: the **DfrPathName** (either absolute or relative) provided in the abstract operations argument is ambiguous, that is, corresponds to more than one DFR-Entry;

d) inappropriate-object-class: The **DfrEntryName** provided in the abstract operation's argument points to a DFR-Object of an inappropriate DFR-Object-Class (e.g. to a DFR-Document in a **List** abstract operation).

8.3.3 Access-error

An **AccessError** reports a problem encountered when attempting to access a DFR-Entry specified in the argument of an abstract operation.

```
AccessError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE OF SEQUENCE {
    entry      [0] DfrEntryName,
    problem    [1] AccessProblem }
```

```
AccessProblem :: = ENUMERATED {
    inappropriate-object-class (1),
    insufficient-access-rights (2),
    reserved-by-another-user (3),
    externally-located-object (4) }
```

An **AccessProblem** may be one of the following:

a) inappropriate-object-class: The **DfrEntryName** provided in the abstract operation's argument refers to a DFR-Object of an inappropriate DFR-Object-Class.

b) insufficient-access-rights: An attempt to read or list a DFR-Entry has been made by a DFR-User with insufficient access rights to this entry;

c) reserved-by-another-user: The DFR-Entry to be accessed is at present reserved by another user (with reservationLevel set to "exclusiveAccess");

d) externally-located-object: the DFR-Entry to be accessed is located in another DFR-Document-Store and it is not suitable for the abstract operation requested.

8.3.4 Update-error

An **UpdateError** reports a problem encountered when attempting to modify (update), explicitly or implicitly, an existing DFR-Entry (implicit modification may be, for example, that caused to a DFR-Group when introducing a new Member into it). Deletion of a DFR-Entry, or moving it into another DFR-Group, are also considered here as modifications.

```
UpdateError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE {
    entry      [0] DfrEntryName,
    problem    [1] UpdateProblem }

  UpdateProblem :: = ENUMERATED {
    inappropriate-object-class          (1),
    insufficient-access-rights          (2),
    reserved-by-another-user            (3),
    quality-of-service-violation        (4),
    illegal-content-modification        (5),
    group-membership-criteria-violation (6),
    reference-loop-detected             (7) }
```

An **UpdateProblem** may be one of the following:

a) inappropriate-object-class: The **DfrEntryName** provided in the abstract operation's argument refers to a DFR-Object of an inappropriate DFR-ObjectClass.

b) insufficient-access-rights: An attempt to modify a DFR-Entry has been made by a DFR-User with insufficient access rights to this entry;

c) reserved-by-another-user: The DFR-Entry to be modified is at present reserved by another DFR-User;

d) quality-of-service-violation: An attempt has been made to modify a DFR-Entry while the FidelityTime specified in its **QosLevel** has not yet elapsed;

e) illegal-content-modification: An attempt has been made to modify the content of a DFR-Entry which is not subject to user specified modifications (e.g. a DFR-Search-Result-List);

f) group-membership-criteria-violation: An attempt has been made to introduce a new member in a DFR-Group, or to modify an existing one, in a way that the member introduced or modified would violate the Group-Membership-Criteria defined for that group;

g) reference-loop-detected: An attempt has been made to create a new DFR-Internal-Reference or to modify an existing one in a way that the reference created or modified would form, together with some other reference-to-referent and group-to-member relations already existing in the DFR-Document-Store, at least one looping sequence of such relations.

NOTE 26
This standard does not prescribe that every DFR-Server detect all such reference loops at the point of their creation. If a DFR-Server does not detect all loops, its abstract operation shall not, however, be affected when such a loop is dynamically discovered when navigating through the store.

8.3.5 ReferentAccess-error

A **ReferentAccessError** reports a problem occurring when attempting to access the Referent of a DFR-Reference by specifying the **DfrEntryName** of the latter. This applies to internal as well as to external references.

```
ReferentAccessError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE {
    entry      [0] DfrEntryName,     --of a reference--
    problem    [1] ReferentAccessProblem }

      ReferentAccessProblem :: = ENUMERATED {
        inappropriate-object-class    (1),
        insufficient-access-rights    (2),
        reserved-by-another-user      (3),
        referent-no-more-exists       (4),
        referent-modified             (5),
        referenceContent-empty        (6) }
```

A **ReferentAccessProblem** may be one of the following:

a) inappropriate-object-class: The referent of the DFR-Reference specified in the abstract operation's argument is of an inappropriate DFR-Object-Class (e.g. when attempting to read the content of the Referent , and the latter is a DFR-Group).

b) insufficient-access-rights: An attempt to read the referent of a DFR-Reference has been made by a DFR-User with insufficient access rights to the Referent;

c) reserved-by-another-user: The Referent to be accessed is at present reserved by another DFR-User (with reservationLevel set to exclusiveAccess).

d) referent-no-more-exists: The Referent of the DFR-Reference specified in the abstract operation's argument has been deleted ("dangling reference").

e) referent-modified: The Referent of the DFR-Reference specified in the abstract operation's argument has been modified after the produce-time of the reference (the latter being stored in the qos-level component of the reference content).

f) reference-content-empty: The content of the DFR-Reference specified in the abstract operation's argument contains no UPI; the reference is at present only a "placeholder".

8.3.6 InterServerAccess-error

An **InterServerAccessError** reports a problem which occurs when the DFR-Server of the DFR-Document-Store containing the DFR-External-Reference specified in the abstract operation's argument (the sink server) has made an attempt to access the DFR-Document-Store containing the Referent of that external reference (the source store).

NOTE 27
This access may be initiated using the Referenced Data Transfer protocol between the two DFR-Servers; but the access problems specified hereafter have nothing specific to the RDT protocol.

```
InterServerAccessError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE {
    entry     [0] DfrEntryName, --of an external reference--
    problem   [1] InterServerAccessProblem }

    InterServerAccessProblem :: = ENUMERATED {
      referent-store-not-found          (1),  --bad store identification--
      referent-store-unreachable        (2),  --no port to reach it--
      referent-store-unavailable        (3),  --temporarily--
      referent-store-security-problem   (4) } --access.rights-
```

An **InterServerAccessProblem** may be one of the following:

a) referent-store-not-found: The source store, as it is specified in the content of the DFR-External-Reference, has not been found.

NOTE 28
This problem may be caused by bad content of the external reference, or by bad entry in the Directory describing the current location of the source store, or by any other access problem between the sink server and the related Directory server.

b) referent-store-unreachable: The DFR-Server managing the source store has no protocol in common with the sink server, through which the Referent could be accessed.

c) referent-store-unavailable: The source store is temporarily unavailable (e.g. because of some administrative abstract operations in progress).

d) referent-store-security-problem: The DFR-User requesting access to the source store has insufficient access rights for doing this, at least through the given sink server (this may also be caused by insufficient access rights of the sink server itself).

8.3.7 Reservation-error

A **ReservationError** reports a problem occurring when an attempt has been made to reserve or to unreserve some DFR-Entry.

```
ReservationError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE {
    entry      [0] DfrEntryName,
    problem    [1] ReservationProblem }

    ReservationProblem :: = ENUMERATED {
        cannot-reserve     (0),
        already-reserved   (1),
        not-yet-reserved   (2) }
```

A **ReservationProblem** may be one of the following:

a) cannot-reserve: The requestor of the **Reserve** abstract operation has insufficient access rights to the specified DFR-Entry in order to reserve it with the specified **Reservation** level.

b) already-reserved: An attempt has been made to reserve or to unreserve a DFR-Entry which is already reserved by another DFR-User.

c) not-yet-reserved: An attempt has been made to unreserve a DFR-Entry which is not reserved at present.

8.3.8 VersionManagement-error

A **VersionManagementError** reports a problem occurring when attempting to create a new version of a Conceptual-Document, or to copy some items from the (unique) previous version (implicitly specified in the modifications parameter of a **Create, Move, Copy** or **Modify** abstract operation).

```
VersionManagementError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE {
    entry      [0] DfrEntryName, -- of the entry itself or of its potential previous version --
    problem    [1] VersionManagementProblem }

VersionManagementProblem :: = ENUMERATED {
  inappropriate-object-class                (1), --not a DFR-document--
  insufficient-access-rights                (2), --when copying items from it--
  belongs-to-another-conceptual-document    (3) }
```

A **VersionManagementProblem** may be one of the following:

a) inappropriate-object-class: An attempt has been made to declare a version some DFR-Entry which is not a DFR-Document.

b) insufficient-access-rights: An attempt to copy some items from the (unique) previous version of the DFR-Entry specified in the abstract operation's argument has been made by a DFR-User with insufficient access rights to this previous entry;

c) belongs-to-another-conceptual-document: An attempt has been made to declare the DFR-Entry specified in the abstract operation's argument a new version of some Conceptual-Document, while this DFR-Entry is already a version of another Conceptual-Document; or, the list of DFR-Entries, specified as PreviousVersions for the given DFR-Entry, do not belong all to the same Conceptual-Document.

8.3.9 Security-error

A **SecurityError** reports a problem occurring when a DFR-User presents security parameters to a DFR-Server. This may occur either when binding or when executing a DFR abstract operation bearing a Privilege parameter.

```
SecurityError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE {
    problem    [0] SecurityProblem }

  SecurityProblem :: = ENUMERATED {
                    inappropriate-authentication (1),
                    invalid-creds                (2),
                    invalid-privilege            (3),
                    insufficient-access-rights   (4),
                    invalid-pac                  (5) }
```

A **SecurityProblem** may be one of the following:

a) inappropriate-authentication: The level of security associated with the requestor's credentials is inconsistent with the level of protection requested;

b) invalid-creds: the supplied simple credentials were invalid;

c) invalid-privileges: invalid privileges used in the PAC passed in the **Privileges** parameter;

d) insufficient-access-rights: The requestor does not have the right to carry out the requested abstract operation (e.g. Modify), independently of the DFR-Entry involved in the abstract operation;

e) invalid-pac: The supplied PAC is invalid

8.3.10 Service-error

A **ServiceError** reports a problem related to the provision of the service, which is not due to an incorrect abstract operation request or the requestor's access rights.

```
ServiceError :: = ABSTRACT-ERROR
  PARAMETER SEQUENCE {
    problem    [0] ServiceProblem }

  ServiceProblem :: = ENUMERATED {
    server-busy                (1),  --please wait and repeat--
    server-unavailable         (2),  --please unbind--
    operation-too-complex      (3),  --e.g. searchCriteria--
    resource-limit-exceeded    (4),  --e.g. when creating a bulky object--
    unclassified-server-error  (5) } --e.g. for any bug in the server--
```

A **ServiceProblem** reported may be one of the following:

a) server-busy: The DFR-Server is presently too busy to perform the requested abstract operation, but may be able to do so after a short while;

b) server-unavailable: The DFR-Server is currently unavailable;

c) operation-too-complex: The requested abstract operation is too complex syntactically or semantically (e.g. the SearchCriteria in a Search abstract operation have too many nesting levels to be correctly understood by the given DFR-Server);

d) resource-limit-exceeded: This may happen for example when a very large object is to be created or copied in the DFR-Document-Store or large List or Search operations are requested. The resource limit may be that for the overall document store, or for the requestor of the abstract operation.

e) unclassified-server-error: This is a place-holder for "any other" errors, primarily to those due to software bugs in a yet unstable DFR-Server implementation, to make it nevertheless possible to access it remotely before it becomes sufficiently robust.

8.3.11 Abandon-error

An **AbandonError** reports a problem occurring when attempting to abandon some previously requested DFR abstract operation.

```
AbandonFailed ::= ABSTRACT-ERROR
  PARAMETER SET {
    problem    [0] AbandonProblem,
    operation  [1] InvokeId }

  InwokeId ::= INTEGER

  AbandonProblem ::= ENUMERATED {
    no-such-operation   (1),
    too-late            (2),
    cannot-abandon      (3) }
```

An **AbandonProblem** may be one of the following:

a) no-such-operation: The DFR abstract operation specified is not currently known to the DFR-Server;

b) too-late: The execution of the abstract operation specified has entered the phase in which it already cannot be abandoned, or such an abandon would make no sense;

c) cannot-abandon: The abstract operation specified is of the kind for which an abandon is not permitted (e.g. **Modify** or **Delete**).

8.3.12 Abandoned

This is a report of a DFR abstract operation after it has been abandoned by an **Abandon** abstract operation. It is not literally an error, but may instead be considered as a success report of the related Abandon abstract operation. It may not occur if the related Abandon abstract operation has reported an **AbandonError**.

```
Abandoned ::= ABSTRACT-ERROR
```

8.3.13 Error Precedence

Should several error conditions occur simultaneously for the same DFR abstract operation only one of them is reported to the requestor. The precedence of these error conditions is as follows beginning with the most important:

SecurityError
ServiceError
NameError
AccessError
InterServerAccessError
ReferentAccessError
ReservationError
Abandoned
AttributeError
UpdateError
VersionManagementError
AbandonFailed

8.4 Function Sets

DFR-Document-Stores have flexible structures as indicated in Figure 2 (see 6.3). But, from user's viewpoint, it is necessary to define functional sets of this structure. Thus, the following usage types and corresponding Function Sets are defined:

Usage Type 1: Flat Store Set

In this Usage Type, the structure of a DFR-Document Store is not important. All DFR-Documents, DFR-References or DFR-Search-Result-Lists in this type of DFR-Document Store are directly included in the DFR-Root-Group.

In this set, no DFR-Groups (except the DFR-Root-Group) are allowed in a DS.

Usage Type 2: Pre-defined Store Structure Set

In this Usage Type, DFR-Objects are stored according to a pre-defined structure of DFR-Groups. A DFR-User can create, modify or delete DFR-Documents, DFR-References and DFR-Search-Result-Lists in each DFR-Group. But, a DFR-User can not create , modify or delete DFR-Groups.

In this set, no DFR-User can create or delete DFR-Groups. DFR-Groups are predefined.

Usage Type 3: Full Set

In this Usage Type, users are free to create, modify or delete any DFR-Documents, DFR-References, DFR-Groups or DFR-Search-Result-Lists in a DFR-Document-Store.

In this set, all DFR-Users can create, modify or delete DFR-Groups.

All DFR abstract operations applied to DFR-Documents, DFR-References and DFR-Search-Result-Lists and the abstract operations **Reserve** and **Abandon** are available in Function Sets 1 and 2 and 3.

For DFR abstract operations applied to DFR-Groups :

Read, List and **Search** are only available in Function Sets 2 and 3;

Create, Delete, Copy, Move and **Modify** are only available in Function Set 3.

SECTION THREE - DFR ATTRIBUTES

9. ATTRIBUTE DEFINITIONS

9.1 Overview of Attributes

The DFR information-model and the attributes were introduced in chapter 6.

For the **DfrEntryAttributes** the following subclauses contain a short description of each Attribute-Type together with its abstract-syntax using the ATTRIBUTE macro defined in 6.3.7.4.

It should be noted that some attributes are used primarily for filtering and listing purposes while others are used for system purposes only.

The concept of how attributes support Security in DFR is described in 6.3.8.

The attributes defined in this Standard are independent of the nature of the stored information, that is no specific attributes for ODA, SGML or other document content standards are provided. However, a subset of attributes from the ODA-Document-Profile can be mapped to DFR-Attributes (see Appendix A). DFR gives support to a subset of ODA Attributes from the ODA Document Profile (see Standard ECMA-101 Part 4). These attributes are considered not to be specific to DFR and therefore their names appear without "DFR" prefix. The DFR attribute mechanism is intended to accommodate different attribute sets defined in International Standards other than ODA also, for example SGML, IPM, and EDI.

NOTE 29
The consistency between the ODA-Document-Profile attributes and their corresponding DFR-Attributes is beyond the scope of this standard.

The attribute types defined in this standard are grouped in two subsets, the DFR-Basic-Attribute-Set and the DFR-Extension-Attribute-Set. DFR supports mandatorily the DFR-Basic-Attribute-Set and optionally the DFR-Extension-Attribute-Set.

NOTE 30
There will eventually be different extension attribute sets, e.g. each having its own abstract syntax. Extension-Attribute-Sets to be used in an application association are negotiable at association establishment time. The DFR-Extension-Attribute-Set (defined in this Standard) is included in the definition of DFR application Contexts (see Appendix of Part 2 of this Standard); its presence however is also negotiable.

The attributes defined in this Standard are listed in the tables in the following pages. The details are given in the attribute-type descriptions. The tables which are provided for the DFR-Basic-Attribute-Set and the DFR-Extension-Attribute-Set each consists of two parts. Tables 3 and 4 show for the various attributes whether the attribute-type is single-valued or multi-valued, by whom the attribute-type is managed in different situations and whether an attribute is automatically copied in a **Copy** abstract operation. Tables 5 and 6 show for the various attributes, to which DFR-Objects these attributes are assigned to, their occurrences, and their matching behavior in **Search** abstract operations including their availability for ordering.

Attribute-type name	Single/ Multi Valued	Assigned by	Modified by	Deleted by	Autom. Copied by Copy
DFR-UPI	S	D			N
DFR-Object-Class	S	O			Y
DFR-Document-Type	S	U	U		Y
Document-Architecture-Class	S	U	U	U	Y
DFR-Title	S	U	U		Y
DFR-Pathname	S	D	D		N
DFR-Parent-Identification	S	D	D		N
DFR-Guarantee-QoS	S	O	O		N
DFR-Referent-Deleted	S	D		D	Y
DFR-Membership-Criteria	S	U	U	U	Y
DFR-Ordering	S	U	U	U	Y
DFR-Resource-Limit	S	O	O	O	Y
DFR-Resource-Used	S	D	D		Y
DFR-Number-Of-Group-Members	S	D	D		Y
Version-Name	S	U		U	N
DFR-Previous-Version	M	U	D	U	N
DFR-Next-Version	M	D	D	U	N
DFR-Version-Root	S	D		U	N
DFR-External-Location	S	U	U	U	Y
User-Reference	S	U	U	U	Y
User-Reference-To-Other-Objects	M	U	U	U	Y
DFR-Attributes-Create-Date-And-Time	S	D			N
DFR-Content-Create-Date-And-Time	S	D			N
DFR-Created-By	S	D			N
DFR-Attributes-Modify-Date-And-Time	S	D	D		N
DFR-Content-Modify-Date-And-Time	S	D	D		N
DFR-Attributes-Modifyed-By	S	D	D		N
DFR-Content-Modifyed-By	S	D	D		N
Document-Date-And-Time	S	U	U	U	Y
DFR-Reservation	S	D	D	D	N
DFR-Reserved-By	S	D	D	D	N
DFR-Access-List	M	O	O		N

Table 3: DFR-Basic-Attribute-Set

S = SINGLE VALUE, M = MULTI VALUE, D = DFR-Server, U = DFR-User, O = OWNERS, Y = YES, N = NO

Attribute types	Used for				Matches in filter for		
	DO	GR	RE	SR	EQ	OR	SU
DFR-UPI	M	M	M	M	M		
DFR-Object-Class	M	M	M	M	M		
DFR-Document-Type	M		R		M		
Document-Architecture-Class	O		R		M	M	
DFR-Title	M	M	M	M	M		M
DFR-Pathname	C	C	C	C	M		M
DFR-Parent-Identification	M	M	M	M			
DFR-Guarantee-QoS	M	M					
DFR-Referent-Deleted			C		M		
DFR-Membership-Criteria		O					
DFR-Ordering		O					
DFR-Resource-Limit	O	O	O	O	O	O	
DFR-Resource-Used	M	M	M	M	O	O	
DFR-Number-Of-Group-Members		M			O	O	
Version-Name	C		R		O		O
DFR-Previous-Version	C				O		
DFR-Next-Version	C				O		
DFR-Version-Root	C				O		
DFR-External-Location	O	O	R		M		M
User-Reference	O	O	R	O	O		O
User-Reference-To-Other-Objects	O	O	R	O	O		O
DFR-Attributes-Create-Date-And-Time	M	M	M	M	M	M	
DFR-Content-Create-Date-And-Time	C	C	C	C	M	M	
DFR-Created-By	M	M	M	M	M		
DFR-Attributes-Modify-Date-And-Time	C	C	C	C	M	M	
DFR-Content-Modify-Date-And-Time	C	C	C	C	M	M	
DFR-Attributes-Modifyed-By	C	C	C	C	M		
DFR-Content-Modifyed-By	C	C	C	C	M	M	
Document-Date-And-Time	O		R		M	M	
DFR-Reservation	C	C	C	C	O	O	
DFR-Reserved-By	C	C	C	C	M		
DFR-Access-List	C	C	C	C	O		

Table 4: DFR-Basic-Attribute-Set

DO = DFR-Document, GR = DFR-Group, RE = DFR-Reference, SR = DFR-Search-Result-List
EQ = Equality, OR = Ordering, SU = Substrings, C = Conditional, M = Mandatory, O, R = Optional

NOTE 31

For the DFR-Reference (RE) the letter R is used to indicate that the values of the marked optional attributes are inherited from the Referent at the DFR-Reference creation time. If the values of these attributes of the Referent are changed later on, the values of the DFR-Reference are not changed.

Attribute-type name	Single/ Multi Valued	Assigned by	Modified by	Deleted by	Autom. Copied by Copy
Other-Titles	M	U	U	U	Y
Subject	S	U	U	U	Y
Document-Type	S	U	U	U	Y
Keywords	M	U	U	U	Y
Creation-Date-And-Time	S	U	U	U	Y
Purge-Date-And-Time	S	U	U	U	Y
Revision-Date-And-Time	S	U	U	U	Y
Organizations	M	U	U	U	Y
Prepares	M	U	U	U	Y
Owners	M	U	U	U	Y
Authors	M	U	U	U	Y
Status	S	U	U	U	Y
User-Specifc-Codes	M	U	U	U	Y
Superseded-Document	M	U	U	U	Y
Number-Of-Pages	S	U	U	U	Y
Languages	M	U	U	U	Y

Table 5: DFR-Extension-Attribute-Set

S = SINGLE VALUE, M = MULTI VALUE, D = DFR-Server, U = DFR-User, Y = YES, N = NO

Attribute types	Used for				Matches in filter for		
	DO	GR	RE	SR	EQ	OR	SU
Other-Titles	O	O	R	O	M		M
Subject	O	O	R	O	M		M
Document-Type	O				M		M
Keywords	O	O	R	O	M		M
Creation-Date-And-Time	O	O	R	O	M	M	
Purge-Date-And-Time	O	O	R	O	M	M	
Revision-Date-And-Time	O	O	R	O	M	M	
Organizations	O		R		M		M
Prepares	O		R				
Owners	O		R				
Authors	O		R				
Status	O		R		M		M
User-Specifc-Codes	O	O	R	O	M		M
Superseded-Document	O		R		M		M
Number-Of-Pages	O		R		M	M	
Languages	O		R		M		M

Table 6: DFR-Extension-Attribute-Set

DO = DFR-Document, GR = DFR-Group, RE = DFR-Reference, SR = DFR-Search-Result-List
EQ = Equality, OR = Ordering, SU = Substrings, C = Conditional, M = Mandatory,
O = Optional, R = Optional

NOTE 32
For the DFR-Reference (RE) the letter R is used to indicate that the values of the marked optional attributes are inherited from the Referent at the DFR-Reference creation time. If the values of these attributes of the Referent are changed later on, the values of the DFR-Reference are not changed.

9.2 DFR-Basic-Attribute-Set

9.2.1 DFR-UPI (DFR-Unique-Permanent-Identifier)

This attribute is used by a DFR-Server to uniquely identify a given DFR-Document, DFR-Group or DFR-Reference within the DFR-Document-Store. Its value cannot be interpreted by the DFR-User. This attribute is associated with each DFR-Object. Once assigned by a DFR-Server, the value of each UPI will not be changed within the life of a DFR-Object. In addition, this UPI value will differ from that of all DFR-Objects which once existed in the same DFR-Server (including all existing DFR-Objects and all deleted DFR-Objects).

```
dfr-upi ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier
    MATCHES FOR EQUALITY
    SINGLE VALUE
    :: = id-att-dfr-upi
```

9.2.2 DFR-Object-Class

This attribute indicates the class of a DFR-Object (DFR-Document, DFR-Group, DFR-Reference or DFR-Search-Result-List). This attribute is associated with each DFR-Object.

```
dfr-object-class ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX DfrObjectClass
    MATCHES FOR EQUALITY
    SINGLE VALUE
    :: = id-att-dfr-object-class
```

9.2.3 DFR-Document-Type

This attribute contains an object identifier whose value defines the representation for the document content, for example ODA or SGML in the DFR access protocol. For a DFR-Reference this attribute will only exist if the Referent is a DFR-Document and in this case the attribute is mandatory.

```
dfr-document-type ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX OBJECT IDENTIFIER
    MATCHES FOR EQUALITY
    SINGLE VALUE
    :: = id-att-dfr-document-type
```

9.2.4 Document-Architecture-Class

This attribute specifies the document architecture class used in the document. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the ODA Document Profile.

document-architecture-class ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DocumentArchitectureClass
MATCHES FOR EQUALITY ORDERING
SINGLE VALUE
:: = id-att-document-architecture-class

9.2.5 DFR-Title

This attribute gives the name of the DFR-Object as specified by the DFR-User.

dfr-title ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
SINGLE VALUE
:: = id-att-dfr-title

9.2.6 DFR-Pathname

This attribute is a sequence of DFR-Title attribute values of the DFR-Object's Parents in descending order beginning from the DFR-Root-Group. It is only defined in the context of a DS for which global or local DFR-Title attribute uniqueness is enforced by the DFR-Server (see 7.1.2). For a DFR-Root-Group this attribute by convention is an empty sequence of DFR-Titles.

dfr-pathname ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreListSyntax
SINGLE VALUE
:: = id-att-dfr-pathname

9.2.7 DFR-Parent-Identification

This attribute identifies the DFR-Group of which this DFR-Object is a member. Its value is equal to the UPI of this DFR-Object's Parent group. This attribute is associated with each DFR-Object. For a DFR-Root-Group it is by convention an octet string of 0 (zero) length.

dfr-parent-identification ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier
SINGLE VALUE
:: = id-att-dfr-parent-identification

9.2.8 DFR-Guarantee-QoS

This attribute determines which QoS is obtainable for DFR-References to this DFR-Document or DFR-Group (see 6.3). No level 3 QoS References will be guaranteed later than the value (date) of this attribute. This attribute is only associated with DFR-Documents or DFR-Groups. Once a guarantee is assigned to a DFR-Object, it can only be set to unguaranteed after the expiration of the guaranteed time. Before the guarantee expires, it may be set to a later date.

dfr-guarantee-qos ATTRIBUTE
WITH ATTRIBUTE-SYNTAX QoS-Level
SINGLE VALUE
:: = id-att-dfr-guarantee-qos

9.2.9 DFR-Referent-Deleted

This attribute is assigned to a DFR-Reference and set to true once the DFR-Server detects that the Referent is deleted. This attribute will be deleted by the DFR-Server once the DFR-User makes this DFR-Reference point to another Referent.

dfr-referent-deleted ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX booleanSyntax
    SINGLE VALUE
    :: = id-att-dfr-referent-deleted

9.2.10 DFR-Membership-Criteria

This attribute is only associated with DFR-Groups. This attribute establishes constraints on group membership based on attribute values. The value of this attribute is a **Filter**.

dfr-membership-criteria ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX Filter
    SINGLE VALUE
    :: = id-att-dfr-membership-criteria

9.2.11 DFR-Ordering

This attribute defines a default ordering of a DFR-Group's members in a **List** abstract operation. The ordering rule is one specific attribute named by the DFR-User and noted in the DFR-Ordering attribute. Attributes available for ordering are those which have a MATCHES FOR ORDERING clause in their definition. This attribute is associated only with DFR-Group objects.

dfr-ordering ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX OrderingRule
    SINGLE VALUE
    :: = id-att-dfr-ordering

9.2.12 DFR-Resource-Limit

This attribute specifies the maximum resource to be used for a DFR-Object based upon accounting information, for example the actual amount of storage in the Document store used. The size includes the space required to store content (if a Document), the set of members (if a DFR-Group) and any associated attributes.

dfr-resource-limit ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX integerSyntax
    SINGLE VALUE
    :: = id-att-dfr-resource-limit

9.2.13 DFR-Resource-Used

This attribute contains information for accounting purposes based on resources used during some period of time, for example the actual amount of storage in the Document Store used. The resource used includes the space required to store content (if a DFR-Document), the set of members (if a DFR-Group) and any associated attributes.

dfr-resource-used ATTRIBUTE
WITH ATTRIBUTE-SYNTAX integerSyntax
SINGLE VALUE
:: = id-att-dfr-resource-used

9.2.14 DFR-Number-Of-Group-Members

This attribute specifies the number of immediate Descendants in a DFR-Group.

dfr-number-of-group-members ATTRIBUTE
WITH ATTRIBUTE-SYNTAX integerSyntax
SINGLE VALUE
:: = id-att-dfr-number-of-group-members

9.2.15 Version Name

This is a free-form attribute intended for the DFR-User's perception and managed by the DFR-User. It is defined primarily for DFR-Documents which are declared to be versions (in the sense of DFR version management as described in 6); but it can also be used for any other DFR-Document. It may also appear in a DFR-Reference to a DFR-Document, normally as a copy of the corresponding attribute of the Referent. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the ODA Document Profile.

version-name ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
SINGLE VALUE
:: = id-att-version-name

9.2.16 DFR-Previous-Versions

This is a multi-valued attribute. It is defined only for DFR-Documents. It is assigned by the DFR-User when the document is declared a new version (in a **Create** or **Modify** abstract operation). It may then be modified provided that the DFR-Document has not yet become a previous version for some other new version(s); after that it cannot be modified. It is updated automatically if any specified previous version disappears (by means of a **Delete** or **Modify** abstract operation).

dfr- previous-versions ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier
MATCHES FOR EQUALITY
MULTI VALUE
:: = id-att-dfr-previous-versions

9.2.17 DFR-Next-Versions

This is a multi-valued attribute. It is defined only for DFR-Documents. It is updated by the DFR-Server each time a new version is declared having this DFR-Document as its previous version (in a **Create** or **Modify** abstract operation), or when such an existing version is discarded (by a **Delete** or **Modify** abstract operation). The DFR-User is prohibited from modifying this attribute explicitly.

dfr-next-versions ATTRIBUTE

WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier

MATCHES FOR EQUALITY

MULTI VALUE

:: = id-att-dfr-next-versions

9.2.18 DFR-Version-Root

This attribute is defined and has the same value for all DFR-Documents which are declared versions of the same Conceptual-Document (see section 6), as well as, optionally, for DFR-References to these documents. It is assigned for the first time by the DFR-Server when a DFR-Document is declared to be a next version of some other DFR-Document, which has not been previously declared to be a version. The UPI attribute of the latter becomes the value of the DFR-Version-Root attribute for both the old and the new version. The value of the DFR-Version-Root attribute is then systematically copied by the DFR-Server into the DFR-Version-Root attribute of each new version of the same Conceptual-Document. The DFR-Version-Root attribute remains valid even when the "original version", bearing the UPI which is the value, has been deleted. When creating a DFR-Reference to a document, it is normally copied by the DFR-Server from the Referent into the corresponding attribute of the DFR-Reference, thus permitting a DFR-User to find some existing version of the same Conceptual-Document even after the version which was the Referent has been deleted.

dfr-version-root ATTRIBUTE

WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier

MATCHES FOR EQUALITY

SINGLE VALUE

:: = id-att-dfr-version-root

9.2.19 DFR-External Location

This attribute contains a user specified description of the location of an object stored outside any DFR-Document-Store.

dfr-external-location ATTRIBUTE

WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax

SINGLE VALUE

:: = id-att-dfr-external-location

9.2.20 User-Reference

This attribute contains an identifier for this DFR object. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the ODA Document Profile. The attributes User-Reference and User-Reference-To-Other-Objects may be used to establish user references between DFR-Objects stored in a DFR-Document-Store. That is, the value of the attribute User-Reference is a DFR-User specific identifier for an object; this identifier can be stored in the attribute User-References-To-Other-Objects. If later a value of the attribute User-References-To-Other-Objects is used for example in a **Search** abstract operation, the Referent will be identified.

user-reference ATTRIBUTE

WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax

SINGLE VALUE

:: = id-att-user-reference

9.2.21 User-References-To-Other-Objects

This attribute contains references to other DFR-Objects. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the ODA Document Profile. The attributes User-Reference and User-Reference-To-Other-Objects may be used to establish user references between DFR-Objects stored in a DFR-Document-Store. That is, the value of the attribute User-Reference is a DFR-User specific identifier for an object; this identifier can be stored in the attribute User-References-To-Other-Objects. If later a value of the attribute User-References-To-Other-Objects is used for example in a **Search** abstract operation, the Referent will be identified. The attribute User-References-To-Other-Objects may contain one or many references to other objects.

user-reference-to-other-objects ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
MULTI VALUE
:: = id-att-user-reference-to-other-objects

9.2.22 DFR-Attributes-Create-Date-And-Time

This attribute contains the date and time when the mandatory attributes of this DFR-Object were stored in a DFR-Document-Store. A DFR-Server sets it to the current date and time during the **Create** abstract operation.

dfr-attributes-create-date-and-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
SINGLE VALUE
:: = id-att-dfr-attributes-create-date-and-time

9.2.23 DFR-Content-Create-Date-And-Time

This attribute contains the date and time the DFR-Content of this DFR-Object was stored in a DFR-Document-Store. DFR-Server will set it to current date and time during the **Create** abstract operation.

dfr-content-create-date-and-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
SINGLE VALUE
:: = id-att-dfr-content-create-date-and-time

9.2.24 DFR-Created-By

This attribute identifies the DFR-User who created this DFR-Object. It is not modified when the DFR-Object is moved. It may only be read by a DFR-User having at least extended-read access right.

dfr-created-by ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DistinguishedName
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-dfr-created-by

9.2.25 DFR-Attributes-Modify-Date-And-Time

This attribute contains the date and time the attributes of this DFR-Object were last modified in a DFR-Document-Store. When a DFR-Object is created, this attribute is set to the current time. Subsequently, the DFR-Server maintains the attribute. This attribute is not updated

when those attributes described in Table 3 as being modified or deleted by the DFR-Server are modified or deleted.

dfr-attributes-modify-date-and-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
SINGLE VALUE
:: = id-att-dfr-attributes-modify-date-and-time

9.2.26 DFR-Content-Modify-Date-And-Time

This attribute contains the date and time the DFR-Content of this DFR-Object was last modified in a DFR-Document-Store. When a DFR-Object is created, this attribute is set to the current time. Subsequently, the DFR-Server maintains the attribute.

dfr-content-modify-date-and-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
SINGLE VALUE
:: = id-att-dfr-content-modify-date-and-time

9.2.27 DFR-Attributes-Modified-By

This attribute identifies the DFR-User who has most recently modified DFR-Attributest of that DFR-Object. It may only be read by a DFR-User having at least extended-read access right.

dfr-attributes-modified-by ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DistinguishedName
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-dfr-attribute-modified-by

9.2.28 DFR-Content-Modified-By

This attribute identifies the DFR-User who has most recently modified the DFR-Content of that DFR-Object. It may only be read by a DFR-User having at least extended-read access right.

dfr-content-modified-by ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DistinguishedName
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-dfr-content-modified-by

9.2.29 Document-Date-And-Time

This attribute specifies the date and time that the DFR-User associates with the DFR-Document or with a DFR-Reference. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the ODA Document Profile.

document-date-and-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
SINGLE VALUE
:: = id-att-document-date-and-time

9.2.30 DFR-Reservation

This attribute indicates whether this DFR-Object is reserved or not. This attribute is associated with each DFR-Object.

dfr-reservation ATTRIBUTE
WITH ATTRIBUTE-SYNTAX Reservation
MATCHES FOR EQUALITY ORDERING
SINGLE VALUE
:: = id-att-dfr-reservation

9.2.31 DFR-Reserved-By

This attribute identifies the security subject on whose behalf the DFR-User has reserved this DFR-Object. It is absent when the DFR-Object is not reserved. It may only be read by a DFR-User having at least extended-read access right.

dfr-reserved-by ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DistinguishedName
MATCHES FOR EQUALITY
SINGLE VALUE
:: = id-att-dfr-reserved-by

9.2.32 DFR-Access-List

This attribute identifies the security subjects allowed to access this DFR-Object specifying for each of them their respective access rights. The complete values of the DFR-Access-List attribute are visible to DFR-Users having at least extended-read access right to this DFR-Object; it can be modified only by DFR-Users having the owner access right for this DFR-Object. A DFR-User having only read access right to the DFR-Object may only read his own access right from this attribute. This attribute is part of the DFR-Security mechanism that is described in 6.3.8.

dfr-access-list ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DfrAccessListElement
MATCHES FOR EQUALITY
MULTI VALUE
:: = id-att-dfr-access-list

9.3 DFR-Extension-Attribute-Set

9.3.1 Other-Titles

This attribute contains alternative titles for this DFR-Object. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

other-titles ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
MULTI VALUE
:: = id-att-other-titles

9.3.2 Subject

This attribute contains information to indicate the subject of a DFR-Object. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

subject ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
SINGLE VALUE
:: = id-att-subject

9.3.3 Document-Type

This attribute specifies the type of DFR-Document, e.g. memorandum, letter, report, resource. This attribute specifies only an informal name; it does not specify a relation to a particular document class description. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

document-type ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
SINGLE VALUE
:: = id-att-document-type

9.3.4 Keywords

This attribute specifies one or more character strings that permit logical associations to be made about the DFR-Content of an DFR-Object. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

keywords ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
MULTI VALUE
:: = id-att-keywords

9.3.5 Creation-Date-And-Time

This attribute specifies the date and, optionally, the time of day when the DFR-Document was created. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

creation-date-and-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
SINGLE VALUE
:: = id-att-creation-date-and-time

9.3.6 Purge-Date-And-Time

This attribute specifies the date and, optionally, the time of day after which the DFR-Document can be purged from the DFR-Document-Store. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

purge-date-and-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX uTCTimeSynatx
SINGLE VALUE
:: = id-att-purge-date-and-time

9.3.7 Version-Date-And-Time

This attribute specifies the date and, optionally, the time of day on which a revision of this DFR-Object occurred. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

version-date-and-time ATTRIBUTE
WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
SINGLE VALUE
:: = id-att-revision-date-and-time

9.3.8 Organizations

This attribute identifies the originating organization(s) associated with the DFR-Document. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

organizations ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
MULTI VALUE
:: = id-att-organizations

9.3.9 Preparers

This attribute identifies the name(s) of the person(s) and/or organization(s) responsible for the physical preparation of the document. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

preparers ATTRIBUTE
WITH ATTRIBUTE-SYNTAX Person
MULTI VALUE
:: = id-att-preparers

9.3.10 Owners

This attribute identifies the name(s) of the person(s) and/or organization(s) responsible for the content of the document In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

owners ATTRIBUTE
WITH ATTRIBUTE-SYNTAX Person
MULTI VALUE
:: = id-att-owners

9.3.11 Authors

This attribute identifies the name(s) of the person(s) and/or organization(s) responsible for the preparation of the intellectual content of the document. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

authors ATTRIBUTE
WITH ATTRIBUTE-SYNTAX Person
MULTI VALUE
:: = id-att-authors

9.3.12 Status

This attribute specifies the DFR-Document status, e.g. working paper, draft proposal. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

status ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
SINGLE VALUE
:: = id-att-status

9.3.13 User-Specific-Codes

This attribute specifies additional user-specific code(s) for a DFR-Object, e.g. contract number, project number, budget. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

user-specific-codes ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
MULTI VALUE
:: = id-att-user-specific-codes

9.3.14 Superseded-Documents

This attribute specifies reference(s) to document(s) superseded by the current DFR-Document. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

superseded-documents ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
MULTI VALUE
:: = id-att-superseded-documents

9.3.15 Number-Of-Pages

This attribute specifies the number of pages in the specific layout structure (if any) of the document. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

number-of-pages ATTRIBUTE
WITH ATTRIBUTE-SYNTAX integerSyntax
SINGLE VALUE
:: = id-att-number-of-pages

9.3.16 Languages

This attribute specifies the primary language(s) in which the content of the document is written. In the case of an ODA-Document the value of this attribute can be taken by the DFR-User from the "ODA Document Profile".

languages ATTRIBUTE
WITH ATTRIBUTE-SYNTAX characterDataSyntax
MULTI VALUE
:: = id-att-languages

9.4 DFR Attribute Syntaxes

The following attribute syntaxes are used in the preceeding attribute definitions.

9.4.1 String Attribute Syntaxes

In the syntaxes specified in 9.4.1.1 to 9.4.1.3, the following spaces are regarded as not significant:

- leading spaces (i.e., those preceding the first printing character);
- trailing spaces (i.e., those following the last printing character);
- multiple consecutive internal spaces (these are taken as equivalent to a single space character).

Attributes conforming to these syntaxes shall be stored and matched in a form which omits those spaces which are not significant according to these rules.

9.4.1.1 Case Ignore String

The case Ignore String attribute syntax is intended for attributes whose values are strings (either T.61 Strings or Printable Strings), but where the case (upper or lower) is not significant for comparison purposes (e.g., "Dundee" and "DUNDEE" match).

```
caseIgnoreStringSyntax ATTRIBUTE-SYNTAX
        CharacterData -- definition see 8.1.4 --
        MATCHES FOR EQUALITY SUBSTRINGS
        ::= {id-dfr-att-syn 4}
```

For two strings having this syntax to match for equality, the strings must be the same length and corresponding characters must be the same, except for case. Two strings of the same or of different types can be compared. For example, a Printable String can be compared with a T.61 String: where the corresponding characters are both in the Printable String character set and in the T.61 String, then comparison proceeds as normal. However if the character in T.61 String is not in the Printable String characters set then matching fails. Similar rules apply when comparing a Graphic String with a Printable String or a T.61 String.

9.4.1.2 Case Ignore List

The Case Ignore List attribute syntax is intended for attributes whose values are sequences of strings (either T.61 Strings or Printable Strings), but where the case (upper or lower) is not significant for comparison purposes.

```
caseIgnoreListSyntax  ATTRIBUTE-SYNTAX
        SEQUENCE OF
                CharacterData -- definition see 8.1.4 --
        MATCHES FOR EQUALITY SUBSTRINGS
        ::= {id-dfr-att-syn 5}
```

Two Case Ignore Lists match for equality if and only if the number of strings in each is the same, and corresponding strings match. The latter matching is as for Case Ignore String attribute syntax.

9.4.2 Miscellaneous Attribute Syntaxes

9.4.2.1 Boolean

The Boolean attribute syntax is intended for attributes whose values are Boolean (i.e., represent true or false).

booleanSyntax ATTRIBUTE-SYNTAX

BOOLEAN

MATCHES FOR EQUALITY

::= {id-dfr-att-syn 2}

Two attribute values of this syntax match for equality if they are both true or both false.

9.4.2.2 Integer

The Integer attribute syntax is intended for attributes whose values are integers.

integerSyntax ATTRIBUTE-SYNTAX

INTEGER

MATCHES FOR EQUALITY ORDERING

::= {id-dfr-att-syn 1}

Two attribute values of this syntax match for equality if the integers are the same. The ordering rules for integers apply.

9.4.2.3 UTC Time

The UTC Time attribute syntax is intended for attributes whose values represent absolute time.

uTCTimeSyntax ATTRIBUTE-SYNTAX

UTCTime

MATCHES FOR EQUALITY ORDERING

::= {id-dfr-att-syn 3}

Two attribute values of this syntax match for equality if they represent the same time. An earlier time is considered "less" than a later time.

SECTION FOUR - DFR REALIZATION

10. SUPPLY OF THE DFR ABSTRACT SERVICE

This clause specifies how a DFR-Server supplies the DFR abstract service. It covers the supply of the **Create, Delete, Copy, Move, Read, Modify, List, Search, Reserve,** and **Abandon** abstract operations. The actual description of the operation is in 8.

The supply of the DFR Port abstract-services assumes that an abstract-association exists between the DFR Port supplier (the DFR-Server) and the DFR Port consumer (the DFR-User).

The DFR-User may have more than one abstract operation outstanding, that is parallel performance of the abstract operations may take place.

Not all error cases are described.

10.1 Performance of the Create abstract operation

When the DFR-Server receives a **Create** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) creates a new DFR-Entry in the DFR-Document-Store placing the new DFR-Object in the appointed DFR-Group;

3) establishes the DFR-Attribute values for the new DFR-Object as appropriate;

4) establishes the value of the DFR-Content of the new DFR-Object as appropriate;

5) returns the **Create** result to the DFR-User

10.2 Performance of the Delete abstract operation

When the DFR-Server receives a **Delete** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) deletes the DFR-Entry removing the DFR-Object from its parent DFR-Group;

3) returns the **Delete** result to the DFR-User.

10.3 Performance of the Copy abstract operation

When the DFR-Server receives a **Copy** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) creates a new DFR-Entry in the destination DFR-Group;

3) copies the values of the DFR-Attributes and the DFR-Content of the appointed DFR-Object(s) as appropriate;

4) updates the DFR-Attributes of the destination DFR-Object as appropriate;

5) returns the **Copy** result to the DFR-User.

10.4 Performance of the Move abstract operation

When the DFR-Server receives a **Move** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) creates a new DFR-Entry in the destination DFR-Group;

3) changes the membership of the appointed DFR-Object from the source DFR-Group to the destination DFR-Group;

4) updates the DFR-Attributes of both the source and destination as appropriate;

5) returns the **Move** result to the DFR-User.

10.5 Performance of the Read abstract operation

When the DFR-Server receives a **Read** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) returns the values of the DFR-Attribute(s) and/or the DFR-Content of the appointed DFR-Object as appropriate to the DFR-User.

10.6 Performance of the Modify abstract operation

When the DFR-Server receives a **Modify** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) modifies the DFR-Attribute(s) and/or the DFR-Content of the appointed DFR-Object as appropriate; see 8.2.6;

3) returns the **Modify** result to the DFR-User; see 8.2.6.

10.7 Performance of the List abstract operation

When the DFR-Server receives a **List** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) returns the values of the DFR-Attributes of the members of the appointed DFR-Group or the elements of the appointed DFR-Search-Result-List as appropriate to the DFR-User.

10.8 Performance of the Search abstract operation

When the DFR-Server receives a **Search** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) searches for the DFR-Objects as specified;

3) if requested, stores the result of the search in the appointed DFR-Search-Result-List as appropriate;

4) returns the **Search** result to the DFR-User.

10.9 Performance of the Reserve abstract operation

When the DFR-Server receives a **Reserve** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) reserves the appointed DFR-Object as appropriate;

3) returns the **Reserve** result to the DFR-User

10.10 Performance of the Abandon abstract operation

When the DFR-Server receives an **Abandon** abstract operation from the DFR-User it performs the following steps:

1) checks that the DFR-User has the access rights as required;

2) abandons the appointed, previously invoked DFR abstract operation as appropriate; see 8.2.11;

3) forces the abstract-error **Abandoned** to be returned for the abstract operation which has been abandoned, see 8.2.11.

11. PORT REALIZATION

The Document Filing and Retrieval Port abstract service is realized on a one-to-one basis between abstract operations defined in this part of the Standard and the real operations in the Document Filing and Retrieval Service Element (DFRSE) specified in Part 2 of this Standard.

APPENDICES Part 1

Appendix A
Overview of Attribute mapping - ODA Document Profile to DFR

Names of attributes in ODA Document Profile	Names of Attributes in DFR attribute sets
title	Other-Title
subject	Subject
document-reference	User-Reference
document-type	Document-Type
keywords	Keywords
document-date-and-time	Document-Date-And-Time
creation-date-and-time	Creation-Date-And-Time
purge-date-and-time	Purge-Date-And-Time
revision-date-and-time	Revision-Date-And-Time
version-number	Version-Name
organizations	Organizations
preparers	Preparers
owners	Owners
authors	Authors
status	Status
user-specific-codes	User-Specific-Codes
superseded-documents	Superseded-Documents
references-to-other-documents	User-References-To-Other-Objects
number-of-pages	Number-Of-Pages
languages	Languages
document-architecture-class	Document-Architecture-Class

Appendix B
Formal Assignment of Object Identifiers

All Object Identifiers this Part 1 of this ECMA Standard assigns are formally assigned in the present Appendix using ASN.1. The specified values are cited in the ASN.1 modules of subsequent Appendices.

This Appendix is definitive for all values except those for ASN.1 modules of this Part of this Standard. The definitive assignments for those occur in the modules themselves.

```
DFRObjectIdentifiers {ISO identified-organization(3) idc-ecma(0012) standard(0) number(137)
part-1(1) modules(0) object-identifiers(0)}
DEFINITIONS ::=
BEGIN

   -- PROLOGUE --

EXPORTS EVERYTHING;

ID ::= OBJECT IDENTIFIER
id-dfr ID ::= {standard ECMA-137 part-1(1)}

-- Categories--

id-mod                      ID ::= {id-dfr 0}
id-ot                       ID ::= {id-dfr 1}
id-pt                       ID ::= {id-dfr 2}
id-dfr-oc                   ID ::= {id-dfr 3}
id-dfr-bas-att              ID ::= {id-dfr 4}
id-dfr-ext-att              ID ::= {id-dfr 5}
id-dfr-att-syn              ID ::= {id-dfr 6}

-- Modules--

id-mod-object-identifiers   ID ::= {id-mod 0}
id-mod-abstract-service     ID ::= {id-mod 1}
id-mod-basic-attributes     ID ::= {id-mod 2}
id-mod-extension-attributes ID ::= {id-mod 3}

-- Objects--

id-dfr-server               ID ::= {id-ot 0}
id-dfr-user                 ID ::= {id-ot 1}

-- Ports--

id-pt-dfr                   ID ::= {id-pt 0}
```

-- DFR Object Classes--

id-dfr-document	ID :: = {id-dfr-oc 0}
id-dfr-root-group	ID :: = {id-dfr-oc 1}
id-dfr-proper-group	ID :: = {id-dfr-oc 2}
id-dfr-internal-reference	ID :: = {id-dfr-oc 3}
id-dfr-external-reference	ID :: = {id-dfr-oc 4}
id-dfr-search-result-list	ID :: = {id-dfr-oc 5}

-- DFR-Basic-Attributes Identification--

id-dfr-basic-attributes	ID:: = {id-dfr-bas-att }

--Attribute Types--

id-att-dfr-upi	ID :: = {id-dfr-bas-att 0}
id-att-dfr-object-class	ID :: = {id-dfr-bas-att 1}
id-att-dfr-document-type	ID :: = {id-dfr-bas-att 2}
id-att-document-architecture-class	ID :: = {id-dfr-bas-att 3}
id-att-dfr-title	ID :: = {id-dfr-bas-att 4}
id-att-dfr-pathname	ID :: = {id-dfr-bas-att 5}
id-att-dfr-parent-identification	ID :: = {id-dfr-bas-att 6}
id-att-dfr-guarantee-qos	ID :: = {id-dfr-bas-att 7}
id-att-dfr-referent-deleted	ID :: = {id-dfr-bas-att 8}
id-att-dfr-membership-criteria	ID :: = {id-dfr-bas-att 9}
id-att-dfr-ordering	ID :: = {id-dfr-bas-att 10}
id-att-dfr-resource-limit	ID :: = {id-dfr-bas-att 11}
id-att-dfr-resource-used	ID :: = {id-dfr-bas-att 12}
id-att-dfr-number-of-group-members	ID :: = {id-dfr-bas-att 13}
id-att-version-name	ID :: = {id-dfr-bas-att 14}
id-att-dfr-previous-versions	ID :: = {id-dfr-bas-att 15}
id-att-dfr-next-versions	ID :: = {id-dfr-bas-att 16}
id-att-dfr-version-root	ID :: = {id-dfr-bas-att 17}
id-att-dfr-external-location	ID :: = {id-dfr-bas-att 18}
id-att-user-reference	ID :: = {id-dfr-bas-att 19}
id-att-user-reference-to-other-objects	ID :: = {id-dfr-bas-att 20}
id-att-dfr-attribute-create-date-and-time	ID :: = {id-dfr-bas-att 21}
id-att-dfr-content-create-date-and-time	ID :: = {id-dfr-bas-att 22}
id-att-dfr-created-by	ID :: = {id-dfr-bas-att 23}
id-att-dfr-attribute-modify-date-and-time	ID :: = {id-dfr-bas-att 24}
id-att-dfr-content-modify-date-and-time	ID :: = {id-dfr-bas-att 25}
id-att-dfr-attributes-modified-by	ID :: = {id-dfr-bas-att 26}
id-att-dfr-content-modified-by	ID :: = {id-dfr-bas-att 27}
id-att-document-date-and-time	ID :: = {id-dfr-bas-att 28}
id-att-dfr-reservation	ID :: = {id-dfr-bas-att 29}
id-att-dfr-reserved-by	ID :: = {id-dfr-bas-att 30}
id-att-dfr-access-list	ID :: = {id-dfr-bas-att 31}

-- DFR-Extension-Attributes Identification--

id-dfr-extension-attributes ID:: = {id-dfr-ext-att }

--Attribute Types--

id-att-other-titles ID :: = {id-dfr-ext-att 0}
id-att-subject ID :: = {id-dfr-ext-att 1}
id-att-document-type ID :: = {id-dfr-ext-att 2}
id-att-keywords ID :: = {id-dfr-ext-att 3}
id-att-creation-date-and-time ID :: = {id-dfr-ext-att 4}
id-att-purge-date-and-time ID :: = {id-dfr-ext-att 5}
id-att-revision-date-and-time ID :: = {id-dfr-ext-att 6}
id-att-organizations ID :: = {id-dfr-ext-att 7}
id-att-preparers ID :: = {id-dfr-ext-att 8}
id-att-owners ID :: = {id-dfr-ext-att 9}
id-att-authors ID :: = {id-dfr-ext-att 10}
id-att-status ID :: = {id-dfr-ext-att 11}
id-att-user-specific-codes ID :: = {id-dfr-ext-att 12}
id-att-superseded-documents ID :: = {id-dfr-ext-att 13}
id-att-numbers-of-pages ID :: = {id-dfr-ext-att 14}
id-att-languages ID :: = {id-dfr-ext-att 15}

--Attribute Syntaxes--

id-dfr-att-syn-int ID :: = {id-dfr-att-syn 1}
id-dfr-att-syn-bool ID :: = {id-dfr-att-syn 2}
id-dfr-att-syn-utc-time ID :: = {id-dfr-att-syn 3}
id-dfr-att-syn-case-ign ID :: = {id-dfr-att-syn 4}
id-dfr-att-syn-case-ign-list ID :: = {id-dfr-att-syn 5}

END *-- of DFR-Object-Identifiers --*

Appendix C
Formal Definition of the DFR Abstract-service

```
DFRAbstractService {iso identified organization (1) idc-ecma(0012) standard(0) number(137) part-1(1)
modules(0) abstract-service(1) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

   -- PROLOGUE --

EXPORTS EVERYTHING

IMPORTS

-- Abstract Service macros --

ABSTRACT-BIND, ABSTRACT-ERROR, ABSTRACT-OPERATION, ABSTRACT-UNBIND,
    OBJECT, PORT
     FROM AbstractServiceNotation { joint-iso-ccitt-mhs-motis(6) asdc(2) modules (0) notation(1)}

-- Object identifiers --

id-dfr-document, id-dfr-rootGroup, id-dfr-properGroup, id-dfr-internalReference, id-dfr-
externalReference, id-dfr-searchResultList, id-dfr-server, id-dfr-user, id-pt-dfr
     FROM DFRObjectIdentifiers {iso identified organization(3) idc-ecma(0012) standard(0)
      number(137) part-1(1) modules(0) object-identifiers(0)}

-- OSI Directory --

Attribute, AttributeType, AttributeValue, AttributeValueAssertion DistinguishedName
     FROM informationFramework { joint-iso-ccitt ds(5) modules(1) informationFramework(1) }

-- Referenced Data Transfer --

RDT-reference
     FROM RDT-reference-definition {ISO identified-organization(3) idc-ecma(0012) standard(0)
     number(137)}

-- DFR Abstract Objects --

dfr-server   OBJECT
                  PORTS { dfr-port [S]}
                  ::= id-dfr-server

dfr-user     OBJECT
                  PORTS { dfr-port [C] }
                  ::= id-dfr-user
```

```
-- Port types --

Dfr  PORT
        CONSUMER INVOKES {
                                Create,
                                Delete,
                                Copy,
                                Move,
                                Read,
                                Modify,
                                List,
                                Search,
                                Reserve,
                                Abandon     }
        SUPPLIER INVOKES { }
        ::= id-pt-dfr

  -- SPECIFICATION OF DFR-OBJECT DATA TYPES --

DfrObjectClass ::= ENUMERATED {
                    dfr-document          (0),
                    dfr-root-group        (1),
                    dfr-proper-group      (2),
                    dfr-internal-reference (3),
                    dfr-external-reference (4),
                    dfr-search-result-list (5) }

DfrEntry ::= SEQUENCE {
            attributes  [0] DfrEntryAttributeSet,
            content     [1] DfrObjectContent  }

DfrEntryAttributes ::= SET OF Attribute

DfrObjectContent ::= CHOICE {
    document-content            [0] DfrDocumentContent,
    root-group-content          [1] DfrGroupContent,
    proper-group-content        [2] DfrGroupContent,
    internal-reference-content  [3] DfrInternalReferenceContent,
    external-reference-content  [4] DfrExternalReferenceContent,
    search-result-list-content  [5] DfrSearchResultListContent }

DfrUniquePermanentIdentifier ::= OCTET STRING

DfrDocumentContent ::= EXTERNAL(WITH COMPONENTS     { ... ,
                                    direct-reference      PRESENT,
                                    indirect-reference    ABSENT,
                                    encoding (WITH COMPONENTS { ... ,
                                        arbitrary ABSENT })})
```

```
DfrInternalReferenceContent ::= RDT-reference
        (WITH COMPONENTS {
          ae-identifier  ABSENT,
          local-reference (WITH COMPONENTS { specific-reference } ),
          data-object-type (DfrObjectClassID (
                    id-dfr-document |
                    id-dfr-proper-group |
                    id-dfr-search-result-list ) ),
          quality-of-service (WITH COMPONENTS {
              qos-level (WITH COMPONENTS { ... , level-1 ABSENT } ),
              single-use-of-reference (FALSE)} ),
          token ABSENT   } )

DfrExternalReferenceContent ::= RDT-reference
        (WITH COMPONENTS {
          ae-identifier,
          local-reference ,
          data-object-type (DfrObjectClassID (
                    id-dfr-document |
                    id-dfr-root-group |
                    id-dfr-proper-group |
                    id-dfr-search-result-list ) ),
          quality-of-service (WITH COMPONENTS {
              qos-level (WITH COMPONENTS { ... , level-1 ABSENT } ),
              single-use-of-reference (FALSE)} ),
          token ABSENT   } )

DfrObjectClassID ::= OBJECT IDENTIFIER (
                    id-dfr-document |
                    id-dfr-root-group |
                    id-dfr-proper-group |
                    id-dfr-internal-reference |
                    id-dfr-external-reference |
                    id-dfr-search-result-list    )

QoS-Level ::= CHOICE {
    level-1       [0] IMPLICIT NULL,
    level-2       [1] IMPLICIT UTCTime,
    level-3       [2] IMPLICIT SEQUENCE {
      produce-time    UTCTime,
      fidelity-time   UTCTime }}

DfrGroupContent ::= SEQUENCE OF DfrUniquePermanentIdentifier
```

```
DfrSearchResultListContent :: = CHOICE {
        empty          NULL,
        produced       SEQUENCE {
          continuation              [0] ContinuationContext OPTIONAL,
          start-date-and-time       [1] UTCTime,
          end-date-and-time         [2] UTCTime,
          object-list               [3] DfrEntryList,
          ordering                  [4] OrderingRule OPTIONAL,
          search-domain             [5] SearchDomain,
          search-criteria           [6] SearchCriteria } }

Filter :: = CHOICE {
        item       [0] FilterItem,
        and        [1] SET OF Filter,
        or         [2] SET OF Filter,
        not        [3] Filter}

FilterItem :: = CHOICE {
     equality          [0] AttributeValueAssertion,
     substrings        [1] SEQUENCE {
                               type   AttributeType,
                               strings    SEQUENCE OF CHOICE {
                                  initial  [0] ANY,
                                  any      [1] ANY,
                                  final    [2] ANY }},
greaterOrEqual     [2] AttributeValueAssertion ,
lessOrEqual        [3] AttributeValueAssertion,
present            [4] AttributeType,
approximateMatch   [5] AttributeValueAssertion}
```

-- ABSTRACT BIND --

```
DfrBind :: = ABSTRACT-BIND
     TO {dfr-port[S]}
     BIND
       ARGUMENT       DfrBindArgument
       RESULT         DfrBindResult
       BIND-ERROR     DfrBindError

DfrBindArgument :: = SEQUENCE {
     credentials                [0] Credentials,
     retrieve-restrictions      [1] Restrictions OPTIONAL,
     dfr-configuration-request  [2] BOOLEAN DEFAULT FALSE,
     bind-security              [3] BindSecurity OPTIONAL,
     priority                   [4] Priority DEFAULT medium}
```

```
Credentials ::= CHOICE {
        simple      [0] Creds,
        certified   [1] PrivilegeAttributeCertificate }
        PrivilegeAttributeCertificate ::= EXTERNAL

Creds ::= OCTET STRING

PrivilegeAttributeCertificate ::= EXTERNAL

Restrictions ::= SET {
    allowed-document-types [0] SET OF OBJECT IDENTIFIER OPTIONAL
    maximum-result-length  [1] ResultLength OPTIONAL  }

ResultLength ::= INTEGER

BindSecurity ::= EXTERNAL

DfrBindResult ::= SET {
    authentication-attributes          [0] SET OF AuthenticationAttribute,
    available-attribute-types          [1] SET OF AttributeType OPTIONAL,
    constraints-supported              [2] SET OF ConstraintsType OPTIONAL,
    dfr-document-types-supported       [3] SET OF OBJECT IDENTIFIER OPTIONAL,
    function-set-supported             [4] FunctionSetType OPTIONAL
    maximum-result-length-supported [5] INTEGER OPTIONAL }

AuthenticationAttribute ::= EXTERNAL

ConstraintsType ::= ENUMERATED {
    global-unambiguity   (0),
    local-unambiguity    (1),
    version-unambiguity  (2) }

FunctionSetType ::= ENUMERATED {
flat-store           (0),
pre-defined-store    (1),
full-set             (3) }

DfrBindError ::= CHOICE {
        service-error    [0] ServiceProblem,
        security-error   [1] SecurityProblem }

DfrUnbind ::= ABSTRACT-UNBIND
    FROM {dfr-port[S]}

-- COMMON DATA TYPES FOR DFR-OPERATION ARGUMENTS --

CommonArguments    ::= SEQUENCE {
    reservation          [27] Reservation  OPTIONAL,
    error-handling       [28] ErrorHandlingMode DEFAULT all-or-nothing,
    priority             [29] Priority DEFAULT medium,
    privileges           [30] Privileges OPTIONAL  }
```

```
CommonResults        ::= SEQUENCE {
    warnings                [30] SEQUENCE OF Warning DEFAULT {} }

ErrorHandlingMode ::= CHOICE {
    all-or-nothing          [0] NULL,
    until-first-warning     [1] NULL,
    report-all-warnings     [2] NULL,
    report-n-warnings       [3] INTEGER }

Warning ::= SEQUENCE {
    entry       [0] DfrEntryName OPTIONAL,
    problem     [1] AccessProblem }

Priority ::= ENUMERATED {
        low         (0),
        medium      (1),
        high        (2) }

Privileges ::= SEQUENCE {
        operation-pac       [0] PrivilegeAttributeCertificate OPTIONAL,
        proxy-pac           [1] PrivilegeAttributeCertificate OPTIONAL}

DfrEntryName ::= CHOICE {
    upi                             [0] DfrUniquePermanentIdentifier,
    path-name                       [1] DfrPathName,
    relative-path-name              [2] SEQUENCE {
                        base            [0] DfrUniquePermanentIdentifier,
                        path            [1] DfrPathName } }

DfrPathName ::= SEQUENCE OF DfrTitle

DfrTitle ::= CharacterData

        CharacterData ::= CHOICE { GraphicString, T61String, PrintableString }

CommonUpdateArguments ::= SEQUENCE {
    object-class    [0] DfrObjectClass OPTIONAL,
    entry           [1] CHOICE {
        local               DfrEntryName,
        external        [3] ExternalReferenceContent} OPTIONAL,
    destination     [2] DfrEntryName OPTIONAL,
    position        [3] GroupMemberPosition OPTIONAL,
    modifications   [4] SEQUENCE OF EntryModification OPTIONAL,
    selection       [5] EntryInformationSelection OPTIONAL}

CommonUpdateResult ::= SEQUENCE {
    upi                 [0] DfrUniquePermanentIdentifier,
    entry-information   [1] EntryInformation OPTIONAL,
    COMPONENTS OF CommonResults }
```

```
GroupMemberPosition ::= CHOICE {
        last        [0] NULL,
        first       [1] NULL,
        after       [2] DfrEntryName,
        before      [3] DfrEntryName }

EntryModification ::= CHOICE{
     put-attribute          [0] Attribute,
     remove-attribute       [1] AttributeType,
     copy-attributes-from   [2] SEQUENCE {
             source                 [0] SourceEntry,
             attribute-selection    [1] SET OF AttributeType OPTIONAL },
     add-values             [3] Attribute,
     remove-values          [4] Attribute,
     add-values-from        [5] SEQUENCE {
             source                 [0] SourceEntry,
             attribute-selection    [1] SET OF AttributeType OPTIONAL },
     put-content            [6] DfrObjectContent,
     remove-content         [7] NULL,
     copy-content-from      [8] SourceEntry }

SourceEntry ::= CHOICE {
     parent             [0] NULL,
     referent           [1] NULL,
     previous-version   [2] NULL,
     specified-entry    [3] DfrEntryName,
     reference          [4] RDT-reference }

EntryInformationSelection ::= SEQUENCE {
     read-selector              [0] ENUMERATED {
        attributes-only                     (0),
        attributes-and-content              (1),
        content-only                        (2),
        rdt-ref-to-attr-only                (3),
        attr-and-rdt-ref-to-content         (4),
        rdt-ref-to-content-only             (5),
        rdt-ref-to-entire-object            (6),
        attr-and-rdt-ref-to-entire-object   (7) }
           DEFAULT attributes-only,
     attribute-selection    [1] AttributeSelection OPTIONAL }
```

```
AttributeSelection ::= CHOICE {
    all             [0] NULL,
    none            [1] NULL,
    unordered       [2] SET OF AttributeType,  --when the delivery order is unsignificant--
    ordered         [3] SEQUENCE OF AttributeType,
    minimum         [4] NULL }

EntryInformation ::= CHOICE {
    attributes-only                 [0] DfrEntryAttributes,
    attributes-and-content          [1] DfrEntry,
    content-only                    [2] DfrObjectContent,
    rdt-ref-to-attr-only            [3] RDT-reference,
    attr-and-rdt-ref-to-content     [4] SEQUENCE {
                attributes              [0] DfrEntryAttributes,
                rdt-ref-to-content      [1] RDT-reference },
    rdt-ref-to-content-only         [5] RDT-reference,
    rdt-ref-to-entire-object        [6] RDT-reference,
    attr-and-rdt-ref-to-entire-object [7] SEQUENCE {
                attributes              [0] DfrEntryAttributes,
                rdt-ref-to-entire-object [1] RDT-reference,     } }

CommonListSearchArguments ::= SEQUENCE {
    continuation    [1] ContinuationContext OPTIONAL,
    limits          [2] Limits OPTIONAL,
    selection       [3] AttributeSelection OPTIONAL,
    ordering        [4] OrderingRule OPTIONAL }

CommonListSearchResult ::= SEQUENCE {
    number-of-entries [0] INTEGER,
    continuation      [1] ContinuationContext  OPTIONAL,
    limit-encountered [2] LimitEncountered  OPTIONAL,
    entry-list        [3] DfrEntryList,
    COMPONENTS OF CommonResults }

ContinuationContext ::= OCTET STRING

Limits ::= SEQUENCE {
    time-limit      [0] INTEGER OPTIONAL,
    count-limit     [1] INTEGER OPTIONAL }

LimitEncountered ::= ENUMERATED {
      time-limit    (0),
      count-limit   (1) }
```

```
DfrEntryList ::= SEQUENCE OF SEQUENCE {
        upi                 [0] DfrUniquePermanentIdentifier,
        class               [1] DfrObjectClass,
        ordering-attribute  [2] Attribute OPTIONAL,
        other-attributes    [3] SEQUENCE OF Attribute OPTIONAL }

OrderingRule ::= CHOICE {
     ascending  [0] AttributeType,
     descending [1] AttributeType }

SearchDomain ::= SEQUENCE OF CHOICE {
        previous-result     [0] DfrEntryName,
        scope               [1] SEQUENCE {
                root                    [0] DfrEntryName,
                descent-depth           [1] INTEGER OPTIONAL,
                dereferencing-depth     [2] INTEGER DEFAULT 0 } }

SearchCriteria ::= Filter

-- ABSTRACT OPERATIONS --

Create ::= ABSTRACT-OPERATION
     ARGUMENT   CreateArgument
     RESULT     CreateResult
     ERRORS{    NameError,
                UpdateError,
                AttributeError,
                VersionManagementError,
                AccessError,
                ReferentAccessError,
                InterServerAccessError,
                SecurityError,
                ServiceError  }

CreateArgument ::= SEQUENCE {
     COMPONENTS OF
       CommonUpdateArguments (WITH COMPONENTS { ... ,
         object-class  PRESENT,
         destination   PRESENT } ),
     attributes   [7] SET OF Attributes OPTIONAL,
     content      [8] DfrObjectContent OPTIONAL,
     COMPONENTS OF CommonArguments (WITH COMPONENTS { ..., error-handling ABSENT })}

CreateResult ::= CommonUpdateResult
```

```
Delete ::= ABSTRACT-OPERATION
    ARGUMENT    DeleteArgument
    RESULT      DeleteResult
    ERRORS {    NameError,
                UpdateError,
                AccessError,
                SecurityError,
                ServiceError}

DeleteArgument ::= SEQUENCE {
    entry           [0] DfrEntryName,
    COMPONENTS OF CommonArguments ( WITH COMPONENTS { ...,
            reservation   ABSENT,
            error-handlingABSENT })}

DeleteResult ::= NULL

Copy ::= ABSTRACT-OPERATION
    ARGUMENT    CopyArgument
    RESULT      CopyResult
    ERRORS{     Abandoned,
                NameError,
                UpdateError,
                AttributeError,
                VersionManagementError,
                AccessError,
                ReferentAccessError,
                InterServerAccessError,
                SecurityError,
                ServiceError}

CopyArgument ::= SEQUENCE {
    COMPONENTS OF
      CommonUpdateArguments (WITH COMPONENTS { ... ,
        entry         PRESENT,
        destination   PRESENT } ),
    COMPONENTS OF CommonArguments }

CopyResult ::= CommonUpdateResult
```

```
Move :: = ABSTRACT-OPERATION
    ARGUMENT    MoveArgument
    RESULT      MoveResult
    ERRORS{     Abandoned,
                NameError,
                UpdateError,
                AttributeError,
                VersionManagementError,
                AccessError,
                ReferentAccessError,
                InterServerAccessError,
                SecurityError,
                ServiceError}

MoveArgument :: = SEQUENCE {
    COMPONENTS OF
        CommonUpdateArguments (WITH COMPONENTS { ... ,
          entry          PRESENT,
          destination    PRESENT }),
    COMPONENTS OF CommonArguments (WITH COMPONENTS { ... ,
          error-handlin  ABSENT })}

MoveResult :: = CommonUpdateResult

Read :: = ABSTRACT-OPERATION
    ARGUMENT    ReadArgument
    RESULT      ReadResult
    ERRORS{     Abandoned,
                NameError,
                AccessError,
                ReferentAccessError,
                InterServerAccessError,
                SecurityError,
                ServiceError}

ReadArgument  :: = SEQUENCE {
    COMPONENTS OF
      CommonUpdateArguments (WITH COMPONENTS
          { entry, selection } ),
    dereferencing      [7] BOOLEAN DEFAULT FALSE,
    COMPONENTS OF CommonArguments (WITH COMPONENTS { ..., error-handling ABSENT })}

ReadResult :: = CommonUpdateResult
    (WITH COMPONENTS { ... , entryInformation PRESENT } )
```

```
Modify :: = ABSTRACT-OPERATION
    ARGUMENT    ModifyArgument
    RESULT      ModifyResult
    ERRORS {    Abandoned,
                NameError,
                UpdateError,
                AttributeError,
                VersionManagementError,
                AccessError,
                ReferentAccessError,
                InterServerAccessError,
                SecurityError,
                ServiceError}

ModifyArgument :: = SEQUENCE {
    COMPONENTS OF
      CommonUpdateArguments (WITH COMPONENTS { ... ,
        entry          PRESENT,
        destination    ABSENT,
        position       ABSENT,
        modifications  PRESENT } ),
    COMPONENTS OF CommonArguments (WITH COMPONENTS { ..., error-handling ABSENT })}

ModifyResult  :: = CommonUpdateResult

List :: = ABSTRACT-OPERATION
    ARGUMENT    ListArgument
    RESULT      ListResult
    ERRORS {    Abandoned,
                NameError,
                AccessError,
                AttributeError,
                SecurityError,
                ServiceError}

ListArgument :: = SEQUENCE {
    entry[0] DfrEntryName,
    COMPONENTS OF
      CommonListSearchArguments (WITH COMPONENTS { ... ,
        selection PRESENT } ),
    COMPONENTS OF CommonArguments }

ListResult :: = CommonListSearchResult
```

```
Search :: = ABSTRACT-OPERATION
    ARGUMENT    SearchArgument
    RESULT      SearchResult
    ERRORS {    Abandoned,
                NameError,
                AccessError,
                AttributeError,
                UpdateError,
                SecurityError,
                ServiceError   }

SearchArgument :: = SEQUENCE {
    search-mode          [0] CHOICE {
      continue                  [0] DfrEntryName,
      update                    [1] DfrEntryName,
      new-search-stored         [2] DfrEntryName,
      non-stored-search         [3] NULL }
    COMPONENTS OF
      CommonListSearchArguments,
      search-domain  [5] SearchDomain OPTIONAL,
      search-criteria  [6] SearchCriteria OPTIONAL,
    COMPONENTS OF CommonArguments } (WITH COMPONENTS { ...,
          error-handling ABSENT })}

SearchResult :: = SEQUENCE {
      COMPONENTS OF CommonListSearchResult,
      removed-entries          [4] DfrEntryList OPTIONAL }

Reserve :: = ABSTRACT-OPERATION
    ARGUMENT    ReserveArgument
    RESULT      ReserveResult
    ERRORS {    NameError,
                ReservationError,
                SecurityError,
                ServiceError }

ReserveArgument :: = SEQUENCE {
    entry           [0] DfrEntryName,
    COMPONENTS OF CommonArguments ( WITH COMPONENTS { ...,
          reservation    PRESENT,
          error-handlingABSENT })}

Reservation :: = ENUMERATED {
    unreserved        (0),
    exclusive-write   (1),
    exclusive-access  (2) }
```

```
ReserveResult ::= NULL

Abandon ::= ABSTRACT-OPERATION
    ARGUMENT        AbandonArgument,
    RESULT          AbandonResult
    ERRORS          { AbandonFailed }

AbandonArgument ::= SEQUENCE {
    object  [0] DfrEntryName    }

AbandonResult ::= NULL

    -- ABSTRACT-ERRORS --

AttributeError ::= ABSTRACT-ERROR
    PARAMETER SEQUENCE {
      entry     [0] DfrEntryName OPTIONAL,
      problems [1] SEQUENCE OF
        SEQUENCE {
            problem  [0] AttributeProblem,
            type     [1] AttributeType,
            value    [2] AttributeValue OPTIONAL }}

AttributeProblem ::= ENUMERATED {
    no-such-attribute                (1),
    invalid-attribute-syntax         (2),
    undefined-attribute-type         (3),
    inappropriate-matching           (4),
    constraint-violation             (5),
    attribute-or-value-already-exists (6),
    illegal-modification             (7),
    inconsistent-with-other-attributes (8),
    undefined-for-this-object-class  (9),
    unsupported-document-type        (10)   }

NameError ::= ABSTRACT-ERROR
    PARAMETER SEQUENCE OF SEQUENCE {
      entry     [0] DfrEntryName,
      problem   [1] NameProblem }

NameProblem ::= ENUMERATED {
    invalid-upi              (1),
    invalid-path-name        (2),
    ambiguous-path-name      (3),
    inappropriate-object-class (4) }
```

```
AccessError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE OF SEQUENCE {
      entry     [0] DfrEntryName,
      problem   [1] AccessProblem }

AccessProblem :: = ENUMERATED {
    inappropriate-object-class    (1),
    insufficient-access-rights    (2),
    reserved-by-another-user      (3),
    externally-located-object     (4)  }

UpdateError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE {
      entry     [0] DfrEntryName,
      problem   [1] UpdateProblem }

UpdateProblem :: = ENUMERATED {
    inappropriate-object-class          (1),
    insufficient-access-rights          (2),
    reserved-by-another-user            (3),
    quality-of-service-violation        (4),
    illegal-content-modification        (5),
    group-membership-criteria-violation (6),
    reference-loop-detected             (7)  }

ReferentAccessError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE {
      entry     [0] DfrEntryName,
      problem   [1] ReferentAccessProblem }

ReferentAccessProblem :: = ENUMERATED {
    inappropriate-object-class          (1),
    insufficient-access-rights          (2),
    reserved-by-another-user            (3),
    referent-no-more-exists             (4),
    referent-modified                   (5),
    reference-content-empty             (6)  }

InterServerAccessError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE {
      entry     [0] DfrEntryName,
      problem   [1] InterServerAccessProblem }

InterServerAccessProblem :: = ENUMERATED {
    referent-store-not-found            (1),
    referent-store-unreachable          (2),
    referent-store-unavailable          (3),
    referent-store-security-problem     (4)  }
```

```
ReservationError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE{
      entry     [0] DfrEntryName,
      problem   [1] ReservationProblem }

ReservationProblem :: = ENUMERATED {
    cannot-reserve        (0),
    already-reserved      (1),
    not-yet-reserved      (2) }

VersionManagementError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE {
      entry     [0] DfrEntryName,
      problem   [1] VersionManagementProblem   }

VersionManagementProblem :: = ENUMERATED {
    inappropriate-object-class                (1),
    insufficient-access-rights                (2),
    belongs-to-another-conceptual-document    (3)   }

SecurityError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE {
      problem   [0] SecurityProblem }

SecurityProblem :: = ENUMERATED {
    inappropriate-authentication (1),
    invalid-creds             (2),
    invalid-privilege         (3),
    insufficient-access-rights (4),
    invalid-pac               (5)   }

ServiceError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE {
      problem   [0] ServiceProblem }

ServiceProblem :: = ENUMERATED {
    server-busy               (1),
    server-unavailable        (2),
    operation-too-complex     (3),
    resource-limit-exceeded   (4),
    unclassified-server-error (5)}

AbandonFailed :: = ABSTRACT-ERROR
    PARAMETER SET {
      problem   [0] AbandonProblem,
      operation [1] Invokeld }

Invokeld :: = INTEGER
```

```
AbandonProblem ::= ENUMERATED {
    no-such-operation       (1),
    too-late                (2),
    cannot-abandon          (3) }
Abandoned ::= ABSTRACT-ERROR

END -- of DFR-Abstract-Service --
```

Appendix D
Formal Definition of DFR-Basic-Attribute-Set

This Appendix, a supplement to clause 9, formally defines the attribute-types of the DFR-Basic-Attribute-Set, and the related attribute-syntax for each attribute-type, applicable for Document Filing and Retrieval. It employs ASN.1 and the ATTRIBUTE and ATTRIBUTE SYNTAX macros.

```
DFRBasicAttributes {iso identified-organization(3) idc-ecma(0012) standard(0) number(137) part-1(1)
modules(0) basic-attributes(2)}
DEFINITIONS ::= BEGIN

   -- PROLOGUE --

EXPORTS EVERYTHING;

IMPORTS
-- DFR Object Identifiers --
id-att-dfr-upi, id-att-dfr-object-class, id-att-dfr-document-type, id-att-document-architecture-class, id-att-
dfr-title, id-att-dfr-pathname, id-att-dfr-parent-identification, id-att-dfr-guarantee-qos, id-att-dfr-referent-
deleted, id-att-dfr-membership-criteria, id-att-dfr-ordering, id-att-dfr-resource-limit, id-att-dfr-resource-
used, id-att-dfr-number-of-group-members, id-att-version-name, id-att-dfr-previous-versions, id-att-dfr-
next-versions, id-att-dfr-version-root, id-att-dfr-external-location, id-att-user-reference, id-att-user-
reference-to-other-objects, id-att-dfr-attribute-create-date-and-time, id-att-dfr-content-create-date-and-
time, id-att-dfr-created-by, id-att-dfr-attribute-modify-date-and-time, id-att-dfr-content-modify-date-and-
time, id-att-dfr-attributes-modified-by, id-att-dfr-content-modified-by, id-att-document-date-and-time, id-
att-dfr-reservation, id-att-dfr-reserved-by, id-att-dfr-access-list, id-dfr-att-syn-int, id-dfr-att-syn-bool, id-
dfr-att-syn-utc-time, id-dfr-att-syn-case-ign, id-dfr-att-syn-case-ign-list

    FROM DFRObjectIdentifiers {iso identified-organization(3) idc-ecma(0012) standard(0)
    number(137) part-1(1) modules(0) object-identifiers(0)}

-- Attribute macros --

ATTRIBUTE, ATTRIBUTE-SYNTAX, DistinguishedName
    FROM informationFramework { joint-iso-ccitt ds(5) modules(1) informationFramework(1) }

-- Data Types from DFR-Abstract-Service --

DfrUniquePermanentIdentifier, QoS-Level, DfrObjectClass, DfrPathName, DfrOrderingRule, Filter,
Reservation
    From DFRAbstractService {iso identified-organization(3) idc-ecma(0012) standard(0)
    number(137) part-1(1) modules (0) abstract-service (1) }

   -- DFR-Basic-Attribute-Set --

dfr-upi ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::= id-att-dfr-upi
```

```
dfr-object-class ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DfrObjectClass
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::= id-att-dfr-object-class

dfr-document-type ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX OBJECT IDENTIFIER
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::= id-att-dfr-document-type

document-architecture-class ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DocumentArchitectureClass
        MATCHES FOR EQUALITY ORDERING
        SINGLE VALUE
        ::= id-att-document-architecture-class

dfr-title ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
        SINGLE VALUE
        ::= id-att-dfr-title

dfr-pathname ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreListSyntax
        SINGLE VALUE
        ::= id-att-dfr-pathname

dfr-parent-identification ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier
        SINGLE VALUE
        ::= id-att-dfr-parent-identification

dfr-guarantee-qos ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX QoS-Level
        SINGLE VALUE
        ::= id-att-dfr-guarantee-qos

dfr-referent-deleted ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX booleanSyntax
        SINGLE VALUE
        ::= id-att-dfr-referent-deleted

dfr-membership-criteria ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX Filter
        SINGLE VALUE
        ::= id-att-dfr-membership-criteria
```

```
dfr-ordering ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX OrderingRule
        SINGLE VALUE
        ::= id-att-dfr-ordering

dfr-resource-limit ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX integerSyntax
        SINGLE VALUE
        ::= id-att-dfr-resource-limit

dfr-resource-used ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX integerSyntax
        SINGLE VALUE
        ::= id-att-dfr-resource-used

dfr-number-of-group-members ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX integerSyntax
        SINGLE VALUE
        ::= id-att-dfr-number-of-group-members

version-name ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX CaseIgnoreStringSyntax
        SINGLE VALUE
        ::= id-att-version-name

dfr- previous-versions ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier
        MATCHES FOR EQUALITY
        MULTI VALUE
        ::= id-att-dfr-previous-versions

dfr-next-versions ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier
        MATCHES FOR EQUALITY
        MULTI VALUE
        ::= id-att-dfr-next-versions

dfr-version-root ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DfrUniquePermanentIdentifier
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::= id-att-dfr-version-root

dfr-external-location ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
        SINGLE VALUE
        ::= id-att-dfr-external-location
```

```
user-reference ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX CaseIgnoreStringSyntax
        SINGLE VALUE
        ::= id-att-user-reference

user-reference-to-other-objects ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX CaseIgnoreStringSyntax
        MULTI VALUE
        ::= id-att-user-reference-to-other-objects

dfr-attributes-create-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
        SINGLE VALUE
        ::= id-att-dfr-attributes-create-date-and-time

dfr-content-create-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
        SINGLE VALUE
        ::= id-att-dfr-content-create-date-and-time

dfr-created-by ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DistinguishedName
        SINGLE VALUE
        ::= id-att-dfr-created-by

dfr-attributes-modify-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
        SINGLE VALUE
        ::= id-att-dfr-attributes-modify-date-and-time

dfr-content-modify-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
        SINGLE VALUE
        ::= id-att-dfr-content-modify-date-and-time

dfr-attribute-modified-by ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DistinguishedName
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::= id-att-dfr-attribute-modified-by

dfr-content-modified-by ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DistinguishedName
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::= id-att-dfr-content-modified-by
```

```
document-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
        SINGLE VALUE
        ::= id-att-document-date-and-time

dfr-reservation ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX Reservation
        MATCHES FOR EQUALITY ORDERING
        SINGLE VALUE
        ::= id-att-dfr-reservation

dfr-reserved-by ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DistinguishedName
        MATCHES FOR EQUALITY
        SINGLE VALUE
        ::= id-att-dfr-reserved-by

dfr-access-list ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX DfrAccessListElement
        MATCHES FOR EQUALITY
        MULTI VALUE
        ::= id-att-dfr-access-list

  -- Attribute Syntaxes --

DocumentArchitectureClass ::= INTEGER {
        formatted                (0),
        processable              (1),
        formatted-processable    (2) }

          DfrAccessListElement ::= SEQUENCE {
                accessId       AccessId,
                accessRights   AccessRights }

            AccessId ::= DistinguishedName

          AccessRights ::= ENUMERATED {
            read               (0),
            extendedRead       (1),
            readModify         (2),
            readModifyDelete   (3),
            owner              (4) }

          integerSyntax      ATTRIBUTE-SYNTAX-
                INTEGER
                MATCHES FOR EQUALITY ORDERING
                ::= {id-dfr-att-syn1}
```

```
        booleanSyntax       ATTRIBUTE-SYNTAX
                BOOLEAN
                MATCHES FOR EQUALITY
                ::= {id-dfr-att-syn2}

        uTCTimeSyntax       ATTRIBUTE-SYNTAX
                UTCTime
                MATCHES FOR EQUALITY ORDERING
                ::= {id-dfr-att-syn3}

        caseIgnoreStringSyntax   ATTRIBUTE-SYNTAX
                CharacterData
                MATCHES FOR EQUALITY SUBSTRINGS
                ::= {id-dfr-att-syn4}

        caseIgnoreListSyntax  ATTRIBUTE-SYNTAX
                SEQUENCE OF
                CharacterData
                MATCHES FOR EQUALITY SUBSTRINGS
                ::= {id-dfr-att-syn5}

END -- of DFR-Basic -Attribute-Set--
```

Appendix E
Formal Definition of DFR-Extension-Attribute-Set

This Appendix, a supplement to clause 9, formally defines the attribute-types of the DFR-Extension-Attribute-Set, and the related attribute-syntax for each attribute-type, applicable for Document Filing and Retrieval. It employs ASN.1 and the ATTRIBUTE and ATTRIBUTE SYNTAX macro, see 6.8.7.4.

```
DFRExtensionAttributes {iso identified-organization(3) idc-ecma(0012) standard(0) number(137)
part-1(1) modules(0) extension-attributes(3)}
DEFINITIONS ::= BEGIN

   -- PROLOGUE --

EXPORTS EVERYTHING;

IMPORTS

uTCTimeSyntax, caseIgnoreStringSyntax
      FROM DFRBasicAttributes {iso identified-organization(3) idc-ecma(0012) standard(0)
      number(137) part-1(1) modules(0) basic-attributes(2)}

-- DFR Object Identifiers --
id-att-other-titles, id-att-subject, id-att-document-type, id-att-keywords, id-att-creation-date-and-time,
id-att-purge-date-and-time, id-att-revision-date-and-time, id-att-organizations, id-att-preparers, id-att-
owners, id-att-authors, id-att-status, id-att-user-specific-codes, id-att-superseded-documents, id-att-
numbers-of-pages, id-att-languages
      FROM DFRObjectIdentifiers {iso identified-organization(3) idc-ecma(0012) standard(0)
      number(137) part-1(1) part-1(1) modules(0) object-identifiers(0) }

-- Attribute macros --
ATTRIBUTE, ATTRIBUTE-SYNTAX
      FROM informationFramework { joint-iso-ccitt ds(5) modules(1) informationFramework(1) }

   -- DFR-Extension-Attribute-Set --

other-titles ATTRIBUTE
         WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
         MULTI VALUE
         ::= id-att-other-titles

subject ATTRIBUTE
         WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
         SINGLE VALUE
         ::= id-att-subject

document-type ATTRIBUTE
         WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
         SINGLE VALUE
         ::= id-att-document-type
```

```
keywords ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
        MULTI VALUE
        ::= id-att-keywords

creation-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
        SINGLE VALUE
        ::= id-att-creation-date-and-time

purge-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
        SINGLE VALUE
        ::= id-att-purge-date-and-time

revision-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX uTCTimeSyntax
        SINGLE VALUE
        ::= id-att-revision-date-and-time

organizations ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
        MULTI VALUE
        ::= id-att-organizations

preparers ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX Person
        MULTI VALUE
        ::= id-att-preparers

owners ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX Person
        MULTI VALUE
        ::= id-att-owners

authors ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX Person
        MULTI VALUE
        ::= id-att-authors

status ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
        SINGLE VALUE
        ::= id-att-status

user-specific-codes ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
        MULTI VALUE
        ::= id-att-user-specific-codes
```

```
superseded-documents ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
        MULTI VALUE
        :: = id-att-superseded-documents

number-of-pages ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX integerSyntax
        SINGLE VALUE
        :: = id-att-number-of-pages

languages ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax
        MULTI VALUE
        :: = id-att-languages

-- Attribute Syntaxes --

Person :: = SEQUENCE {
        surname      [0] IMPLICIT CharacterData OPTIONAL,
        givenname    [1] IMPLICIT CharacterData OPTIONAL,
        initials     [2] IMPLICIT CharacterData OPTIONAL,
        title        [3] IMPLICIT CharacterData OPTIONAL,
        organization [4] IMPLICIT Character-Data OPTIONAL }

END -- of DFR-Extension-Attribute-Set--
```

Part 2
Protocol Specification

SECTION ONE - DFR ACCESS PROTOCOL SPECIFICATION

1. OVERVIEW OF THE PROTOCOL

1.1 DFR Access Protocol Model

Part 1 of this Standard ECMA-137 describes an abstract model of the Document Filing and Retrieval Application, and the DFR Abstract Service which is provided to the DFR-User.

This clause describes how the DFR Abstract Service is supported by instances of OSI communication when an abstract-service user and an abstract-service provider are realized as application-process located in different open systems.

In the OSI environment, communication between application-processes is represented in terms of communication between a pair of application-entities (AEs) using the presentation-service. The functionality of an application-entity is factored into a set of one or more application-service-elements (ASEs). The interaction between AEs is described in terms of their use of the services provided by the ASEs.

Access to the DFR Abstract Service is supported by the DFR Service Element (DFRSE), supporting a port paired between a DFR-User and the DFR-Server in the abstract model. The DFR Service Element is an asymmetric ASE, that is, the DFR-User acts as the consumer, and the DFR-Server acts as the supplier, of the DFR Abstract Service.

The DFRSE is in turn supported by other application-service-elements.

The Remote Operations Service Element (ROSE) supports the request/reply paradigm of the abstract operations that occur at the DFR-Port in the abstract model. The DFRSE provides the mapping function of the abstract-syntax notation of this abstract-service onto the services provided by the ROSE.

Optionally, the Reliable Transfer Service Element (RTSE) may be used to reliably transfer the application-protocol-data-units (APDUs) that contain the parameters of the operations between AEs.

The Association Control Service Element (ACSE) supports the establishment and release of an application-association between a pair of AEs. Associations between a DFR-User and the DFR-Server may be established only by the DFR-User, and only the initiator of an established association can release it.

The combination of the DFRSE together with the supporting ASEs, defines the application-context of an application-association. Note that a single application-association may be used to support one or more port types paired between two objects in the abstract model.

Figure 1 models an application-context between a DFR-User and and a DFR-Server. The consumer role of the DFR-User ASE and the supplier role of the DFR-Server ASE, is indicated by the subscript "c" or "s".

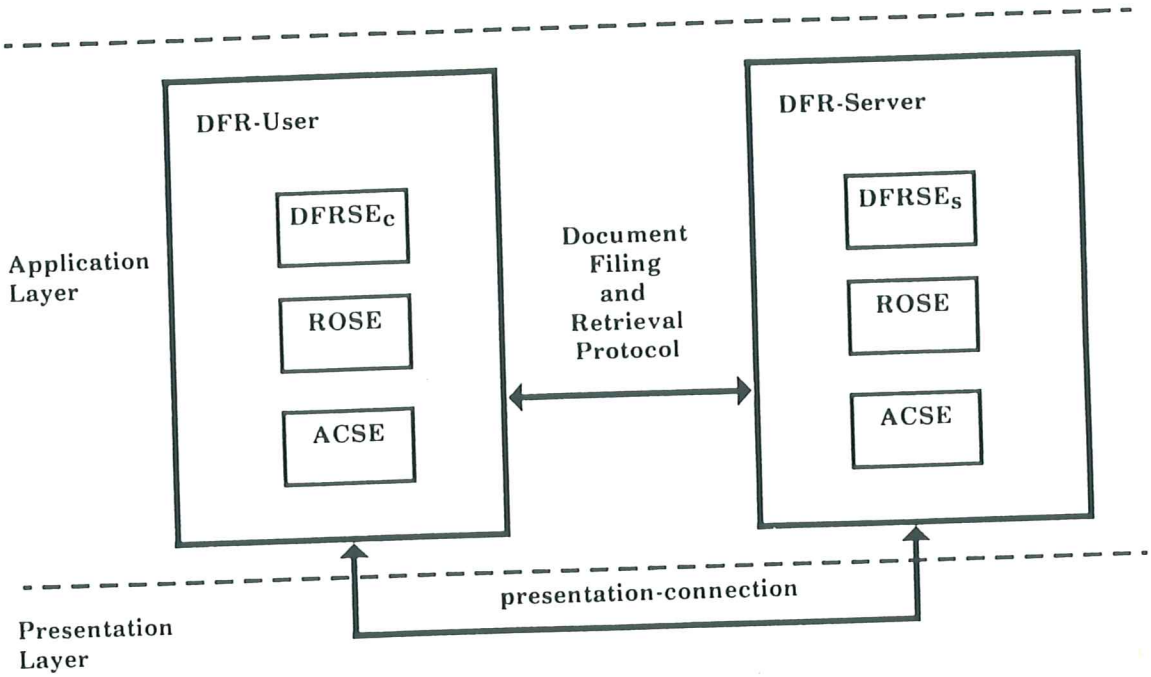

Fig. 1 - The DFR Model

1.2 Services Provided by the DFR Access Protocol

The DFR Access Protocol comprises the following operations which provide the services defined in Part 1 of this Standard.

DFR-bind and DFR-unbind

- DFR-bind
- DFR-unbind

DFR Service Element (DFRSE)

- Create
- Delete
- Copy
- Move
- Read
- Modify
- List
- Search
- Reserve
- Abandon

The DFR Access Protocol makes use of underlying services as defined in ISO/IEC 9072-1.

2. DFR ACCESS PROTOCOL ABSTRACT SYNTAX DEFINITION

The abstract-syntax of the DFR Access Protocoll is defined using the abstract syntax notation (ASN.1) defined in ISO 8824, and the remote operations notation defined in ISO/IEC 9072-1.

```
DFRAccessProtocol {iso identified-organization(3) idc-ecma(0012) standard(0) number(137)
part-2(2) modules(0) access-protocol(1)}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
-- PROLOGUE --
EXPORTS
-- DFR  ASE --
dFRSE;
IMPORTS
-- Application Service Elements and Application Contexts --
APPLICATION-SERVICE-ELEMENT, APPLICATION-CONTEXT, aCSE
FROM Remote-Operations-Notation-extension { joint-iso-ccitt
  remote-operations(4) notation-extension(2)}
rTSE
  FROM Reliable-Transfer-APDUs { joint-iso-ccitt reliable-transfer(3) apdus(0) }
  -- DFR Abstract Service Parameters --
DfrBind, DfrUnbind, Create, Delete, Copy, Move, Read, Modify, List, Search, Reserve, Abandon,
AttributeError, NameError, AccessError, UpdateError, ReferentAccessError,
InterServerAccessError, ReservationError, VersionManagementError, SecurityError, ServiceError,
AbandonFailed, Abandoned
  FROM DFRAbstractService {iso identified-organization(3) idc-ecma(0012) standard(0)
  number(137) part-1(1) modules(0) abstract-service(1) }
-- Object Identifiers --
id-ac-dfr-access, id-as-acse, id-as-dfrse, id-as-rdtse, id-ase-dfrse
  FROM DFRProtocolObjectIdentifiers{iso identified-organization(3) idc-ecma(0012) standard(0)
  number(137) part-2(2) modules(0) object-identifiers(0)}
-- Application Context Without RTSE --
dfr-access APPLICATION-CONTEXT
        APPLICATION SERVICE ELEMENTS { aCSE }
        BIND DfrBind
        UNBIND DfrUnbind
        REMOTE OPERATIONS { rOSE }
        INITIATOR CONSUMER OF    {dFRSE}
        ABSTRACT SYNTAXES {
          id-as-acse,           -- of ACSE --
          id-as-dfrse           -- of DFRSE, including ROSE and the DFR-Basic-Attribute-Set--
          id-as-dfr-ext-attr }  -- optional for the DFR-Extension Attribute-Set. Other abstract--
                                -- syntax names for other extension attribute sets may be negotiated--
      ::= id-ac-dfr-access
```

```
-- Application Context including RTSE --
dfr-reliable-access  APPLICATION-CONTEXT
                     APPLICATION SERVICE ELEMENTS { aCSE,  rTSE }
                     BIND DfrBind
                     UNBIND DfrUnbind
                     REMOTE OPERATIONS    {rOSE}
                     INITIATOR CONSUMER OF{dFRSE}
        ABSTRACT SYNTAXES {
             id-as-acse,        -- of ACSE --
             id-as-dfrse        -- of DFRSE, including ROSE and the DFR-Basic-Attribute-Set--
             id-as-dfr-ext-attr }      -- optional for the DFR-Extension Attribute-Set. Other abstract--
                                -- syntax names for other extension attribute sets may be negotiated--
        ::= id-ac-dfr-reliable-access

id-as-acse ::= aCSE-as
aCSE-as OBJECT IDENTIFIER ::= {
        joint-iso-ccitt association-control (2) abstractSyntax (1) apdus (0) version1 (1)}
                                -- as defined in ISO 8650--

-- DFR Service Element --
dFRSE APPLICATION-SERVICE-ELEMENT
        CONSUMER INVOKES {
          create,
          delete,
          copy,
          move,
          read,
          modify,
          list,
          search,
          reserve,
          abandon }
        SUPPLIER INVOKES { }
        ::= id-ase-dfrse

-- Remote Operations --
```

```
create          Create          ::= 01
delete          Delete          ::= 02
copy            Copy            ::= 03
move            Move            ::= 04
read            Read            ::= 05
modify          Modify          ::= 06
list            List            ::= 07
search          Search          ::= 08
reserve         Reserve         ::= 09
abandon         Abandon         ::= 10

-- Remote Errors --

attribute-error             AttributeError          ::= 01
name-error                  NameError               ::= 02
access-error                AccessError             ::= 03
update-error                UpdateError             ::= 04
referent-access-error       ReferentAccessError     ::= 05
inter-server-access-error   InterServerAccessError  ::= 06
reservation-error           ReservationError        ::= 07
version-management-error    VersionManagementError  ::= 08
security-error              SecurityError           ::= 09
service-error               ServiceError            ::= 10
abandon-failed              AbandonFailed           ::= 11
abandoned                   Abandoned               ::= 12

END -- of DFRAccessProtocol --
```

3. CONFORMANCE

A DFR system claiming conformance to the DFR Access Protocol specified in this Standard shall comply with the requirements noted below.

3.1 Static Requirements

The system shall

- conform to the abstract-syntax definition(s) of the DFR Access Protocol defined in 4 of this part of this Standard;
- conform to the Basic-Attribute-Set defined in Part 1 of this Standard;
- conform to the application-context defined in 4 of this part of this Standard.

3.2 Dynamic Requirements

The system shall

- supply operations conforming to clauses 8 and 10 of part 1 of this Standard;
- conform to the mapping onto used services, required by the application-context defined in 4 of this part of this Standard;
- conform to the use of underlying services as defined in ISO/IEC 9072-1.

Appendix A
Formal Assignment of Object Identifiers

All Object Identifiers this Part 2 of this Standard assigns are formally assigned in the present Appendix using ASN.1. The specified values are cited in the ASN.1 module of this Part of this Standard.

This Appendix is definitive for all values except those for ASN.1 modules of this Part of this Standard. The definitive assignments for those occur in the modules themselves.

```
DFRProtocolObjectIdentifiers {iso identified-organization(3) idc-ecma(0012) standard(0)
number(137) part-2(2) modules (0) object-identifiers(0)}
DEFINITIONS ::=
BEGIN

   -- PROLOGUE --

EXPORTS EVERYTHING;

IMPORTS  -- nothing -- ;

-- DFR Protocol --

ID ::= OBJECT IDENTIFIER

id-dfr-protocol    ID ::= { standard ECMA-137 part-2(2) }

-- Categories --

id-mod                    ID ::=  { id-dfr-protocol 0 }
id-ac                     ID ::=  { id-dfr-protocol 1 }
id-as                     ID ::=  { id-dfr-protocol 2 }
id-ase                    ID ::=  { id-dfr-protocol 3 }

-- Modules --

id-mod-object-identifiers  ID ::= { id-mod 0 }
id-mod-access-protocol     ID ::= { id-mod 1 }

-- Application Context --

id-ac-dfr-access           ID ::= { id-ac 0 }
id-ac-dfr-reliable-access  ID ::= { id-ac 1 }

-- Abstract Syntaxes --

id-as-acse                 ID ::= { id-as 0 }
id-as-dfrse                ID ::= { id-as 1 }
id-as-dfr-ext-attr         ID ::= {id-as 2 }

-- Application Service Element --

id-ase-dfrse               ID ::= {id-ase 0}

END
```