INTERNATIONAL STANDARD

ISO/IEC 10166-1

First edition
1991-12-15

Information technology — Text and office systems — Document Filing and Retrieval (DFR) —

Part 1:
Abstract service definition and procedures

Technologies de l'information — Bureautique — Classement et récupération de documents —

Partie 1: Procédures et définition de service abstrait

Reference number
ISO/IEC 10166-1:1991(E)

Table of Contents

Annexes

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 10166-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

ISO/IEC 10166 consists of the following parts, under the general title *Information technology — Text and office systems — Document Filing and Retrieval (DFR)*:

— *Part 1: Abstract service definition and procedures*

— *Part 2: Protocol specification*

Annexes B, C, D and E form an integral part of this part of ISO/IEC 10166.

Annexes A and F are for information only.

Introduction

The Document Filing and Retrieval (DFR) application 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 one of a series of International Standards defining applications needed in the area of office automation, as described in the Distributed-office-application model [ISO/IEC 10031-1]. ISO/IEC 10166 provides the functionality of document filing and retrieval which directly supports the user in an office environment. This 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.

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.

NOTES

1 ISO/IEC 10166 deals with individual Document Filing and Retrieval servers, it defines the Document Filing and Retrieval (DFR-) protocol. This International 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 extensible and support this future work.

2 ISO/IEC 10166 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.

This page intentionally left blank

Information technology — Text and office systems — Document Filing and Retrieval (DFR) — Part 1: Abstract service definition and procedures

Section 1: General

1 Scope

This part of ISO/IEC 10166 specifies the Document Filing and Retrieval Abstract Service that enables a user to communicate with a remote Document Filing and Retrieval server (DFR-Server) in order to access a remote document store.

This part of ISO/IEC 10166

- specifies a client-server type model in accordance with the Distributed-Office-Application Model [ISO/IEC 10031-1];
- specifies functions and services provided by Document Filing and Retrieval servers;
- specifies a specific Document Filing and Retrieval model for managing documents and groups of documents;
- specifies the Document Filing and Retrieval Abstract Service using the principles established by the Abstract Service Definition Conventions (ISO/IEC 10021-3);
- specifies the usage of other services.

ISO/IEC 10166 serves the following important fields of application:

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

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 10166. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 10166 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 7498:1984, *Information processing systems - Open Systems Interconnection - Basic Reference Model.*

ISO/IEC 7498-2:1989, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture.*

ISO 8613-1:1989, *Information processing - Text and office systems - Office Document Architecture (ODA) and interchange format - Part 1: Introduction and general principles.*

ISO 8613-4:1989, *Information processing - Text and office systems - Office Document Architecture (ODA) and interchange format - Part 4: Document profile.*

ISO 8613-5:1989, *Information processing - Text and office systems - Office Document Architecture (ODA) and interchange format - Part 5: Office Document Interchange Format (ODIF).*

ISO 8649:1988, *Information processing systems - Open Systems Interconnection - Service definition for the Association Control Service Element.*

ISO 8650:1988, *Information processing systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element.*

ISO/IEC 8824:1990, *Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8825:1990, *Information technology - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO/IEC 9066-1:1989, *Information processing systems - Text communication - Reliable Transfer - Part 1: Model and service definition.*

ISO/IEC 9066-2:1989, *Information processing systems - Text communication - Reliable Transfer - Part 2: Protocol specification.*

ISO/IEC 9072-1:1989, *Information processing systems - Text communication - Remote Operations - Part 1: Model, notation and service definition.*

ISO/IEC 9072-2:1989, *Information processing systems - Text communication - Remote Operations - Part 2: Protocol specification.*

ISO/IEC 9594-1:1990, *Information technology - Open Systems Interconnection - The Directory - Part 1: Overview of concepts, models and services.*

ISO/IEC 9594-2:1990, *Information technology - Open Systems Interconnection - The Directory - Part 2: Models.*

ISO/IEC 9594-3:1990, *Information technology - Open Systems Interconnection - The Directory - Part 3: Abstract service definition.*

ISO/IEC 10021-3:1990, *Information technology - Text communication - Message Oriented Text Interchange Systems (MOTIS) - Part 3: Abstract service definition conventions.*

ISO/IEC 10031-1:1991, *Information technology - Text and office systems - Distributed-office-applications model - Part 1: General model.*

ISO/IEC 10031-2:1991, *Information technology - Text and office systems - Distributed-office-applications model - Part 2: Distinguished-object-reference and associated procedures.*

ISO/IEC 10166-2:1991, *Information technology - Text and office systems - Document Filing and Retrieval (DFR) - Part 2: Protocol specification.*

3 Definitions

3.1 General Terminology

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

Application Layer
application-entity
Presentation Layer
protocol
service definition

3.1.2 The following terms are used with the meanings defined in ISO/IEC 7498-2:

access control
authentication
authorization
credentials
security policy

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

macro

3.1.4 The following terms are used with the meanings defined in ISO 8649:

application context
Association Control Service Element

3.1.5 The following terms are used with the meanings defined in ISO/IEC 9066-1:

Reliable Transfer Service Element

3.1.6 The following terms are used with the meanings defined in ISO/IEC 9072-1:

Remote Operations: bind-operation, unbind-operation, operation
Remote Operation Service Element

3.1.7 The following terms are used with the meanings defined in ISO/IEC 10031:

accessee
accessor
consume-operation
distributed-office-application
data-object-value
distinguished-object-reference
document
produce-operation
privilege-attributes
referenced-object-access
ROA-operation
ROA-protocol

security-attributes
security-object
security-subject
user

3.2 Specific Terminology

For the purpose of ISO/IEC 10166 the following definitions apply:

3.2.1 **ancestor**: The *parent* of a *DFR-object* and, recursively, any ancestor of the former, including the *DFR-root-group*.

3.2.2 **attribute-type**: That component of an attribute which indicates the type of information given by the *attribute-value*.

3.2.3 **attribute-value**: A particular instance of that class of information indicated by an *attribute-type*.

3.2.4 **attribute-value-assertion**: A proposition, which may be true, false, or undefined, concerning the values of *DFR-attributes* in a *DFR-entry*.

3.2.5 **conceptual-document:** A set of *DFR-documents*, considered to be "different *versions* of the same document".

3.2.6 **Control-Attribute-Package:** A collection of attributes used to control access to a *DFR-object*.

3.2.7 **descendant:** For a given *DFR-group*, any of the *DFR-group-members*, and recursively, any descendant thereof.

3.2.8 **DFR-attribute:** A data item that identifies a *DFR-object*, describes its *DFR-content*, helps control access to it, or in some other way is associated with the *DFR-object*.

3.2.9 **DFR-basic-attribute-set:** The set of *DFR-attributes*, that will mandatorily be supported by every *DFR-server*.

3.2.10 **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*.

3.2.11 **DFR-document:** A structured amount of information that can be filed, retrieved, and interchanged consisting of a *DFR-document-content* and associated *DFR-attributes*.

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

3.2.13 **DFR-document-store:** A named collection of *DFR-objects* which is logically arranged in a hierarchical structure.

3.2.14 **DFR-entry:** A *DFR-object* together with additional *DFR-attributes* describing its hierarchical place in the *DFR-document-store*.

3.2.15 **DFR-extension-attribute-set:** The set of *DFR-Attributes* (beyond the *DFR-basic-attribute-set*) which are optionally supported by some *DFR-server*.

3.2.16 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*.

3.2.17 DFR-group-content: A sequence of UPIs identifying all *DFR-group-members* of the *DFR-group*.

3.2.18 DFR-group-member: A *DFR-object* which is identified in the *DFR-content* of its parent *DFR-group*.

3.2.19 DFR-membership-criteria: A *DFR-attribute* of a *DFR-group* establishing constraints on *DFR-group* membership based on attribute values.

3.2.20 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*.

3.2.21 DFR-object-class: A *DFR-attribute* indicating the class of a *DFR-object* (*DFR-document*, *DFR-group*, *DFR-reference* or *DFR-search-result-list*).

3.2.22 DFR-object-tree: The DFR-object-tree of a *DFR-group* is the tree formed by this *DFR-group* and all its *descendants*.

3.2.23 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.

3.2.24 DFR-proper-group: Any *DFR-group* other than the *DFR-root-group*.

3.2.25 DFR-reference: A *DFR-object* which acts as a link to another *DFR-object*, which is called the *referent* of the *DFR-reference*.

3.2.26 DFR-reference-content: The information stored in a *DFR-reference* for the purpose of identifying the *referent*.

3.2.27 DFR-root-group: The distinguished *DFR-group* within a *DFR-document-store* having no *Ancestor* and whose *DFR-object-tree* encompasses all *DFR-objects* in the *DFR-document-store*.

3.2.28 DFR-search-criteria: A *filter*.

3.2.29 DFR-search-result-list: A *DFR-object* which has information about a set of *DFR-objects* satisfying specified search criteria.

3.2.30 DFR-search-result-list-content: Information about the result of a DFR Search abstract operation.

3.2.31 DFR-server: That part of the DFR application which supplies Document filing and retrieval services.

3.2.32 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*.

3.2.33 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 privileges of that security subject.

3.2.34 filter: A construct specifying assertions about the presence or value of *DFR-attributes*, it is the same as in Directory (ISO/IEC 9594).

3.2.35 member: see *DFR-group-member*.

3.2.36 owner: A security subject, with owner access right to a specific *DFR-object*.

3.2.37 parent: Each *DFR-object*, except the *DFR-root-group*, is a *DFR-group-member* of a *DFR-group*, which is termed its *parent*.

3.2.38 Privilege-Attribute-Certificate: A certified set of access privileges that can be presented by a *DFR-user* to establish access rights.

3.2.39 referent: That *DFR-Object* to which a *DFR-Reference* refers.

3.2.40 version: *DFR-document* specified by the user as a derivation of one or more other *DFR-documents* by means of specific *DFR-attributes*.

4 Abbreviations

AE	application-entity
ASN.1	Abstract Syntax Notation One
CAP	Control-Attribute-Package
DFR	Document filing and retrieval
DOAM	Distributed-office-application model
DOR	Distinguished-object-reference
DS	DFR-Document-Store
PAC	Privilege-Attribute-Certificate
QoS	Quality of Service
ROA	Referenced-object-access
ROSE	Remote Operations Service Element
UPI	DFR-Unique-Permanent-Identifier

5 Conventions

This part of ISO/IEC 10166 uses the description conventions listed in the following clauses.

5.1 Conventions for Abstract-services

This part of ISO/IEC 10166 uses the following ASN.1-based descriptive conventions for the indicated purposes:

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

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

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

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

NOTE - ASN.1 specifications in this International Standard make full use of ISO 8824:1990 features, especially such syntactical constructs as "WITH COMPONENTS" (subtyping 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 a ROSE-based International 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 of ISO/IEC 10166 the following rules apply:

a) 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 refer to clause 3 (if it is an attribute see also clause 9); exceptions are titles of other International Standards which also begin with capital letters, see clause 2.

b) 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 annexes or to the corresponding sections in the main text.

The following characters are used in this part of ISO/IEC 10166 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 (shall be supported by DFR);

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

- C (Conditional) stands for the condition that an item shall be present under some circumstances defined in this part of ISO/IEC 10166.

Section 2 - 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.

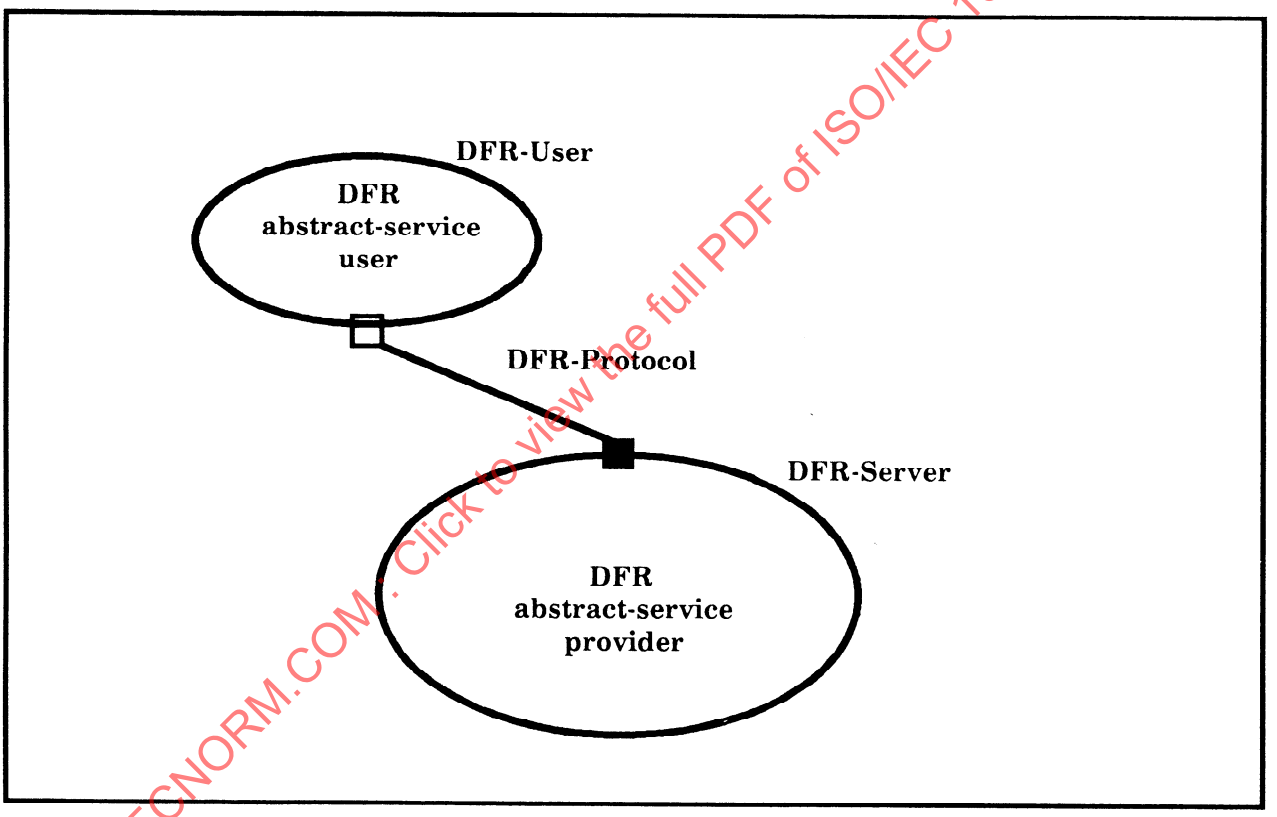

Figure 1 - Document Filing and Retrieval Abstract-service

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 modelled 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 port is the concern of this part of ISO/IEC 10166 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 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 in order to use 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. This gives the following specification for all DFR-Object classes.

```
DfrObjectClass :: = ENUMERATED {

                    dfr-document            (0),
                    dfr-root-group          (1),
                    dfr-proper-group        (2),
                    dfr-reference           (3),
                    dfr-search-result-list  (4)  }
```

A DFR-Document-Store is accessed through a DFR-Server. A DFR-Document-Store is assigned to one DFR-Server only. A DFR-Server manages one DFR-Document-Store. 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. A DFR-

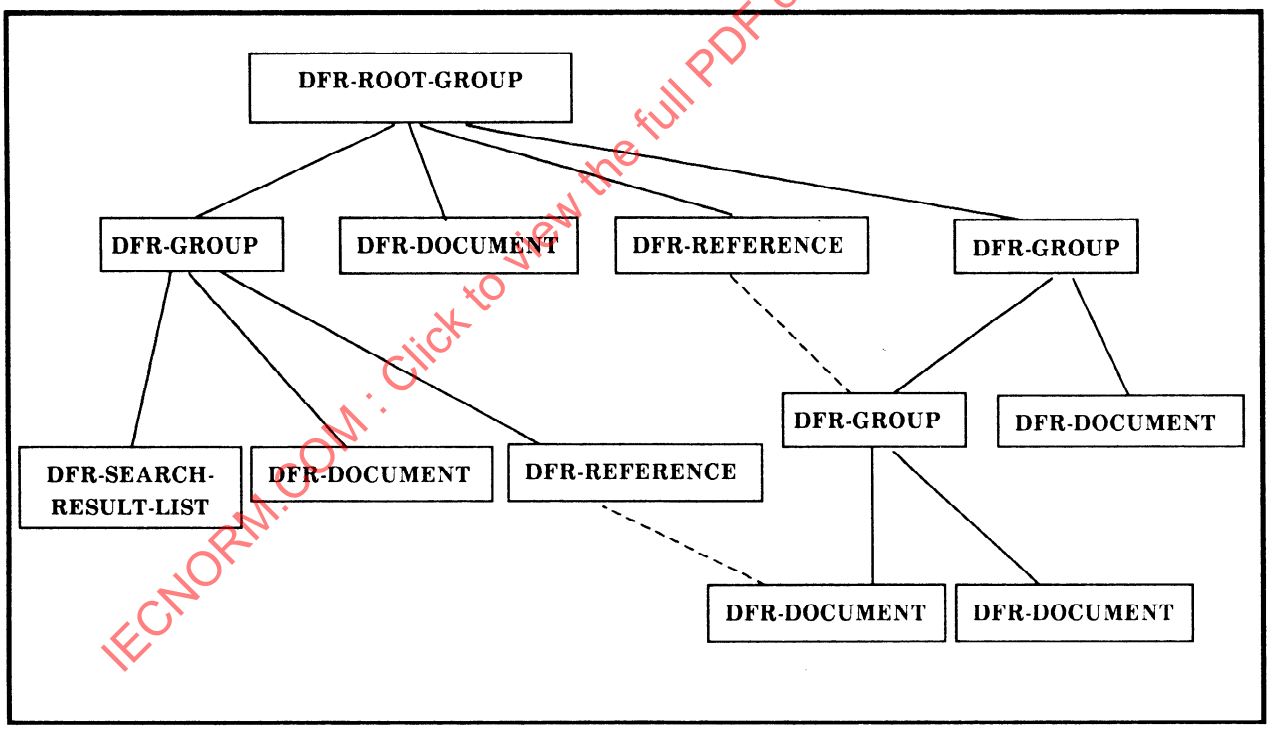

Figure 2 - Example DFR Document Store structure

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 can 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 can be referenced from more than one DFR-Reference. A DFR-Group can be viewed as the root of a DFR-Object-Tree consisting of 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 corresponding DFR-Object plus two supplementary attributes, DFR-Parent-Identification and DFR-Pathname.

The attributes forming the **DfrEntryAttributes** are defined either in this part of ISO/IEC 10166, 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 International Standard, the mandatory DFR-Basic-Attribute-Set and one optional DFR-Extension-Attribute-Set derived from the ODA-Document-Profile. Other optional extension attribute sets, for example one extension attribute set which includes 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 Annex E.

NOTE 1 Negotiation of extension attribute sets is performed by means of negotiation of the corresponding abstract syntax during the bind-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 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,
    reference-content         [3] DfrReferenceContent,
    search-result-list-content [4] DfrSearchResultListContent }
```

Each DFR-Object has a unique identification within the DFR-Document-Store. This identification is given by a DFR-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 2 UPI is also modelled as a DFR-Attribute, see 9.2.1.

6.3.1 DFR-Document-Store

A DFR-Document-Store (DS) 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 identify unambiguously 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. Maintenance of consistency between DFR-Documents and their copies in different DFR-Document-Stores is outside the scope of this part of ISO/IEC 10166, 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-Document-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 - The co-operation of DFR with a document content access protocol may be the subject of future standardization.

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 can be a DFR-Document, a DFR-Group or a DFR-Search-Result-List; it shall not be another DFR-Reference. The DFR-Object-Class of the Referent is noted in the DFR-Reference-Content.

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

A DFR-User with read access permission (see 6.3.8.3) to the DFR-Reference can also access the Referent by using the DOR stored in the DFR-Reference-Content. In the case of a local access to the Referent (i.e. DFR-Reference und Referent are in the same DS) only those DFR-Users who are noted in the DFR-Access-List attribute of the Referent can access the Referent. In the case of a remote access to the Referent (i.e. DFR-Reference and Referent are in different DS) the access permission can only be verified by the optional token in the DOR (see ISO/IEC 10031-1).

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-Referent-Deleted attribute of the DFR-Reference.

The content of a DFR-Reference is structured like a Distinguished Object Reference (see ISO/IEC 10031-2) to simplify its use in an ROA-operation (see ISO/IEC 10031-1). The Distinguished Object Reference (DOR) is used by DFR in two different ways:

a) Use of a DOR as the content of a DFR-Reference: A DOR to an entire DFR-Object can be produced at the DFR-User's request by the DFR-Server. This DOR can then be stored in the content of a DFR-Reference at the DFR-User's request. This DFR-Reference can later be used in an ROA-operation. The DFR-User can also later read the content of this DFR-Reference and transfer it to another application for subsequent access to the referenced DFR-Object by that application.

The DOR can be stored in a DFR-Reference of the same DFR-Document-Store that contains also the referenced DFR-Object or in another DFR-Document-Store. In the first case, the content of the DFR-Reference is a DOR whose **ae-identifier** and application sub-components point to the DFR-Server which manages both Referent and reference.

b) Use of a DOR for immediate transfer: A DOR to an entire DFR-Object, to the attributes of a DFR-Object, or to the content of a DFR-Object can 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-object-value to some other DFR-Server or to another application. If some DFR data-object-value is to be transferred by an ROA-operation to another application, it is necessary that the latter be able to interpret this DFR data-object-value (for example the print application might only accept a DFR-Document-Content).

6.3.3.1 DFR-Reference Content

The DOR stored in the content of a DFR-Reference contains among others a pointer to the Referent (local reference) and an identifier (data-object-type) of the Referent (to indicate whether the Referent is a DFR-Document, a DFR-Group or a DFR-Search-Result-List).

The ASN.1 specification of a DFR-Reference-Content is as follows:

```
DfrReferenceContent :: = DOR
            (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,
                token ABSENT      } )

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

The semantics of the different components of the DFR-Reference-Content (the DOR) are as defined in ISO/IEC 10031-2.

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 can 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.

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 Support of the ROA-model by DFR

This clause describes the support of the referenced-object-access functional model (ROA-model) by DFR. The ROA-model is described in ISO/IEC 10031-1. DFR uses the DOR as defined in ISO/IEC 10031-2.

6.3.3.3.1 Role of DFR in the ROA-model

A DFR-Server shall perform both the accessee role and the accessor role in the ROA-model:

1) The DFR-Server as an accessee performs the following role as defined in the ROA-model.

   a) The DFR-Server accepts produce-operations.

   b) The DFR-Server accepts certain ROA-operations.

2) The DFR-Server as an accessor performs the following role as defined in the ROA-model.

   a) The DFR-server accepts consume-operations.

   b) The DFR-Server can issue certain ROA-operations.

6.3.3.3.2 Produce-operations in DFR

A DFR abstract operation is termed a produce-operation as defined in ISO/IEC 10031-1 if a DFR-User requests the transfer of a reference to the data-object-value in that operation, that is the DFR-Server returns a DOR instead of the data-object-value. A DFR-User can request a DOR in the result of the following DFR abstract operations:

- **Create**
- **Copy**
- **Move**
- **Read**
- **Modify**

At the time of a produce-operation, DFR-Users can request the required Quality-of-Service (QoS) by the Requested-QoS-level parameter. The DFR-Server determines its available QoS and sets the appropriate values in the resulting DOR.

NOTE - It is outside the scope of this International Standard to define whether the capability to change a QoS associated with a DOR is supported or not.

6.3.3.3.3 Consume-operations in DFR

A DFR abstract operation is termed a consume-operation as defined in ISO/IEC 10031-1 if a DFR-User either provides a DOR instead of a data-object-value to the DFR-Server or provides an identifier (DfrEntryName) for a DOR already stored (in a DFR-Reference) in the DFR-Server.

A DOR can be included in the following DFR abstract operations:

- **Create**

- **Copy**
- **Move**
- **Read**
- **Modify**

After receiving the consume-operation, the DFR-Server performs the accessor role of an ROA-operation by issuing an appropriate ROA-protocol abstract operation acceptable to the accessee. How the DFR-Server knows the appropriate ROA-protocol and ROA-operation to use when accessing the accessee is outside the scope of this part of ISO/IEC 10166. After receiving the result of the ROA-operation, the DFR-Server performs the requested original consume-operation as if the data-object-value was included in the original abstract operation.

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 can 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 - 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 6.3.8) is performed by some administration authority. This administration authority is not defined in this part of ISO/IEC but left to local implementation.

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 can 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 9.2.9) of the DFR-Group.

```
DfrGroupContent :: = SEQUENCE OF DfrUniquePermanentIdentifier
```

A DFR-Group can be assigned a DFR-Membership-Criteria attribute. The membership criteria govern membership in the group; new Members are required to satisfy the membership criteria before being added. The attributes of the new Member are verified against the membership criteria and the addition is denied if the verification fails. A group having no specified membership criteria permits any new direct 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 shall 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 can 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 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 abstract 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 can 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 can only be modified by another **Search** abstract operation. A **Search** abstract operation can 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 {
    start-date-and-time     [0] GeneralizedTime,
    end-date-and-time       [1] GeneralizedTime,
    object-list             [2] DfrEntryList,
    ordering                [3] OrderingRule OPTIONAL,
    search-domain           [4] SearchDomain,
    search-criteria         [5] SearchCriteria } }
```

A **DfrSearchResultListContent** can only be "produced" by a **Search** abstract operation; at the DFR-Search-Result-List 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** can 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 DFR-Search-Result-List, describe its content, help control access to it, or are in some other way associated with the DFR-Search-Result-List.

DFR-Search-Result-List attributes serve to qualify or enhance a DFR-Search-Result-List-Content, or to define additional characteristics of the DFR-Search-Result-List 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.

6.3.6 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 the user 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 can 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 DFR-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 can 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 can 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).

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 can be reached by navigating through it using the DFR-Next-Versions attribute.

In a more general case, a given version of a Conceptual-Document can have several next versions (tree model). In addition, a version can 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 can be linked only to existing ones (a DFR-Previous-Versions attribute shall 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. The DFR-User shall have at least read access right to the DFR-Document(s) specified as previous version(s). 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 to point to each of its predecessors, and all links to it from its predecessors are redirected to point to each of its successors. An existing version can be "discarded" either by deletion of the whole DFR-Document or by deletion of the attribute DFR-Previous-Versions of the DFR-Document in which case the DFR-Server also deletes the attributes DFR-Next-Versions and DFR-Version-Root.

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 and Filter

DFR uses **Attribute**, **ATTRIBUTE MACRO**, **ATTRIBUTE-SYNTAX MACRO**, **Filter** and **FilterItem** as defined 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 the convenience of the reader a shortened description of Attributes and Filter is in annex F. The complete definition is in ISO/IEC 9594.

For an overview of the DFR-Attributes see clause 9.

NOTE - 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 collating sequences. The definition of collating sequences is outside the scope of this part of ISO/IEC 10166.

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. The DFR-User is not allowed multiple binds, at the same time, to the same DFR-Server.

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 part of ISO/IEC 10166 defines one specific control attribute, the DFR-Access-List.

If the accessing DFR-User has presented a Privilege-Attribute-Certificate (PAC) containing privilege attributes for a DFR-Object, to determine what access the user has, the privilege attributes 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 8.1.3.5), or both. The DFR-User can 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 8.1.3.5).

Each specific DFR realisation shall use either the DFR-Access-List attribute, or 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 security subject 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** abstract operation, it will not be made visible to a DFR-User that does not have 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 its 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 {
        access-id       AccessId,
        access-rights   AccessRights  }

AccessId ::= DistinguishedName
```

NOTE - 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-Attributes-Modified-By, DFR-Content-Modified-By, DFR-Reserved-By. Attempts to read the DFR-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** abstract 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 apply an uncommitted reserve or unreserve to that DFR-Object and modify its content and/or attributes with the exception of the attribute DFR-Access-List. Modification of the content of a DFR-Group means new Members can 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 can move the whole DFR-Object-Tree of this DFR-Group regardless of the access rights in the DFR-Access-Lists of the Descendants, but the DFR-User can delete the whole DFR-Object-Tree of this DFR-Group or any Descendant of the DFR-Group only if this DFR-User is noted with read-modify-delete access permission in the DFR-Access-List of all Descendants if the whole DFR-Object-Tree of this DFR-Group

shall be deleted or in the DFR-Access-List of a Descendant if this particular Descendant shall be deleted. 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 and a committed reservation can be applied on 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 can 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.

NOTE - The constraints concerning access of a Referent by using a DOR are noted in 6.3.3.

7 Abstract-bind and Abstract-unbind Parameters

7.1 Abstract-bind Parameters

This clause defines and describes the bind-operation parameters for the DFR port.

```
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 (see also 6.3.8). It also accepts a set of restrictions for entries to be returned as result of a DFR abstract operation.

The definition is:

```
DfrBindArgument :: = SEQUENCE {
    initiator-name              [0] DistinguishedName,
    credentials                 [1] Credentials,
    retrieve-restrictions       [2] Restrictions OPTIONAL, -- default is none--
    dfr-configuration-request   [3] BOOLEAN DEFAULT FALSE,
    bind-security               [4] BindSecurity OPTIONAL,
    priority                    [5] Priority DEFAULT medium,
    dor-for-produce-operations  [6] BOOLEAN DEFAULT TRUE,
    dor-for-consume-operations  [7] BOOLEAN DEFAULT TRUE }
```

a) **initiator-name** (M): This argument contains the distinguished name of the initiator of the association and is supplied by the DFR-User.

b) **credentials** (M): Credentials can 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 outside the scope of this part of ISO/IEC 10166 (these matters are common to all bind-operations). Use of certified security attributes is described in the outline below. Semantic details are not defined however since this is dependent on the actual security policy formulated and implemented by the organization operating the DFR application.

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 bind-operation 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 can abort the bind-operation if the authentication parameters do not justify successful completion of the bind-operation.

The credentials passed in a bind-operation can be modified by the **Privileges** parameter of a specific DFR abstract operation (see 8.1.3.5).

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

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

A **PrivilegeAttributeCertificate** contains attributes associated with the DFR-User such as 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 - 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.

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

In the absence of this argument, the default is that no restrictions exist.

This argument consists of the following components:

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

1) **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 will 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 exist.

2) **maximum-length** (C): The maximum-length that the DFR-User is prepared to provide as the argument or to accept as the result of an abstract operation. Any result prepared by the DFR-Server exceeding the maximum length specified will not be returned, but will result in an error.

In the absence of this component, the default is that no restrictions on result-length exist.

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

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

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

```
BindSecurity :: = EXTERNAL
```

f) **priority** (C): Priority requested for this DFR-User during the association. If there is no other priority specified in the abstract operations (see 8.1.3.4), this priority shall be applied. By default medium priority is applied.

g) **dor-for-produce-operations** (O): This parameter, if left TRUE, indicates that the DFR-Server shall, if requested, provide a DOR in the result of an abstract operation. Whether the DFR-Server commits to this request is reported in the DfrBindResult parameters.

h) **dor-for-consume-operations** (O): This parameter, if left TRUE, indicates that the DFR-Server shall, if requested, accept a DOR in the arguments of an abstract operation. Whether the DFR-Server commits to this request is reported in the DfrBindResult parameters.

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,
    constraints-supported              [1] SET OF ConstraintsType OPTIONAL,
    dfr-document-types-supported       [2] SET OF TypeAndAttribute OPTIONAL,
    function-set-supported             [3] FunctionSetType OPTIONAL,
    maximum-length-supported           [4] INTEGER OPTIONAL,
    dor-for-produce-operations         [5] BOOLEAN DEFAULT TRUE,
    dor-for-consume-operations         [6] BOOLEAN DEFAULT TRUE,
    rOA-protocols-accessee             [7] ROAProtocols OPTIONAL,
    rOA-protocols-accessor             [8] ROAProtocols OPTIONAL }
```

a) **authentication-attributes** (M): Any information returned as confirmation of an authentication check. This information is unconstrained by this part of ISO/IEC 10166.

```
AuthenticationAttribute :: = EXTERNAL
```

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

```
ConstraintsType :: = SEQUENCE {
    name-constraint    [0] NameConstraint,
```

```
                    version-constraint  [1] VersionConstraint }

NameConstraint :: = ENUMERATED {
    no-name-constraint      (0),
    local-unambiguity       (1),
    global-unambiguity      (2) }


VersionConstraint :: = ENUMERATED {
    no-version-constraint       (0),
    version-unambiguity         (1) }
```

For a DFR-Document-Store the following constraints can be specified:

1) Name-constraint specifies the uniqueness of the DFR-Title :

   i) No constraints for DFR-Titles (no uniqueness of DFR-Titles enforced);

   ii) Global unambiguity of DFR-Titles (each DFR-Entry has a DFR-Title unique within the DS);

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

2) Version-constraint specifies the uniqueness of the Version-Name attribute:
   i) No constraints for Version-Names (no uniqueness of Version-Names enforced);
   ii) Version unambiguity (each Version of a Conceptual-Document is uniquely identified by its Version-Name in the scope of its Conceptual-Document).

c) **dfr-document-types-supported** (C): This is a list of document types supported by this DFR-Server, with a list of extension attributes supported for each document type. Only present if a dfr-configuration-request was made.

```
TypeAndAttribute :: = SEQUENCE {
    document-types      [0] OBJECT IDENTIFIER,
    attributes          [1] SET OF AttributeType }
```

d) **function-set-supported** (C): Specifies which function-set of the three defined sets (Flat Store Set, Pre-defined Store Structure Set, Full Set - see 8.4) is supported. Only present if a dfr-configuration-request is made.

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

e) **maximum-length-supported** (C): Specifies the maximum-length the server supports. This value shall be less than or equal to the maximum length specified by the DFR-User in the retrieve-restrictions of the bind-operation. This parameter is only present if a maximum length was specified in the bind arguments.

f) **dor-for-produce-operations** (O): This parameter, if left TRUE, indicates that the DFR-Server can provide a DOR in the result of an abstract operation.

g) **dor-for-consume-operations** (O): This parameter, if left TRUE, indicates that the DFR-Server accepts a DOR in the arguments of an abstract operation.

h) **rOA-protocols-accessee** (O): The DFR optionally returns information about which ROA-protocol application context(s) it supports, as accessee, for the transfer of objects corresponding to DORs.

```
ROAProtocols :: = SEQUENCE OF OBJECT IDENTIFIER
```

i) **rOA-protocols-accessor** (O): The DFR optionally returns information about which ROA-protocol application context(s) it supports, as accessor, for the transfer of objects corresponding to DORs.

7.1.3 Bind-error Parameters

The DFR-port can report either of two errors, **SecurityProblem** or **ServiceProblem**. **SecurityProblem** indicates that the credentials parameter was not adequate to gain access to the DFR-Server. **ServiceProblem** reports that the DFR-Server can not establish the association due to some operational exception. The same errors can occur in DFR abstract operations, see 8.3.

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

7.2 Abstract-unbind Parameters

An unbind-operation closes the DFR port. The issuing of an unbind-operation results in the deletion of any retrieve-restrictions that were specified in the bind-operation argument. There are no arguments or errors associated with the unbind-operation.

```
DfrUnbind :: = ABSTRACT-UNBIND
    FROM {dfr-ports [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 clause 6 of this part of ISO/IEC 10166. 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),
DfrReferenceContent (see 6.3.3.1),
DfrObjectClassID (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 clause 6 as well as in this clause are actually defined in other International 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.

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

DOR,
Requested-QoS-level

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 {
    task-id         [26] TaskId OPTIONAL,
    reservation     [27] Reservation OPTIONAL,
    error-handling  [28] ErrorHandlingMode DEFAULT all-or-nothing,
    priority        [29] Priority DEFAULT medium,
    privileges      [30] Privileges OPTIONAL }
```

8.1.3.1 Task-id

```
TaskId ::= OCTET STRING
```

TaskId is an optional identifier for an abstract operation. A DFR-User can include this parameter in an abstract operation for identification purposes, if it is to be abandoned or continued. The **TaskId** can either be specified by the DFR-User or can be that previously provided by the DFR-Server as continuation-parameter in the result of a **List** or **Search** abstract operation if such an abstract operation was stopped due to a limit-encountered situation (see 8.1.6).

A DFR-User shall not re-use the **TaskId** value in another abstract operation until the respective abstract operation is (perhaps after a continuation) completed, or abandoned.

8.1.3.2 Reservation

The semantics of this parameter when it is specified in an abstract operation are 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-Object concerned. For each DFR abstract operation, the DFR-Object to which a reservation applies is specified in the description of that abstract operation. The reservation is defined by reservation-duration, reservation-level and reservation-status.

The **reservation-duration** optionally specifies the time for which the reservation shall exist. If no reservation-duration is specified the DFR-Object will remain reserved indefinitely.

The **reservation-level** specifies which level of reservation shall be applied. Level unreserved is the lowest level, level read-only2 is the highest level. If a change to a lower level from an existing higher level in the case of an uncommitted reservation is requested it is always performed as the last step, in the execution of the abstract operation. If a change to a higher level from a lower level or from uncommitted to committed is requested, it is always performed as the first step in the execution of the abstract operation.

The **reservation-status** defines the quality of the reservation. The reservation-status "committed" is only applicable for the Owner of a DFR-Object. Committed means, the reservation will be maintained until the reservation duration expires. After a committed reservation is specified by the Owner it can not be withdrawn, i.e. the reservation level can only be increased and/or the reservation duration can only be set to a later date or be omitted, in which case the reservation will exist indefinitely. It is also not permitted to set the reservation status from committed to uncommitted.

After a DFR-Object is reserved by some user reservation requests of other users will be rejected. The reserved DFR-Object is only re-available for a reservation by another user if it is set to unreserved by the user who has reserved it in the case of an uncommitted reservation or the reservation duration has expired. The DFR-Reserved-By attribute is used to indicate the identity of the DFR-User who reserved the DFR-Object.

```
Reservation ::= SEQUENCE {
    reservation-duration        [0] GeneralizedTime OPTIONAL,
    reservation-level           [1] ReservationLevel,
    reservation-status          [2] ReservationStatus DEFAULT uncommitted }

ReservationLevel ::= ENUMERATED {
        unreserved              (0),
        exclusive-write         (1),
        exclusive-access        (2),
        read-only1              (3),
        read-only2              (4) }

ReservationStatus ::= ENUMERATED {
    uncommitted (0),
    committed   (1) }
```

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

a) **unreserved**
The DFR-Users have unrestricted access to the DFR-Object by using the abstract operations defined in this part of ISO/IEC 10166.

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

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

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

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

iii) other users shall not

- **Delete** or **Move** any Member from the DFR-Group,
- **Copy** or **Move** any DFR-Object to the DFR-Group,
- insert new Members into the DFR-Group by **Create** abstract operations;

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

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

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

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

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

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

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

iii) other users 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.

d) **read-only1**
The restrictions on this reservation level are the same as described for the reservation level exclusive-write. But in this case the restrictions are valid for other users and for the user who has reserved the DFR-Object.

e) **read-only2**
The restrictions on this reservation level are for other users the same as described for the reservation level exclusive-access. But the access is also restricted for the user who has reserved the DFR-Object, i.e. the access to the read-only2 reserved DFR-Object is restricted for the user who has reserved it as described for other users for the reservation level exclusive-write.

8.1.3.3 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, --
                                               -- these are List or Copy group. --

Warning ::= SEQUENCE {    -- warning will never be reported if the DFR-User has no read access right --
    entry     [0] DfrEntryName,
    access1   [1] AccessProblem OPTIONAL,            -- see 8.3.3 --
    access2   [2] ReferentAccessProblem OPTIONAL }   -- see 8.3.5 --
```

When several DFR-Entries need to be accessed during the execution of the DFR abstract operations **List** and **Copy**, different situations can occur:

a) none of the DFR-Entries involved can in fact be accessed;

b) all the involved entries can be accessed;

c) some entries can be accessed, while others can not.

The first situation (a) is that of a DFR error (of access type), and the second (b) that of success (if no other error occurs). The third situation (c) 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.

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.

The error handling mode **until-first-warning** means that the abstract operation will be performed until an inaccessible Member is detected, in which case the abstract operation is stopped. Only the so far accessed entries are copied or listed and that entry causing the termination of the abstract operation is reported as a warning. The result of the abstract operation is in this case DFR-Server implementation dependent.

The error handling mode **report-all-warnings** means that the abstract operation will be performed to its conclusion regardless of the detection of inaccessible Members. Inaccessible Members, if detected, are reported as warnings in the result of the abstract operation. All accessible Members are copied or listed.

The error handling mode **report-n-warnings** means that the abstract operation will be performed to its conclusion regardless of the detection of inaccessible Members. Inaccessible Members, if detected, are reported as warnings in the result of the abstract operation up to the specified number. All accessible Members are copied or listed independent of the limitation specified for warnings. If the limitation specified for warnings is zero, the abstract operation will be performed to its conclusion without any warnings being reported.

The DFR service definition leaves it to the DFR-User to specify the error-handling mode desired on a per abstract operation basis.

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 can appear in the abstract operation result provided that the abstract operation argument had **ErrorHandlingMode** specified as until-first-warning, report-all-warnings or report-n-warnings.

Any problems related to the modifications parameter (see 8.1.5.5) are always handled as if the all-or-nothing mode was specified.

8.1.3.4 Priority

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

This common parameter shall 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** can already have been established at bind time (see 7.1.1); any **Priority** requested here will override 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 layers.

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

8.1.3.5 Privileges

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

The **Privileges** parameter enables a **PrivilegeAttributeCertificate** to be associated directly with an abstract operation request, modifying that available from the bind-operation 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 this abstract operation.

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

The **operation-pac** can 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 shall present his own access privileges.

The **proxy-pac** can be required if the DFR-Server needs to 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, GeneralString }
```

A DFR-Entry, and thus the DFR-Object it represents in the DS can always be accessed using the DFR-Unique-Permanent-Identifier 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 sometimes do not lead to the same DFR-Entry.

NOTE - For the CharacterData strings of the DFR-Titles which form a DfrPathName the caseIgnoreStringSyntax also applies as defined in 9.4.1.1.

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 at least 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 can be useful to identify some Descendants of this DFR-Group, the latter specifying some "working domain".

8.1.5 Data types common for abstract operations on single entries

```
CommonUpdateArguments ::= SEQUENCE {
    object-class        [0] DfrObjectClass OPTIONAL,
    entry               [1] CHOICE {  local     [1]  DfrEntryName,
                                      external  [2]  DOR } 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,
    reference-qos       [6] Requested-QoS-level OPTIONAL }

CommonUpdateResult ::= SEQUENCE {
    upi                 [0] DfrUniquePermanentIdentifier,
    entry-information   [1] EntryInformation OPTIONAL,
    warnings            [2] SEQUENCE OF Warning DEFAULT {},
    continuation        [3] TaskId OPTIONAL,
    reference-qos       [4] Quality-of-Service OPTIONAL, -- imported by DOR --
    referent-altered    [5] ReferentStatus OPTIONAL }

ReferentStatus ::= ENUMERATED {
    not-changed-since-produce  (0),
    changed-since-produce      (1) }
```

DFR abstract operations concerned are: **Create**, **Copy**, **Move**, **Read**, and **Modify**. Not all of the above parameters of **CommonUpdateArguments** can be specified in each of these abstract operations. Details are given below in the description of the parameters of **CommonUpdateArguments**, as well as in 8.2 where the individual abstract operation types are defined.

8.1.5.1 Object-Class

This common parameter specifies the **DfrObjectClass** of the DFR-Object 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 (or a DOR to) the DFR-Object 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 can 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 can 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 }, -- as default 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 }, -- as default 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). For the **Create** abstract operation, two methods are available for specifying different items of the newly created DFR-Entry: either by the **EntryModifications** parameter, or by directly specifying the items in the abstract operation.

An **EntryModification** can apply to an attribute of a specified type, or to the entire DFR Content of the DFR-Entry to be updated. An entire attribute can 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 can 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 can 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 can 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 shall not be modified - this can 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) shall be the same.

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,
    dor                 [4] DOR }
```

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, can be the parent DFR-Group of this DFR-Object, or its Referent (for DFR-References only; this can involve the transfer of attributes from a remotely stored Referent to the DFR-Reference using the DOR stored in the DFR-Reference-Content) or its previous version (only in the case where the DFR-Object has a unique previous version), or some explicitly specified source DFR-Entry, or can be specified by a DOR (this can be used for the transfer of attributes or content or both as specified by the DOR).

8.1.5.6 Selection

```
EntryInformationSelection ::= SEQUENCE {
    read-selector                   [0] ENUMERATED {
        attributes-only                 (0),
        attributes-and-content          (1),
        content-only                    (2),
        dor-to-attr-only                (3),
        attr-and-dor-to-content         (4),
        dor-to-content-only             (5),
        dor-to-entire-object            (6),
        attr-and-dor-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 not significant--
    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 --

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

Any of the five abstract operations considered (namely **Create**, **Move**, **Copy**, **Read** and **Modify**) can 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. The **Read** abstract operation is provided for this purpose. However, the facility can also be used by 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 a DOR or some values are required, and whether the DOR or the 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. The **attribute-selection** component optionally indicates, where appropriate, which attributes of the DFR-Entry are to be returned.

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

8.1.5.7 Reference-qos

This parameter optionally specifies the requested Quality of Service (Qos) for a DOR if a DOR is to be produced.

8.1.6 Data types common to abstract operations on multiple entries

These data types are used by **List** and **Search**. They are related to the data-type **DfrSearchResultListContent** defined in 6.3.5.1.

```
CommonListSearchArguments ::= SEQUENCE {
    continue            [0] BOOLEAN DEFAULT FALSE,
    limits              [1] Limits OPTIONAL,
    selection           [2] AttributeSelection OPTIONAL,
    ordering            [3] OrderingRule OPTIONAL }

CommonListSearchResult ::= SEQUENCE {
    number-of-entries   [0] INTEGER,
    limit-encountered   [1] LimitEncountered    OPTIONAL,
    entry-list          [2] DfrEntryList,
    warnings            [3] SEQUENCE OF Warning DEFAULT {},
    continuation        [4] TaskId OPTIONAL }
```

8.1.6.1 Continue and limits

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

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

When initiating a **List** or a **Search** abstract operation, the DFR-User can specify a maximum number of entries to be returned, a maximum execution time (for the **Search** abstract operation only), or both. The result returned in this case contains an indication if the time-limit, count-limit or maximum length as specified during binding has been reached. In the case where 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, the DFR-Server communicates to the DFR-User a continuation parameter (the **TaskId** - see 8.1.3.1, 8.1.5 and 8.1.6). This continuation parameter can be provided to the DFR-Server in the **CommonArguments** when resuming "the same" abstract operation (immediately or later), as indicated by the continue parameter, to instruct the DFR-Server not to start the abstract operation "from the beginning", but to continue it from where it has been previously stopped.

8.1.6.2 Selection

```
DfrEntryList  ::= SEQUENCE OF SEQUENCE {
    upi                 [0] DfrUniquePermanentIdentifier,
    class               [1] DfrObjectClass,
    ordering-attribute  [2] SEQUENCE OF Attribute OPTIONAL, -- ordered as specified in the OrderingRule --
    other-attributes    [3] SEQUENCE OF Attribute OPTIONAL }
```

This is the information about the DFR-Entries listed or found. Only attributes of entries can 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. The ordering-attributes are optionally returned (see 8.1.6.3). 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

```
OrderingRule :: = SET {
    list-attributes          [0] BOOLEAN DEFAULT FALSE,
    ordering-attributes      [1] OrderingAttributes }

OrderingAttributes :: = SEQUENCE OF SEQUENCE {
    attribute      [0] AttributeType,
    direction      [1] Direction }

Direction :: = ENUMERATED {
    ascending      (0),
    descending     (1) }
```

The user can 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 attributes to be used as ordering keys, and the ordering direction (ascending or descending). If such an ordering rule has been specified, each entry concerned is optionally listed together with the value of its ordering-attributes; if, however, no explicit ordering has been specified, then the ordering-attributes are absent from the abstract operation result. The **OrderingRule** is retained with the **DfrSearchResultList** if the abstract operation requests the results are to be placed in one. An **OrderingRule** can only be applied for attributes type having a **MATCHES FOR ORDERING** clause in their definition.

NOTE - 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 International Standard does not specify how and when a collating sequence is established.

8.1.6.4 Search-domain

```
SearchDomain :: = SEQUENCE OF CHOICE {
        previous-result    [0] DfrEntryName,
         --specifies an entry of "DFR-Search-Result-List" 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 then can be 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 can be limited (by the descent-depth parameter), as well as the depth of dereferencing. Depth of dereferencing means the number of times the server can indirectly proceed from a DFR-Reference to its Referent when executing this search; for example, starting from a DFR-Reference the DFR-Server might have to proceed 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 Search-criteria

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 a DFR-Entry an abstract operation is related to.

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

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	
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								

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.

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

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

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: Not 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-Entry to be created.

Source Parent DFR-Entry: The Parent of the source DFR-Entry.

Source 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 a DFR-Entry for which the requestor has read-modify-delete access right to an Ancestor, regardless of the requestor´s access rights to the DFR-Entry to be moved.

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 DFR-Membership-Criteria attribute for future Members can be defined. Consequential changes to the Parent DFR-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 DFR-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 DFR-Documents specified as a previous version.

Whenever the security policy defined in 6.3.8 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,
        AccessError,
        AttributeError,
        InterServerAccessError,
        NameError,
        ReferentAccessError,
        SecurityError,
        ServiceError,
        UpdateError,
        VersionManagementError}
```

8.2.1.1 Create-argument

```
CreateArgument :: = SEQUENCE {
    COMPONENTS OF
        CommonUpdateArguments (WITH COMPONENTS { ... ,
            object-class PRESENT, entry ABSENT,
            destination PRESENT, reference-qos ABSENT } ),
    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;
- 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 DFR-Attributes and/or DFR-Content of the new DFR-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;
- reference-qos is not applicable.

b) Specific **Create** arguments:

- attributes is the primary means to provide DFR-Attributes for the DFR-Entry to be created (alternative means is modifications argument above; both can be used in the same **Create** abstract operation, in which case attributes applies first);
- content is the primary means to provide a DFR-Content for the DFR-Entry to be created (alternative means is modifications argument; they can not be used simultaneously in the same **Create** abstract operation).

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

c) Common arguments:

- task-id, see 8.1.3.1;
- reservation, if requested, applies to the newly created DFR-Entry;
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

8.2.1.2 Create-result

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

```
CreateResult :: = CommonUpdateResult (WITH COMPONENTS { ... , warnings ABSENT, reference-qos ABSENT } )
```

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 (DFR-Attributes and/or DFR-Content) which have been requested by the selection component of the **CreateArgument**, and which are present in the DFR-Entry;
- referent-altered is returned if the different items of the newly created DFR-Entry are specified by the **EntryModification** parameter of **CommonUpdateArguments**, it indicates if (one of) the Referents from which items are copied to the created DFR-Entry are changed or not (after the DOR produce time).

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 DFR-Group attributes are made. The UPI of the specified DFR-Object ceases to be valid. If a DFR-Group is deleted the DFR-Group and all its Descendants are deleted provided the DFR-User has sufficient access rights to all of its 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 DFR-Attributes in all its previous and next Versions. The delete request fails if the DFR-Object to be deleted, or any of its descendants in the case of a DFR-Group, is reserved by another DFR-User.

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

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** arguments:

- entry identifies the DFR-Entry to be deleted by **DfrEntryName**.

b) Common arguments:

- task-id, see 8.1.3.1;
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

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 the Descendants are copied according to the **ErrorHandlingMode** specified. A DFR-Entry is copied if the DFR-User has at least read access permission to it and if the DFR-Entry is currently not exclusive-access reserved by another DFR-User (see 8.1.3.2) and the DFR-User has at least read access permission to all Ancestors which exist between this DFR-Entry and the DFR-Entry specified in the entry parameter to the **Copy** arguments. Modifications are applied to the newly created DFR-Entry if requested by the DFR-User (see 8.2.3.1). Some DFR-Attribute modifications are also automatically applied by the DFR-Server (see clause 9). Consequential changes to the DFR-Attributes of the destination DFR-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 can not be used to copy a DFR-Group from another DFR-Document-Store.

Whenever the security policy defined in 6.3.8 is applied, the DFR-Access-List attribute for the newly created DFR-Object (the copy) and for all the newly created Descendants is not taken from the original(s), but is supplied explicitly by the requesting DFR-User or is 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, the 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,
        AccessError,
        AttributeError,
        InterServerAccessError,
        NameError,
        ReferentAccessError,
        SecurityError,
        ServiceError,
        UpdateError,
        VersionManagementError}
```

8.2.3.1 Copy-argument

```
CopyArgument :: = SEQUENCE {
    COMPONENTS OF
        CommonUpdateArguments (WITH COMPONENTS { ... ,
                               entry PRESENT,
                               destination PRESENT, reference-qos ABSENT } ),
                    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 a DOR 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 DFR-Attributes and/or DFR-Content of the new DFR-Entry (the copy), as compared to the original; no modifications will be applied to any Descendants 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;
- reference-qos is not applicable.

b) Common arguments:

- task-id, see 8.1.3.1;
- reservation, if requested, applies to the newly created DFR-Entry (see 8.1.3.2);
- error-handling, any of four modes (**ErrorHandlingMode**) can be specified (see 8.1.3.3) when copying a DFR-Group; when until-first-warning mode is specified the result is DFR-Server implementation dependent; when copying a single DFR-Object, only the all-or-nothing mode applies (taken by default), but the referent modified problem will not be reported as a **ReferentAccessProblem** and the **Copy** abstract operation will in this case not be terminated (the modification of the Referent is reported in the CopyResult in the referent-altered parameter in this case);
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

8.2.3.2 Copy-result

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

```
CopyResult ::= CommonUpdateResult (WITH COMPONENTS { ... , reference-qos ABSENT } )
```

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 (DFR-Attributes and/or DFR-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;
- referent-altered indicates if (one of) the Referents from which items are to be taken (as updates for the copied DFR-Entry) are changed or not (after the DOR produce time), it is not returned if a DFR-Group was copied.

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 DFR-Group is deleted. This DFR-Object ceases to be a Member of the source DFR-Group and becomes a Member of the destination DFR-Group. Modifications are applied to the moved DFR-Object if requested by the DFR-User (see 8.1.5.5). Some DFR-Attribute modifications are also automatically applied by the DFR-Server (see clause 9). Consequential changes to the DFR-Attributes of the source DFR-Group and to the DFR-Attributes of the destination DFR-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 Descendants are moved with it regardless of the requesting DFR-User´s access rights to each of them, or reservation made by other DFR-Users. This abstract operation can not be used to move a DFR-Object from another DFR-Document-Store. The DFR-Root-Group can not be moved.

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

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

8.2.4.1 Move-argument

```
MoveArgument :: = SEQUENCE {
    COMPONENTS OF
        CommonUpdateArguments (WITH COMPONENTS { ... ,
                                entry PRESENT,
                                destination PRESENT , reference-qos ABSENT } ),
                    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 (to which 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 DFR-Attributes and/or DFR-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 the move has been performed;
- reference-qos is not applicable.

b) Common arguments:

- task-id, see 8.1.3.1;
- reservation, if requested, applies to the newly created DFR-Entry (by default the previously defined reservation remains in force); reservations applied to any Descendant of the moved DFR-Object are not changed;
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

8.2.4.2 Move-result

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

```
MoveResult :: = CommonUpdateResult (WITH COMPONENTS { ... , warnings ABSENT, reference-qos ABSENT } )
```

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

- upi is the **DfrUniquePermanentIdentifier** of the newly created DFR-Entry (the UPI of the moved DFR-Entry remains unchanged);
- entry-information returns all those items from the new DFR-Entry (DFR-Attributes and/or DFR-Content) which have been requested by the selection component of the move-argument, and which are present in the DFR-Entry;
- referent-altered indicates if (one of) the Referents from which items are to be taken (as updates for the moved DFR-Object) are changed or not (after the DOR produce time).

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 DFR-Attributes and/or DFR-Content of a DFR-Entry. Some or all DFR-Attributes can be read. The content if requested is returned entirely. The content of a DFR-

Group and DFR-Search-Result-List shall not be requested. If required, the attributes and content of the Referent can be accessed via the DFR-Reference(with the same restriction as above).

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

8.2.5.1 Read-argument

```
ReadArgument :: = SEQUENCE {
    COMPONENTS OF
        CommonUpdateArguments (WITH COMPONENTS
                    { entry PRESENT, selection PRESENT } ),
        dereferencing       [7] BOOLEAN DEFAULT FALSE,
        token               [8] Token OPTIONAL,  -- imported from DOR --
        COMPONENTS OF CommonArguments (WITH COMPONENTS { ... ,
                                    error-handling ABSENT } ) }
```

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

a) Common update arguments:

- entry gives the **DfrEntryName** or a DOR (if a DOR is provided no DOR shall be requested in the selection parameter) of the DFR-Entry to be read;
- selection specifies which items from the DFR-Entry are to be read;
- reference-qos specifies, in the case a DOR is requested, the desired QoS for the DOR.

b) Specific **Read** arguments:

- 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-Reference dereferencing implies an ROA-operation;
- token is optionally specified if dereferencing is requested (only applicable if a token was in the DOR at produce time).

b) Common arguments:

- task-id, see 8.1.3.1;
- reservation, applies to the DFR-Entry specified in the request;
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

8.2.5.2 Read-result

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

```
ReadResult :: = CommonUpdateResult
    (WITH COMPONENTS { ... , entry-information PRESENT, warnings ABSENT } )
```

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

- upi is the **DfrUniquePermanentIdentifier** of the DFR-Entry read;
- 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;
- reference-qos returns the QoS assigned to the DOR by the DFR-Server (that can be different from the reference-qos specified as a Read argument), it is only returned if a DOR was requested;
- referent-altered indicates if the Referent is changed or not (after the DOR produce time), it is only returned if the DFR-Entry read is a DFR-Reference and dereferencing was requested.

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 DFR-Attributes and/or DFR-Content of the specified DFR-Entry. Existing DFR-Attributes that are not specified in the argument are left unchanged. Affected DFR-Server maintained attributes of the parent DFR-Group are changed. DFR-Content can be modified only for a DFR-Document. Modification of a DFR-Content means the complete DFR-Content is replaced. DFR-Content can be provided explicitly or taken from a specified existing source DFR-Entry.

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

8.2.6.1 Modify-argument

```
ModifyArgument :: = SEQUENCE {
    COMPONENTS OF
        CommonUpdateArguments (WITH COMPONENTS { ... ,
            entry PRESENT, destination ABSENT, position ABSENT,
            modifications PRESENT, reference-qos ABSENT } ),
    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 DFR-Attributes and/or DFR-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;
- reference-qos is not applicable.

b) Common arguments:

- task-id, see 8.1.3.1;
- reservation, if requested, applies to the DFR-Entry being modified;
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

8.2.6.2 Modify-result

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

```
ModifyResult :: = CommonUpdateResult (WITH COMPONENTS { ... , warnings ABSENT, reference-qos ABSENT } )
```

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

- upi is the **DfrUniquePermanentIdentifier** of the DFR-Entry modified;
- entry-information returns all those items from the DFR-Entry modified (DFR-Attributes and/or DFR-Content) which have been requested by the selection component of the **ModifyArgument**, and which are present in the DFR-Entry;
- reference-altered indicates if (one of) the Referents from which items are to be taken for the modifications, are changed or not (after the DOR produce time).

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 DFR-Attributes for the Members of a specified DFR-Group or for the DFR-Entries identified by the elements of a specified DFR-Search-Result-List. For a DFR-Group only the Members are listed, not their Descendants. Members of the DFR-Group or DFR-Entries identified by the elements of the DFR-Search-Result-List which are not accessible (insufficient access rights) to the initiating DFR-User are not included in this list (other Members or entries identified by the elements are however returned provided that an appropriate **ErrorHandlingMode** is specified). The number of Members or elements (DFR-Entries) to be listed can be limited by the count-limit argument. If this limit would be exceeded the listing of Members or elements (DFR-Entries) can be continued with the next **List** abstract operation. For each Member or element (DFR-Entries) only specified DFR-Attributes are returned. The Members or elements (DFR-Entries) are listed in the specified order.

```
List :: = ABSTRACT-OPERATION
    ARGUMENT        ListArgument
    RESULT          ListResult
    ERRORS {
        Abandoned,
        AccessError,
        AttributeError,
        NameError,
        ReservationError,
        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** arguments:

- entry identifies the DFR-Group or the DFR-Search-Result-List to be listed.

b) Common **List/Search** arguments:

- continue, if set to TRUE, specifies the list process, identified by task-id in the **CommonArguments**, shall be continued from where it has previously terminated;
- limits optionally specifies the maximum number of DFR-Entries to be returned for this abstract operation;

- selection specifies which DFR-Attributes of each DFR-Entry shall be returned in the result of the **List** abstract operation;
- ordering optionally specifies in which order the DFR-Entries shall be put in the list-result (this ordering, if specified, overrides the predefined ordering for the DFR-Group or the DFR-Search-Result-List listed).

c) Common arguments:

- task-id, see 8.1.3.1;
- reservation, if requested, applies to the DFR-Entry to be listed;
- error-handling, any of four modes (**ErrorHandlingMode**) can be specified (see 8.1.3.3); 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 a warning is encountered;
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

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 the result of the **List** abstract operation;
- limit-encountered optionally indicates that the maximum number of entries has been encountered; it is absent if the maximum number of entries has not been specified in the **ListArgument**;
- 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 DFR-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;
- continuation optionally provides an identifier for the continuation, if the list process has terminated, after the limit (specified in the **ListArgument**) has been encountered; it is present only when limit-encountered component is also present.

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 DFR-Entries 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 (insufficient access rights) 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 matching DFR-Entries noted in the resulting **DfrEntryList** can 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. The Search domain is always bound to the DFR-Document-Store. DFR-References to external Referents are never dereferenced even if a dereferencing depth is specified.

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

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 possibly 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, where 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, also the      --
    --result 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:

- continue, if set to TRUE, specifies the search process, identified by task-id in the **CommonArguments**, shall be continued from where it has previously terminated;
- 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 optionally specifies which DFR-Attributes of each DFR-Entry found in this **Search** abstract operation shall be returned in the entry-list parameter of the result of this abstract operation;
- ordering optionally specifies in which order the DFR-Entries shall be put in the result of the **Search** abstract operation.

c) Common arguments:

- task-id, see 8.1.3.1;
- reservation, if requested, applies to the DFR-Search-Result-List if specified in the search-mode parameter;
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

8.2.8.2 Search-result

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

```
SearchResult :: = SEQUENCE { COMPONENTS OF
                                CommonListSearchResult
                                    (WITH COMPONENTS { ... , warnings ABSENT } ) ,
                                removed-entries [5] DfrEntryList OPTIONAL }
```

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

a) Common list/search result:

- number-of-entries gives the number of DFR-Entries found in this search;
- 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 **SearchArgument**, 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 DFR-Attributes are returned which have been requested by the selection component in the **SearchArgument**;
- continuation optionally provides an identifier for the continuation, if the search process has terminated, after one of the limits (specified in the **SearchArgument**) has been encountered; it is present only when limit-encountered component is also present.

b) Specific **Search** result:

- 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, class and possibly ordering-attribute components;

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-Entry specified. There are five levels of reservation defined. The reservation levels are described in 8.1.3.2. A reservation never affects read access to the attributes of the DFR-Entries, that is, all users can always read the DFR-Attributes of DFR-Objects. The reservation level of a DFR-Entry can only be changed by a **Resere** 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.

```
Reserve :: = ABSTRACT-OPERATION
    ARGUMENT        ReserveArgument
    RESULT          ReserveResult
    ERRORS {
        Abandoned,
        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 } ) }
```

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

a) specific Reserve arguments:

- entry gives the **DfrEntryName** of the DFR-Entry to be reserved.

b) Common arguments:

- task-id, see 8.1.3.1;
- reservation specifies the level of reservation;
- priority, see 8.1.3.4;
- privileges, see 8.1.3.5.

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 or in the continuation of a previously performed abstract operation. The DFR-Server shall cease processing the outstanding abstract operation, and shall discard any result so far achieved, or shall discard any internal information preserved for the continuation of an abstract operation.

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

8.2.10.1 Abandon-argument

```
AbandonArgument :: = CHOICE {
    task-id         [0] TaskId,
    invoke-id       [1] InvokeIDType }
```

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

- task-id can be used to identify an outstanding abstract operation to be abandoned or to indicate to the DFR-Server that any internal information possibly preserved for the continuation of a **List** or **Search** abstract operation shall be discarded (see also 8.1.3.1);

- invoke-id can be used to identify an outstanding 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 can be reported in response to the invocation of abstract operations at the DFR-Server port are defined and described in this clause.

If any error occurs during the course of an abstract operation invocation, the DFR-Server assures that nothing has been changed by the abstract operation (this does not apply, however, to warnings, reported by the DFR-Server as part of a normal result of a DFR abstract operation; see 8.1.3.3). 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 DFR-Entry and the destination DFR-Group in a **Copy** abstract operation), then an abstract error can 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 can be reported in response to a single operation invocation.

8.3.1 Attribute-error

An **AttributeError** reports one or more 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 behaviour 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, can not be modified in the way specified in the abstract operation (the rules are specified in clause 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 shall 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 DFR-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. If a DFR-User has no read access right to a DFR-Entry a **NameError** (i.e. **NameProblem** 1 or 2) will be reported in order to hide the existence of that DFR-Entry.

```
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** shall 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 DS;

c) **ambiguous-path-name**: The **DfrPathName** (either absolute or relative) provided in the abstract operation´s 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),
reserved-by-a-user            (2),
externally-located-object     (3) }
```

An **AccessProblem** shall 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) **reserved-by-a-user**: The DFR-Entry to be accessed or its Parent is at present reserved by the same or another user;

c) **externally-located-object**: The DFR-Entry to be accessed is located in another DFR--Document-Store and 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 can 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-a-user                  (3),
illegal-content-modification        (4),
group-membership-criteria-violation (5),
reference-loop-detected             (6) }
```

An **UpdateProblem** shall 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 modify a DFR-Entry has been made by a DFR-User with insufficient access rights to this entry. If a DFR-User has no read access right to a DFR-Entry a **NameError** (see 8.3.2) will be reported in order to hide the existence of that DFR-Entry;

c) **reserved-by-a-user**: The DFR-Entry to be modified or its Parent is at present reserved by the same or another DFR-User;

d) **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);

e) **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 DFR-Membership-Criteria defined for that DFR-Group. Or an attempt has been made to modify the DFR-Membership-Criteria attribute in such a way that some or all existing Members would no longer satisfy the DFR-Membership-Criteria;

f) **reference-loop-detected**: An attempt has been made to create a new DFR-Reference or to modify an existing one in such 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.
This part of ISO/IEC 10166 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 DS.

8.3.5 ReferentAccess-error

A **ReferentAccessError** reports a problem occurring when attempting to access a Referent.

```
ReferentAccessError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE {
        entry     [0] CHOICE {  dfr-entry [0] DfrEntryName,
                                dor       [1] NULL }
        problem   [1] ReferentAccessProblem }

ReferentAccessProblem :: = ENUMERATED {

    inappropriate-object-class      (1),
    insufficient-access-rights      (2),
    reserved-by-a-user              (3),
    referent-no-longer-exists       (4),
    referent-modified               (5),  -- only used in the case a DFR-Group is copied or listed and --
                                          -- only if warnings shall be reported, i.e. referent modified will --
                                          -- not terminate the operation --
    reference-content-empty         (6) }
```

A **ReferentAccessProblem** shall 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 access the Referent of a DFR-Reference has been made by a DFR-User with insufficient access rights to the Referent;

c) **reserved-by-a-user**: The Referent to be accessed or its Parent is at present reserved by the same or another DFR-User;

d) **referent-no-longer-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 a DFR-Reference to an external Referent (the accessor) has made an attempt to access the accessee containing the Referent.

NOTE - This access may be initiated using the ROA-protocol between the two servers; but the access problems specified hereafter are not specific to the ROA-protocol.

```
InterServerAccessError :: = ABSTRACT-ERROR
    PARAMETER SEQUENCE {
        entry     [0] DfrEntryName, --of the 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** shall be one of the following:

a) **referent-store-not-found**: The accessee, as it is specified in the content of the DFR-Reference, has not been found;

NOTE - This problem can be caused by bad content of the reference, or by a bad entry in the Directory describing the current location of the accessee, or by any other access problem between the accessor and the related Directory server;

b) **referent-store-unreachable**: The DFR-Server managing the accessee has no protocol in common with the accessor, through which the Referent could be accessed;

c) **referent-store-unavailable**: The accessee 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 accessee has insufficient access rights for doing this, at least through the given accessor (this can also be caused by insufficient access rights of the accessor 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),
    cannot-unreserve          (3),
    reservation-not-changed   (4) }
```

A **ReservationProblem** shall 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 or **Reservation** status. If a DFR-User has no read access right to a DFR-Entry a **NameError** (see 8.3.2) will be reported in order to hide the existence of that DFR-Entry;

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;

d) **cannot-unreserve**: An attempt has been made to unreserve a DFR-Entry which has the Reservation status "committed ";

e) **reservation-not-changed**: An attempt has been made to change the Reservation level to a lower level of a DFR-Entry that has the Resevation status "committed" or the Reservation status should be changed from "committed" to "uncommitted".

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).If a DFR-User has no read access right to a DFR-Entry specified as previous version, when creating a new version, a **NameError** (see 8.3.2) will be reported in order to hide the existence of that DFR-Entry.

```
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 --
    belongs-to-another-conceptual-document (2) }
```

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

a) **inappropriate-object-class**: An attempt has been made to declare as a Version some DFR-Entry which is not a DFR-Document;

b) **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 previous Versions 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 can 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),
invalid-pac             (4),
already-active          (5)  }
```

A SecurityProblem shall 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) **invalid-pac**: the supplied PAC is invalid;

e) **already-active**: the user, identified by credentials (during binding) or privileges (associated with an abstract operation), is already bound to the DFR-Server.

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. search-criteria--
    resource-limit-exceeded     (4),  --e.g. when creating a bulky object--
    maximum-length-exceeded     (5),  -- in an abstract operation --
    cannot-continue             (6),  -- e.g. search-domain altered --
    unclassified-server-error   (7),  --implementation specific--
    function-set-violation      (8) } -- see 8.4 --
```

A **ServiceProblem** reported shall 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 search-criteria in a Search abstract operation has too many nesting levels to be correctly understood by the given DFR-Server);

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

e) **maximum-length-exceeded**: The argument provided by the DFR-User or the result prepared by the DFR-Server exceeds the maximum length negotiated during binding;

f) **cannot-continue**: The DFR-Server cannot continue because of changes in the DS after the previous result of the respective abstract operation;

g) **unclassified-server-error**: An error which can not be categorized in any other way. The reason for this error is implementation specific, and out of the scope of this part of ISO/IEC 10166;

h) **function-set-violation**: The abstract operation cannot be performed due to the restrictions of the function set (see 8.4). Which function set the DFR-Server supports is optionally reported during binding.

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 }

InvokeId :: = INTEGER

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

An **AbandonProblem** shall 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 cannot be abandoned, or such an abandon would make no sense;

c) **cannot-abandon**: The abstract operation specified can not be abandoned because of an implementation specific reason.

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 can instead be considered as a success report of the related Abandon abstract operation. It can 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 the user´s viewpoint, it is necessary to define functional sets of this structure. Thus, the following usage types and corresponding function-sets are determined when a DS is installed:

Usage Type 1: Flat Store Set

In this usage type, the structure of a DFR-Document Store is one group, single level. 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, DFR-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, 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 3 : DFR Attributes

9 Attribute Definitions

9.1 Overview of Attributes

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

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

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

The attributes defined in this part of ISO/IEC 10166 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 annex A). DFR gives support to a subset of ODA attributes from the ODA Document Profile [ISO 8613-4]. These attributes are considered not to be specific to DFR and therefore their names appear without the "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 1 The consistency between ODA-Document-Profile attributes and their corresponding DFR-Attributes is beyond the scope of this International Standard.

The attribute types defined in this part of ISO/IEC 10166 are grouped into 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 2 There will eventually be different extension attribute sets, 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 part of ISO/IEC 10166) is included in the definition of DFR Application Context (see annex of ISO/IEC 10166-2); its presence however is also negotiable.

The attributes defined in this part of ISO/IEC 10166 are listed in the tables below. The details are given in the attribute type descriptions. Tables 3 and 5 show for the various attributes whether the attribute type is single-valued or multi-valued, by which entity the attribute type is managed in different situations and whether an attribute is automatically copied in a **Copy** abstract operation. Tables 4 and 6 show for the various attributes, to which DFR-Objects these attributes are assigned, their occurrences, and their matching behavior in **Search** abstract operations including their availability for ordering.

Table 3 - DFR-Basic-Attribute-Set (Part 1 of 2)

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
DFR-Title	S	U	U	-	Y
DFR-Pathname	S	D	D	-	N
DFR-Parent-Identification	S	D	D	-	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

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

Table 3 - DFR-Basic-Attribute-Set (Part2 of 2)

Attribute-type name	Single-/ Multi-valued	Assigned by	Modified by	Deleted by	Autom. copied by Copy
Version-Name	S	U	-	U	N
DFR-Previous-Versions	M	U	D	U/D	N
DFR-Next-Versions	M	D	D	D	N
DFR-Version-Root	S	D	-	D	N
DFR-External-Location	S	U	U	U	Y
User-Reference	S	U	U	U	Y
User-References-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-Modified-By	S	D	D	-	N
DFR-Content-Modified-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/D	O	-	N

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

Table 4: DFR-Basic-Attribute-Set (Part1 of 2)

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		O		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-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	

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

Table 4 - DFR-Basic-Attribute-Set (Part2 of 2)

Attribute types	Used for				Matches in Filter for		
	DO	GR	RE	SR	EQ	OR	SU
Version-Name	C		O		O		O
DFR-Previous-Versions	C				O		
DFR-Next-Versions	C				O		
DFR-Version-Root	C		O		O		
DFR-External-Location	O	O	O		M		M
User-Reference	O	O	O	O	O		O
User-References-To-Other-Objects	O	O	O	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-Modified-By	C	C	C	C	M		
DFR-Content-Modified-By	C	C	C	C	M		
Document-Date-And-Time	O		O		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		

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

Table 5 - DFR-Extension-Attribute-Set

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
Document-Architecture-Class	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
Preparers	M	U	U	U	Y
Owners	M	U	U	U	Y
Authors	M	U	U	U	Y
Status	S	U	U	U	Y
User-Specific-Codes	M	U	U	U	Y
Superseded-Documents	M	U	U	U	Y
Number-Of-Pages	S	U	U	U	Y
Languages	M	U	U	U	Y
S = SINGLE VALUE, M = MULTI VALUE, D = DFR-Server, U = DFR-User, O = OWNERS, Y = YES, N = NO					

Table 6 - DFR-Extension-Attribute-Set

Attribute-Type	Used for				Matches in Filter for		
	DO	GR	RE	SR	EQ	OR	SU
Other-Titles	O	O	O	O	M		M
Subject	O	O	O	O	M		M
Document-Type	O				M		M
Document-Architecture-Class	O		O		M	M	
Keywords	O	O	O	O	M		M
Creation-Date-And-Time	O	O	O	O	M	M	
Purge-Date-And-Time	O	O	O	O	M	M	
Revision-Date-And-Time	O	O	O	O	M	M	
Organizations	O		O		M		M
Preparers	O		O		M		M
Owners	O		O		M		M
Authors	O		O		M		M
Status	O		O		M		M
User-Specific-Codes	O	O	O	O	M		M
Superseded-Documents	O		O		M		M
Number-Of-Pages	O		O		M	M	
Languages	O		O		M		M

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

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, DFR-Reference, or DFR-Search-Result-List 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.

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

9.2.4 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
```

9.2.5 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
```

9.2.6 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 DFR-Group. This attribute is associated with each DFR-Object. For a DFR-Root-Group it is by convention an octet string of length 0 (zero).

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

9.2.7 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
```

9.2.8 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
```

9.2.9 DFR-Ordering

This attribute defines a default ordering of DFR-Group's members in a **List** abstract operation. The ordering rule is specified in 8.1.6.3. This attribute is associated only with DFR-Group objects.

```
dfr-ordering ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX OrderingRule
```

```
SINGLE VALUE
:: = id-att-dfr-ordering
```

9.2.10 DFR-Resource-Limit

This attribute specifies the maximum resource to be used for a DFR-Object based upon accounting information. The resource limit includes the space required to store content (if a Document), the DFR-Object-Tree (if a DFR-Group) and any associated attributes.

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

9.2.11 DFR-Resource-Used

This attribute contains information for accounting purposes based on resources used during some period of time, for example the actual amount used for the storage of the DFR-Object in the DS. The semantics of the integer value is implementation specific. The resource used includes the space required to store content (the DFR-Object-Tree in the case of a DFR-Group) and any associated attributes.

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

9.2.12 DFR-Number-Of-Group-Members

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

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

9.2.13 Version-Name

This is a free-form attribute intended for the DFR-User's use and is 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 6.3.6); but it can also be used for any other DFR-Document. It can 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
```

9.2.14 DFR-Previous-Versions

This is a multi-valued attribute. It is defined only for DFR-Documents (see also 6.3.6). It is assigned by the DFR-User when the document is declared a new Version (in a **Create** or **Modify** abstract operation). It can then be modified by the DFR-Server 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). When the value of this attribute is read by the DFR-User, only those DFR-Documents to which this DFR-User has at least read access right are included in the result of a DFR abstract operation.

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

9.2.15 DFR-Next-Versions

This is a multi-valued attribute. It is defined only for DFR-Documents (see also 6.3.6). 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. When the value of this attribute is read by the DFR-User, only those DFR-Documents to which this DFR-User has at least read access right are included in the result of a DFR abstract operation.

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

9.2.16 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 6.3.6), and optionally, for DFR-References to these documents. It is assigned 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.

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

9.2.17 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
```

9.2.18 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-References-To-Other-Objects can 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 a DFR-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
```

9.2.19 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-References-To-Other-Objects can be used to establish DFR-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 a DFR-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 can 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
```

9.2.20 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 generalizedTimeSyntax
        SINGLE VALUE
        :: = id-att-dfr-attributes-create-date-and-time
```

9.2.21 DFR-Content-Create-Date-And-Time

This attribute contains the date and time when the DFR-Content of this DFR-Object was stored in a DFR-Document-Store. A DFR-Server will set it to the current date and time when the DFR-Content of this DFR-Object is created.

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

9.2.22 DFR-Created-By

This attribute identifies the DFR-User which created this DFR-Object; it is not modified when the DFR-Object is moved. It can 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
```

9.2.23 DFR-Attributes-Modify-Date-And-Time

This attribute contains the date and time when 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 generalizedTimeSyntax
        SINGLE VALUE
        :: = id-att-dfr-attributes-modify-date-and-time
```

9.2.24 DFR-Content-Modify-Date-And-Time

This attribute contains the date and time when 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 generalizedTimeSyntax
        SINGLE VALUE
        :: = id-att-dfr-content-modify-date-and-time
```

9.2.25 DFR-Attributes-Modified-By

This attribute identifies the DFR-User which has most recently modified the DFR-Attributes of that DFR-Object. It can only be read by a DFR-User having at least extended-read access rights.

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

9.2.26 DFR-Content-Modified-By

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

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

9.2.27 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 generalizedTimeSyntax
        SINGLE VALUE
        :: = id-att-document-date-and-time
```

9.2.28 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
```

9.2.29 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 can only be read by a DFR-User having at least extended-read access rights.

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

9.2.30 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 rights to this DFR-Object; it can be modified only by DFR-Users having the owner access rights for this DFR-Object. A DFR-User having only read access rights to the DFR-Object can only read its own access rights 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 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 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
```

9.3.5 Keywords

This attribute specifies one or more character strings that permit logical associations to be made about the content 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".

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

9.3.6 Creation-Date-And-Time

This attribute specifies the date and, optionally, the time of day when the 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 generalizedTimeSyntax
        SINGLE VALUE
        :: = id-att-creation-date-and-time
```

9.3.7 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 generalizedTimeSyntax
        SINGLE VALUE
        :: = id-att-purge-date-and-time
```

9.3.8 Revision-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".

```
revision-date-and-time ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX generalizedTimeSyntax
        SINGLE VALUE
        :: = id-att-revision-date-and-time
```

9.3.9 Organizations

This attribute identifies the originating organization(s) associated with 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".

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

9.3.10 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
        MATCHES FOR EQUALITY  SUBSTRINGS
        MULTI VALUE
        :: = id-att-preparers
```

9.3.11 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
        MATCHES FOR EQUALITY  SUBSTRINGS
        MULTI VALUE
        :: = id-att-owners
```

9.3.12 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
        MATCHES FOR EQUALITY  SUBSTRINGS
        MULTI VALUE
        :: = id-att-authors
```

9.3.13 Status

This attribute specifies the 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
```

9.3.14 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
```

9.3.15 Superseded-Documents

This attribute specifies reference(s) to document(s) superseded by the current 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
```

9.3.16 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
        MATCHES FOR EQUALITY ORDERING
        SINGLE VALUE
        ::= id-att-number-of-pages
```

9.3.17 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 caseIgnoreStringSyntax
        MULTI VALUE
        ::= id-att-languages
```

9.4 DFR Attribute Syntaxes

The following attribute syntaxes are used in the preceding attribute definitions.

9.4.1 String Attribute Syntaxes

In the syntaxes specified in this clause, 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 shall be the same length and corresponding characters shall 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 character set then matching fails. Similar rules apply when comparing a Graphic String with a Printable String, a T.61 String or a General String.

9.4.1.2 Case Ignore List

The Case Ignore List attribute syntax is intended for attributes whose values are sequences of 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 the values of which 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 Generalized Time

The GeneralizedTime attribute syntax is intended for attributes the values of which represent absolute time.

```
generalizedTimeSyntax ATTRIBUTE-SYNTAX
    GeneralizedTime
    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.

9.4.2.4 Person

The Person attribute syntax is intended for attributes whose value represent a person.

```
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 CharacterData OPTIONAL }
```

Section 4 : 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 description of the operations is in clause 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 can have more than one abstract operation outstanding, that is parallel performance of the abstract operations can 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:

a) checks that the DFR-User has the access rights as required;

b) creates a new DFR-Entry in the DFR-Document-Store placing the new DFR-Object in the appointed DFR-Group;

c) establishes the DFR-Attribute values for the new DFR-Object as appropriate;

d) establishes the value of the DFR-Content of the new DFR-Object as appropriate;

e) 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:

a) checks that the DFR-User has the access rights as required;

b) deletes the DFR-Entry removing the DFR-Object from its parent DFR-Group;

c) 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:

a) checks that the DFR-User has the access rights as required;

b) creates a new DFR-Entry in the destination DFR-Group;

c) copies the values of the DFR-Attributes and the DFR-Content of the appointed DFR-Object(s) as appropriate;

d) updates the DFR-Attributes of the destination DFR-Object as appropriate;

e) 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:

a) checks that the DFR-User has the access rights as required;

b) creates a new DFR-Entry in the destination DFR-Group;

c) changes the membership of the appointed DFR-Object from the source DFR-Group to the destination DFR-Group;

d) updates the DFR-Attributes of both the source and destination as appropriate;

e) 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:

a) checks that the DFR-User has the access rights as required;

b) 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:

a) checks that the DFR-User has the access rights as required;

b) modifies the DFR-Attribute(s) and/or the DFR-Content of the appointed DFR-Object as appropriate;

c) returns the **Modify** result to the DFR-User.

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:

a) checks that the DFR-User has the access rights as required;

b) 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:

a) checks that the DFR-User has the access rights as required;

b) searches for the DFR-Objects as specified;

c) stores, if requested, the result of the search in the appointed DFR-Search-Result-List;

d) 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:

a) checks that the DFR-User has the access rights as required;

b) changes the reservation level of the appointed DFR-Object as appropriate;

c) 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:

a) checks that the DFR-User has the access rights as required;

b) abandons the appointed, previously invoked DFR abstract operation as appropriate;

c) forces the abstract-error **Abandoned** to be returned for the abstract operation which has been abandoned.

11 Port Realization

This clause describes how the Document Filing and Retrieval port of the Document Filing and Retrieval abstract service is provided.

The Document Filing and Retrieval port abstract service is realized on a one-to-one basis between abstract operations defined in this part of ISO/IEC 10166 and the real operations in the Document Filing and Retrieval Service Element (DFRSE) specified in ISO/IEC 10166-2.

Annex A
(informative)

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

Annex B
(normative)

Formal Assignment of Object Identifiers

All Object Identifiers that this part of ISO/IEC 10166 assigns are formally assigned in this annex using ASN.1. The specified values are cited in the ASN.1 modules of subsequent annexes.

This annex is definitive for all values except those for ASN.1 modules of this part of ISO/IEC 10166. The definitive assignments for those occur in the modules themselves.

```
DFRObjectIdentifiers { iso standard 10166 part-1(1) modules (0) object-identifiers(0)}
DEFINITIONS ::=
BEGIN

-- PROLOGUE --

-- EXPORTS everything --

ID ::= OBJECT IDENTIFIER
id-dfr ID ::= { iso standard 10166 part-1(1)}

-- Categories --

id-mod          -- modules --                 ID ::= {id-dfr 0}
id-ot           -- objects --                 ID ::= {id-dfr 1}
id-pt           -- ports --                   ID ::= {id-dfr 2}
id-dfr-oc       -- object classes --          ID ::= {id-dfr 3}
id-dfr-bas-att  -- dfr basic attributes --    ID ::= {id-dfr 4}
id-dfr-ext-att  -- dfr extension attributes -- ID ::= {id-dfr 5}
id-dfr-att-syn  -- dfr attribute syntaxes --  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-reference              ID :: = {id-dfr-oc 3}
id-dfr-search-result-list     ID :: = {id-dfr-oc 4}
```

-- 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-dfr-title                                ID :: = {id-dfr-bas-att 3}
id-att-dfr-pathname                             ID :: = {id-dfr-bas-att 4}
id-att-dfr-parent-identification                ID :: = {id-dfr-bas-att 5}
id-att-dfr-referent-deleted                     ID :: = {id-dfr-bas-att 6}
id-att-dfr-membership-criteria                  ID :: = {id-dfr-bas-att 7}
id-att-dfr-ordering                             ID :: = {id-dfr-bas-att 8}
id-att-dfr-resource-limit                       ID :: = {id-dfr-bas-att 9}
id-att-dfr-resource-used                        ID :: = {id-dfr-bas-att 10}
id-att-dfr-number-of-group-members              ID :: = {id-dfr-bas-att 11}
id-att-version-name                             ID :: = {id-dfr-bas-att 12}
id-att-dfr-previous-versions                    ID :: = {id-dfr-bas-att 13}
id-att-dfr-next-versions                        ID :: = {id-dfr-bas-att 14}
id-att-dfr-version-root                         ID :: = {id-dfr-bas-att 15}
id-att-dfr-external-location                    ID :: = {id-dfr-bas-att 16}
id-att-user-reference                           ID :: = {id-dfr-bas-att 17}
id-att-user-reference-to-other-objects          ID :: = {id-dfr-bas-att 18}
id-att-dfr-attributes-create-date-and-time      ID :: = {id-dfr-bas-att 19}
id-att-dfr-content-create-date-and-time         ID :: = {id-dfr-bas-att 20}
id-att-dfr-created-by                           ID :: = {id-dfr-bas-att 21}
id-att-dfr-attributes-modify-date-and-time      ID :: = {id-dfr-bas-att 22}
id-att-dfr-content-modify-date-and-time         ID :: = {id-dfr-bas-att 23}
id-att-dfr-attributes-modified-by               ID :: = {id-dfr-bas-att 24}
id-att-dfr-content-modified-by                  ID :: = {id-dfr-bas-att 25}
id-att-document-date-and-time                   ID :: = {id-dfr-bas-att 26}
id-att-dfr-reservation                          ID :: = {id-dfr-bas-att 27}
id-att-dfr-reserved-by                          ID :: = {id-dfr-bas-att 28}
id-att-dfr-access-list                          ID :: = {id-dfr-bas-att 29}
```

-- 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-document-architecture-class           ID :: = {id-dfr-ext-att 3}
id-att-keywords                              ID :: = {id-dfr-ext-att 4}
id-att-creation-date-and-time                ID :: = {id-dfr-ext-att 5}
id-att-purge-date-and-time                   ID :: = {id-dfr-ext-att 6}
id-att-revision-date-and-time                ID :: = {id-dfr-ext-att 7}
id-att-organizations                         ID :: = {id-dfr-ext-att 8}
id-att-preparers                             ID :: = {id-dfr-ext-att 9}
id-att-owners                                ID :: = {id-dfr-ext-att 10}
id-att-authors                               ID :: = {id-dfr-ext-att 11}
id-att-status                                ID :: = {id-dfr-ext-att 12}
id-att-user-specific-codes                   ID :: = {id-dfr-ext-att 13}
id-att-superseded-documents                  ID :: = {id-dfr-ext-att 14}
id-att-number-of-pages                       ID :: = {id-dfr-ext-att 15}
id-att-languages                             ID :: = {id-dfr-ext-att 16}
```

-- 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-gen-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 --*

Annex C
(normative)
Formal Definition of the DFR Abstract-service

```
DFRAbstractService { iso standard 10166 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-root-group, id-dfr-proper-group, id-dfr-reference, id-dfr-search-result-list,
id-dfr-server, id-dfr-user, id-pt-dfr
    FROM DFRObjectIdentifiers { iso standard 10166 part-1 (1) modules (0) object-identifiers (0)}

-- OSI Directory --

Attribute, AttributeType, AttributeValue, AttributeValueAssertion, DistinguishedName, Filter, FilterItem
    FROM InformationFramework { joint-iso-ccitt ds(5) modules(1) informationFramework(1) }

-- Distinguished Object Reference --

DOR, Requested-QoS-level
    FROM DOR-definition { joint-iso-ccitt dor(11) reference-definition (0)}

-- Invoke-Id --

InvokeIDType
    FROM Remote-Operations-APDUs { joint-iso-ccitt remote Operations (4) apdus (1) };
```

```
-- 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-reference         (3),
                                dfr-search-result-list (4)  }

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

DfrEntryAttributes :: = SET OF Attribute

DfrObjectContent :: = CHOICE {
     document-content                [0] DfrDocumentContent,
     root-group-content              [1] DfrGroupContent,
     proper-group-content            [2] DfrGroupContent,
     reference-content               [3] DfrReferenceContent,
     search-result-list-content      [4] DfrSearchResultListContent }

DfrUniquePermanentIdentifier :: = OCTET STRING
```