INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.841

(10/2000)

SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

Security

Information technology – Security techniques – Security information objects for access control

ITU-T Recommendation X.841

(Formerly CCITT Recommendation)

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

PUBLIC DATA NETWORKS	
Services and facilities	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalling and switching	X.50–X.89
Network aspects	X.90–X.149
Maintenance	X.150–X.179
Administrative arrangements	X.180–X.199
OPEN SYSTEMS INTERCONNECTION	
Model and notation	X.200–X.209
Service definitions	X.210–X.219
Connection-mode protocol specifications	X.220–X.229
Connectionless-mode protocol specifications	X.230–X.239
PICS proformas	X.240–X.259
Protocol Identification	X.260–X.269
Security Protocols	X.270–X.279
Layer Managed Objects	X.280–X.289
Conformance testing	X.290–X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300–X.349
Satellite data transmission systems	X.350–X.369
IP-based networks	X.370–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600–X.629
Efficiency	X.630–X.639
Quality of service	X.640–X.649
Naming, Addressing and Registration	X.650–X.679
Abstract Syntax Notation One (ASN.1)	X.680–X.699
OSI MANAGEMENT	
Systems Management framework and architecture	X.700–X.709
Management Communication Service and Protocol	X.710–X.719
Structure of Management Information	X.720–X.729
Management functions and ODMA functions	X.730–X.799
SECURITY	**X.800–X.849**
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850–X.859
Transaction processing	X.860–X.879
Remote operations	X.880–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999

For further details, please refer to the list of ITU-T Recommendations.

INTERNATIONAL STANDARD 15816

ITU-T RECOMMENDATION X.841

INFORMATION TECHNOLOGY – SECURITY TECHNIQUES – SECURITY INFORMATION OBJECTS FOR ACCESS CONTROL

Summary

This Recommendation | International Standard provides object definitions that are commonly needed in security standards to avoid multiple and different definitions of the same functionality. Precision in these definitions is achieved by use of the Abstract Syntax Notation One (ASN.1).

This Recommendation | International Standard covers only static aspects of Security Information Objects (SIOs).

Source

ITU-T Recommendation X.841 was prepared by ITU-T Study Group 7 (1997-2000) and approved by the World Telecommunication Standardization Assembly (Montreal, 27 Septembre – 6 October 2000). An identical text is also published as ISO/IEC 15816.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Page

Introduction

This Recommendation | International Standard on Security Information Objects (SIOs) for Access Control provides object definitions that are commonly needed in more than one security standard such that multiple and different definitions of the same functionality may be avoided. Precision in these definitions is achieved by use of the Abstract Syntax Notation One (ASN.1) defined in ITU-T Rec. X.680 (1997) | ISO/IEC 8824-1:1998, and ITU-T Rec. X.681 (1997) | ISO/IEC 8824-2:1998.

The aim of security management is to ensure that assets, including information, are protected appropriately and cost effectively. In order to protect proprietary interests and Intellectual Property Rights, organizations need to control the handling of their information. Severe damage or embarrassment can be caused to either the originator or holder of sensitive information, for example, if it is released to those not authorized to receive it (a breach of confidentiality), or if it is modified in any way (a breach of integrity). Each organization needs to ensure that it protects its own information and assets adequately in all forms during its storage, processing and transmission between and within organizations over both private and public networks. Organizations must be satisfied that their assets will be protected properly when they are held or processed by others if business is to be conducted more widely.

The motivation for development of SIOs for Access Control is the achievement of the flexibility and interoperability in security management that accrues from the use of common structures for similar functions. Standardization of security labels and alternative methods for access control have been pursued in this Recommendation | International Standard.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – SECURITY TECHNIQUES – SECURITY INFORMATION OBJECTS FOR ACCESS CONTROL

1 Scope

The scope of this Recommendation | International Standard is:

a) the definition of guidelines for specifying the abstract syntax of generic and specific Security Information Objects (SIOs) for Access Control;

b) the specification of generic SIOs for Access Control;

c) the specification of specific SIOs for Access Control.

The scope of this Recommendation | International Standard covers only the "statics" of SIOs through syntactic definitions in terms of ASN.1 descriptions and additional semantic explanations. It does not cover the "dynamics" of SIOs, for example rules relating to their creation and deletion. The dynamics of SIOs are a local implementation issue.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

– ITU-T Recommendation X.411 (1999) | ISO/IEC 10021-4, *Information technology – Message Handling Systems (MHS): Message transfer system: Abstract service definition and procedures.*

– ITU-T Recommendation X.500 (2001) | ISO/IEC 9594-1:2001, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services.*

– ITU-T Recommendation X.501 (2001) | ISO/IEC 9594-2:2001, *Information technology – Open Systems Interconnection – The Directory: Models.*

– ITU-T Recommendation X.509 (2000) | ISO/IEC 9594-8:2001, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks.*

– ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, *Information technology – Abstract syntax notation one (ASN.1): Specification of basic notation.*

– ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2:1998, *Information technology – Abstract syntax notation one (ASN.1): Information object specification.*

– ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3:1998, *Information technology – Abstract syntax notation one (ASN.1): Constraint specification.*

– ITU-T Recommendation X.683 (1997) | ISO/IEC 8824-4:1998, *Information technology – Abstract syntax notation one (ASN.1): Parameterization of ASN.1 specifications.*

– ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1:1998, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

- CCITT Recommendation X.722 (1992) | ISO/IEC 10165-4:1992, *Information technology – Open Systems Interconnection – Structure of management information: Guidelines for the definition of managed objects.*
- ITU-T Recommendation X.741 (1995) | ISO/IEC 10164-9:1995, *Information technology – Open Systems Interconnection – Systems Management: Objects and attributes for access control.*
- ITU-T Recommendation X.803 (1994) | ISO/IEC 10745:1995, *Information technology – Open Systems Interconnection – Upper layers security model.*
- ITU-T Recommendation X.810 (1995) | ISO/IEC 10181-1:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Overview.*
- ITU-T Recommendation X.830 (1995) | ISO/IEC 11586-1:1996, *Information technology – Open Systems Interconnection – Generic upper layers security: Overview, models and notation.*

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.800 (1991), *Security architecture for Open Systems Interconnection for CCITT applications.*

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

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 **Compartmentalization**: As defined in ISO/IEC DIS 2382-8.

3.2 **Generic SIO Class**: An SIO Class in which the data types for one or more of the components are not fully specified.

3.3 **Information Object**: As defined in ITU-T Rec. X.681 | ISO/IEC 8824-2.

3.4 **Information Object Class**: As defined in ITU-T Rec. X.681 | ISO/IEC 8824-2.

3.5 **Object Identifier (OID)**: As defined in ITU-T Rec. X.680 | ISO/IEC 8824-1.

3.6 **Seal**: As defined in ITU-T Rec. X.810 | ISO/IEC 10181-1.

3.7 **Security Authority**: The entity accountable for the administration of a security policy within a security domain.

3.8 **Security Domain**: A collection of users and systems subject to a common security policy.

3.9 **Security Information Object**: An instance of an SIO Class.

3.10 **Security Information Object Class**: An Information Object Class that has been tailored for security use.

3.11 **Security Label**: As defined in CCITT Rec. X.800 and ISO/IEC 7498-2.

3.12 **Security Policy**: As defined in ISO/IEC DIS 2382-8.

3.13 **Security Policy Information File**: A construct that conveys domain-specific security policy information.

3.14 **Specific SIO Class**: An SIO Class in which the data types for all components are fully specified.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

ASN.1	Abstract Syntax Notation One
EE	End Entity
IT	Information Technology

OID	Object Identifier
RBAC	Rule Based Access Control
SIO	Security Information Object
SPIF	Security Policy Information File

5 Conventions

5.1 Security Information Object Class Description

An SIO Class comprises:

- a value for a SIO Class identifier;
- a set of one or more data type specifications, one for each component the SIO Class contains; and
- a statement of the semantics associated with use of the SIO Class.

5.2 Generic Security Information Object Class Correspondence

A Generic SIO Class is an SIO Class in which the data types for one or more of the components are not fully specified. A Specific SIO Class is an SIO Class in which the data types for all components are fully specified. A generic SIO Class corresponds to a family of specific SIO Classes.

5.3 Security Information Object Composition

The specification of each SIO in this Recommendation | International Standard contains the following parts:

- a description of the SIO;
- an explanation of the usage of the SIO;
- a description of the components of the SIO.

The description of the components of the SIO includes the ASN.1 specification and the object identifier of the object class being defined.

6 Specification of Security Information Objects

When a new requirement is identified for an SIO, the following steps shall be followed to encourage reuse of existing specifications and to reduce the proliferation of different specifications meeting the same requirements:

- If this Recommendation | International Standard defines an SIO that meets the new requirement, the definition in this Recommendation | International Standard shall be used.
- Components of SIOs defined in this Recommendation | International Standard should be used in the definition of the new SIO if they satisfy part of the new requirement.

Specifications of the SIOs that have been developed to support access control are included in the following subclauses. A complete ASN.1 definition for the Security Information Objects discussed in these subclauses is included as a module in Annex A. This module is identified as follows:

```
id-SIOsAccessControl-MODULE OBJECT IDENTIFIER  ::=  {
        joint-iso-itu-t sios(24) specification(0) modules(0) accessControl(0)}
```

6.1 Confidentiality Label

6.1.1 Introduction

Organizations typically have one or more security policies that provide for the compartmentalization of data into groupings that are to be protected and handled in the same way. The security policy defines the protection to be applied to each compartment.

The aspects of security expressed by a security policy, indicated in a security label, include the following:

- the level of protection to be given to data stored on a system;
- who is authorized to access data, processes or resources;
- security markings required to be shown on any display or print of the material;
- routing and enciphering requirements for data transmitted between systems;
- requirements for protection against unauthorized copying;
- methods for storage of data;
- enciphering algorithms to be used;
- methods of authenticating entities;
- whether operations on the object are to be audited;
- whether preventing repudiation of receipt of an object by recipients is required;
- whether, and whose, digital signatures are required to authenticate the data.

When data is held on an Information Technology (IT) system, or when it is transmitted electronically between systems, the data are labelled to indicate the security compartment to which the data belongs and thus how the data is to be handled for security. The label may be separately identifiable from the protected information but is logically bound to it. The integrity of the labels, and the integrity of their binding to the information, must be assured. This allows IT systems and networks to make security-relevant decisions, such as access control and routing, without the need to access the information that is being protected. The security label may be associated with each data object in an IT system, such as documents, electronic mail messages, display windows, database entries, directory entries and electronic forms. The labels are intended for use when objects are stored, moved around (particularly between systems), and when they are being handled by applications that act on labels, including applications that create new objects from existing ones.

When labelled data is to be passed between different security domains, the domains should agree on a security policy to be applied to that data. If the labels specified by the policy applied within a domain differ from the labels specified by the policy for shared data, then the policy for the shared data shall specify how to translate between the two sets of labels.

Labels alone are not sufficient to ensure the security of information. The security policy that applies to the information needs to be enforced by each organization while the labelled information is within the scope of their control. All the organizations, individuals and IT systems that process an item of information are presumed to know the security policy for that information. Organizations that exchange information need to establish trust in one another to be satisfied that information will be handled according to agreed security policies. This trust is usually established through a formal agreement.

6.1.2 ASN.1 Specification of the Label

The confidentiality label is identified as follows:

```
id-ConfidentialityLabel OBJECT IDENTIFIER  ::= {
        joint-iso-itu-t sios(24) specification(0) securityLabels(1) confidentiality(0)}

ConfidentialityLabel  ::=  SET {
        security-policy-identifier         SecurityPolicyIdentifier OPTIONAL,
        security-classification            INTEGER(0..MAX) OPTIONAL,
        privacy-mark                       PrivacyMark OPTIONAL,
        security-categories                SecurityCategories OPTIONAL }
(ALL EXCEPT({-- none; at least one component shall be present --}))

SecurityPolicyIdentifier  ::=  OBJECT IDENTIFIER

PrivacyMark  ::=  CHOICE {
  pString                    PrintableString (SIZE(1..ub-privacy-mark-length)),
  utf8String                 UTF8String (SIZE(1..ub-privacy-mark-length))
}

ub-privacy-mark-length INTEGER  ::=  128  -- as defined in ITU-T Rec. X.411 | ISO/IEC 10021-4

SecurityCategories  ::=  SET SIZE (1..MAX) OF SecurityCategory

SecurityCategory  ::=  SEQUENCE {
     type [0]   SECURITY-CATEGORY.&id ({SecurityCategoriesTable}),
     value      [1]   SECURITY-CATEGORY.&Type ({SecurityCategoriesTable} {@type})
}
```

```
SECURITY-CATEGORY ::= TYPE-IDENTIFIER

SecurityCategoriesTable SECURITY-CATEGORY ::= {...}
```

An example of the expansion of the TYPE-IDENTIFIER information object class is provided in Annex B.

6.1.3 Binding Methods for Confidentiality Labels

6.1.3.1 Binding Method 1

A copy of the data (D) and a copy of the security label (L) are stored together, as a data record, inside the secure boundary of the system. It is assumed that the system is capable of protecting the integrity of the security label and the integrity, as well as possibly the secrecy, of the data. The protection provided by the system must be such that an unauthorized user or application is not capable of altering the data or its associated security label. With this binding method, no cryptographic function is needed to bind the data and the security label.

6.1.3.2 Binding Method 2

A non-secret digital signature (S) is calculated on D and L using a digital signature algorithm (SigAlg) and the private key (X) of a public key algorithm. That is,

S = SigAlg(X,f(D),L)

The digital signature is stored together with D and L in a data record. The generated digital signature binds L to D. In this definition, f is a public function such that f(D) does not reveal information about D.

With this binding method, L and S need not be stored inside the secure boundary of the system. If a cryptographic service is invoked with an incorrect value of L, D or S, the inconsistency is detected. This is accomplished using the public key of the public key algorithm as a verification key to verify the signature.

6.1.3.3 Binding Method 3

A non-secret message authentication code (MAC) is calculated on D and L using a MAC-generation mode of an encipherment algorithm (MacAlg) and a secret MAC algorithm key (K-MAC). That is,

MAC = MacAlg(K-MAC,f(D),L)

The MAC is stored together with D and L in a data record. The generated MAC binds L to D. In this definition, f is a public function such that f(D) does not reveal information about D.

With this binding method, L and MAC need not be stored inside the secure boundary of the system. If a cryptographic service is invoked with an incorrect value of L, D or MAC, the inconsistency is detected. This is accomplished by calculating a MAC-of-reference using the provided values of L and D and a copy of K-MAC, and comparing the result against the provided MAC.

6.2 Security Policy Information File

6.2.1 Introduction

A security policy in its simplest form is a set of criteria for the provision of security services. With regard to access control, security policy is a subset of a higher system-level security policy that defines the means for enforcing access control policies between initiators and targets. The access control mechanisms must:

- allow communication where a specific policy permits; and
- deny communication where a specific policy does not explicitly permit.

A security policy is the basis for the decisions made by the access control mechanisms. Domain-specific security policy information is conveyed via the Security Policy Information File.

The Security Policy Information File contains a sequence of the following:

- **versionInformation** – indicates the version of the ASN.1 syntax and associated semantics of the Security Policy Information File specification.
- **updateInformation** – indicates the currency of the security policy information file data.
- **securityPolicyIdData** – identifies the security policy to which the Security Policy Information File applies.
- **privilegeId** – indicates the OID that identifies the syntax included in the clearance attribute security category of relying certificates used in conjunction with the Security Policy Information File. The syntax indicated by **privilegeId** must be consistent with that indicated by **rbacId**.

- **securityClassifications** – maps the classification of the security label to a classification in the clearance attribute, and also provides equivalency mappings.
- **rbacId** – rule based access control object identifier which identifies the syntax included in the **securityLabel** security category that is used in conjunction with the Security Policy Information File. The syntax indicated by **rbacId** must be consistent with that indicated by **privilegeId**.
- **securityCategories** – maps the security categories of the security label to the security categories in the clearance attribute, and also provides equivalency mappings.
- **equivalentPolicies** – consolidates all equivalent policies in the SPIF.
- **defaultSecurityPolicyIdData** – identifies the security policy which will apply if data is received without a security label.
- **extensions** – provides a mechanism to include additional capabilities as future requirements are identified.

The Security Policy Information File is a signed object to protect it from unauthorized changes.

6.2.2 ASN.1 Specification of the Security Policy Information File

The Security Policy Information File is defined by the following syntax:

```
SecurityPolicyInformationFile  ::=  SIGNED {EncodedSPIF}

EncodedSPIF  ::=  TYPE-IDENTIFIER.&Type( SPIF )

SPIF  ::=  SEQUENCE    {
        versionInformation                          VersionInformationData DEFAULT v1,
        updateInformation                           UpdateInformationData,
        securityPolicyIdData                        ObjectIdData,
        privilegeId                                 OBJECT IDENTIFIER,
        rbacId                                      OBJECT IDENTIFIER,
        securityClassifications          [0]        SEQUENCE OF SecurityClassification OPTIONAL,
        securityCategories               [1]        SEQUENCE OF SecurityCategory OPTIONAL,
        equivalentPolicies               [2]        SEQUENCE OF EquivalentPolicy OPTIONAL,
        defaultSecurityPolicyIdData      [3]        ObjectIdData OPTIONAL,
        extensions                       [4]        Extensions OPTIONAL }
```

6.2.2.1 Version Information

The **versionInformation** field indicates the ASN.1 syntax version as well as the associated semantics.

```
VersionInformationData  ::=  INTEGER { v1(0) } (0..MAX)
```

6.2.2.2 Update Information

The **updateInformationData** is a sequence of information pertaining to the specific version of the SPIF data. The **sPIFVersionNumber** differentiates between different versions of the SPIF information for the security policy identified in **securityPolicyIdData** in the SPIF. The **creationDate** indicates when the SPIF was generated. The **originatorDistinguishedName** identifies the signer of the SPIF. The **keyIdentifier** identifies the key used to sign the SPIF.

```
UpdateInformationData  ::=  SEQUENCE    {
        sPIFVersionNumber                           INTEGER (0..MAX),
        creationDate                                GeneralizedTime,
        originatorDistinguishedName                 Name,
        keyIdentifier                               OCTET STRING OPTIONAL }
```

6.2.2.3 Security Policy ID Data

The **securityPolicyIdData** identifies the security policy to which the SPIF applies. The **securityPolicyIdData** is defined as **ObjectIdData**, where **ObjectIdData** is a sequence of **objectId** and **objectIdName**. An **objectId** is the OID assigned to a specific object, while the **objectIdName** is a string identifying a specific object.

```
ObjectIdData  ::=  SEQUENCE  {
        objectId                OBJECT IDENTIFIER,
        objectIdName            ObjectIdName }
ObjectIdName  ::=  DirectoryString {ubObjectIdNameLength}
```

6.2.2.4 Privilege Identifier

The **privilegeId** object identifier identifies the syntax that is included in the clearance attribute security category of relying certificates used in conjunction with the SPIF.

6.2.2.5 RBAC Identifier

The **rbacId** object identifier identifies the syntax that is included in the **securityLabel** security categories used in conjunction with the SPIF. The syntax indicated by **rbacId** must be consistent with that indicated by **privilegeId**.

6.2.2.6 Security Classifications

A **SecurityClassification** SEQUENCE is present in the SPIF for each security classification value defined for the security policy identified in **securityPolicyIdData**. This is an OPTIONAL element.

The **labelAndCertValue** represents the value assigned to this classification in the security label and the integer value representing the bit location of this security classification in the clearance attribute **classList BIT STRING**.

The **classificationName** is a string identifying this classification used by the application to determine the text to be displayed to the user when selecting or viewing the classification value in a Security Label.

The **equivalentClassifications** is a sequence of classification values (defined in security policies other than **securityPolicyIdData**) that are equivalent to the **SecurityClassification labelAndCertValue**.

The **hierarchyValue** indicates the relative position of the **SecurityClassification labelAndCertValue** in the hierarchy of security classifications in the security policy indicated by the **securitypolicyIdData**. The **hierarchyValue** must be unique within the security policy.

The **markingData** identifies the marking information attached to the data object. **markingData** is composed of strings and marking codes which identify where the string is physically displayed. If the **markingPhrase** is absent, then the **markingCode** applies to the **SecurityClassification classificationName**.

When a security category or security classification is selected for inclusion in the security label, the associated SPIF **requiredCategory** field, if present, indicates the security categories that must also be included in the security label in conjunction with the selected value. If the **requiredCategory** field is not present, then the selected value has no dependencies on any security categories.

If **OptionalCategoryGroup** operation is **onlyOne**, then one (and only one) of the security categories included in **categoryGroup** must be included in the security label. If **OptionalCategoryGroup** operation is **oneOrMore**, then one or more of the security categories included in **categoryGroup** must be included in the security label. If **OptionalCategoryGroup** operation is all, then all of the security categories included in **categoryGroup** must be included in the security label. The user must select each value. If multiple **OptionalCategoryGroups** are present in **requiredCategories**, then the requirement expressed by all of the **OptionalCategoryGroups** must be satisfied. **categoryGroup** is a sequence of **OptionalCategoryData**. The **optCatDataId** object identifier must specify a syntax for use in the **OptionalCategoryData categorydata** field that is consistent with those specified by the **rbacId, privilegeId** and SPIF **SecurityCategory** type object identifiers.

The **obsolete** component, when set to TRUE, indicates that a formerly valid classification is now obsolete. Such a classification may be associated with old data objects, but it should not be associated with the new ones.

```
SecurityClassification ::= SEQUENCE  {
        labelAndCertValue                       LabelAndCertValue,
        classificationName                      ClassificationName,
        equivalentClassifications    [0]        EquivalentClassifications OPTIONAL,
        hierarchyValue                          INTEGER,
        markingData                  [1]        MarkingDataInfo OPTIONAL,
        requiredCategory             [2]        OptionalCategoryGroups OPTIONAL,
        obsolete                                BOOLEAN DEFAULT FALSE      }

LabelAndCertValue ::= INTEGER (0..MAX)
ClassificationName ::= DirectoryString {  ubClassificationNameLength }
EquivalentClassifications ::= SEQUENCE SIZE(0..MAX) OF EquivalentClassification

EquivalentClassification ::= SEQUENCE {
        securityPolicyId             OBJECT IDENTIFIER,
        labelAndCertValue            LabelAndCertValue,
        applied Applied }

Applied ::= INTEGER {
        encrypt (0),
        decrypt (1),
        both     (2) }
  (encrypt | decrypt | both)

MarkingDataInfo ::= SEQUENCE SIZE(1..MAX) OF MarkingData
```

```
MarkingData ::= SEQUENCE {
        markingPhrase              MarkingPhrase OPTIONAL,
        markingCodes               MarkingCodes OPTIONAL }
  (ALL EXCEPT({-- none; at least one component shall be present --}))

MarkingPhrase ::= DirectoryString { ubMarkingPhraseLength }

MarkingCodes ::= SEQUENCE SIZE(1..MAX) OF MarkingCode

MarkingCode ::= INTEGER {
        pageTop                    (1),
        pageBottom                 (2),
        pageTopBottom              (3),
        documentEnd                (4),
        noNameDisplay              (5),
        noMarkingDisplay           (6),
        unused                     (7),
        documentStart              (8),
        suppressClassName          (9) }

OptionalCategoryGroups ::= SEQUENCE SIZE(1..MAX) OF OptionalCategoryGroup

OptionalCategoryGroup ::= SEQUENCE {
        operation                  Operation,
        categoryGroup              CategoryGroup }

Operation ::= INTEGER {
        onlyOne                    (1),
        oneOrMore                  (2),
        all                        (3)}
(onlyOne | oneOrMore | all)

CategoryGroup ::= SEQUENCE SIZE(1..MAX) OF OptionalCategoryData
OptionalCategoryData ::= SEQUENCE {
        optCatDataId               OC-DATA.&id({CatData}),
        categorydata               OC-DATA.&Type({CatData}{@optCatDataId }) }

OC-DATA ::= TYPE-IDENTIFIER
CatData OC-DATA ::= { ... }
```

6.2.2.7 Security Categories

A **SecurityCategory** SEQUENCE is present in the SPIF for each security category defined for the security policy identified in **securityPolicyIdData**. The **SecurityCategory** syntax is defined in the confidentiality label given in 6.1. The syntax defined for use in the **SecurityCategory** value field that is indicated by the **SecurityCategory** type object identifier must be consistent with the syntaxes indicated by the **privilegeId, rbacId** and **optCatDataID** object identifiers.

6.2.2.8 Equivalent Policies

The **equivalentPolicies** is a list of all security policies for which values have been included in the SPIF as equivalent values. The **securityPolicyId** is an Object Identifier that identifies the equivalent security policy. The **securityPolicyName** is an optional directory string identifying the name of the equivalent security policy.

```
EquivalentPolicy ::= SEQUENCE {
        securityPolicyId           OBJECT IDENTIFIER,
        securityPolicyName         SecurityPolicyName  OPTIONAL}

SecurityPolicyName ::= DirectoryString {ubObjectIdNameLength}
```

6.2.2.9 Default Security Policy Identifier

The value for **defaultSecurityPolicyIdData** supports interoperability with applications that do not support access control. This object identifier is referenced when no security label is used.

Note that the default security policy will be mapped to a single classification level. When a security classification value is mapped to the Default Security Policy, then the SPIF **SecurityClassification** SEQUENCE for the designated value will include an **equivalentClassification** SEQUENCE in which the **policyId** is set to the Default Security Policy object identifier.

6.2.2.10 Extensions

The **extension** field is a sequence of information that allows for future expansion of the SPIF as additional requirements are identified while maintaining interoperability with previous SPIF implementations. It contains the **extnId, critical**, and **extnValue** components. The syntax is imported from ITU-T Rec. X.509 | ISO/IEC 9594-8.

An extension may be designated as critical or non-critical. A SPIF-using system MUST reject the SPIF if it encounters a critical extension it does not recognize; however, a non-critical extension may be ignored if it is not recognized. Caution should be exercised in adopting any critical extensions that might prevent use in a general context.

```
Extensions  ::=  SEQUENCE OF Extension

Extension  ::=  SEQUENCE {
        extnId                    EXTENSION.&id ({ExtensionSet}),
        critical                  BOOLEAN DEFAULT FALSE,
        extnValue                 OCTET STRING
          -- contains a DER encoding of a value of type &ExtnType
          -- for the extension object identified by extnId -- }

ExtensionSet EXTENSION  ::=  { ... }

EXTENSION  ::=  CLASS {
        &id                       OBJECT IDENTIFIER UNIQUE,
        &ExtnType }
WITH SYNTAX { SYNTAX &ExtnType IDENTIFIED BY &id }
```

6.3 Clearance Attribute

6.3.1 Introduction

The Clearance attribute is used to define the authorizations granted a specific user or application entity. The authorizations granted to a user or application entity may be a subset of the organization's total security policy (or policies). Alternately, authorizations may encompass the security policy in its entirety.

The clearance attribute contains three components: **policyId**, **classList**, and, optionally, **securityCategory** as illustrated in Figure 1 below.

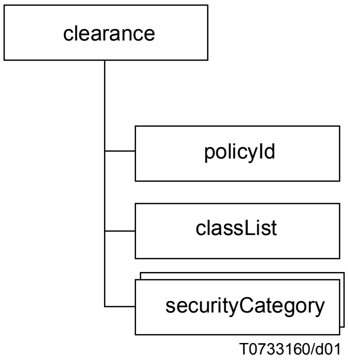
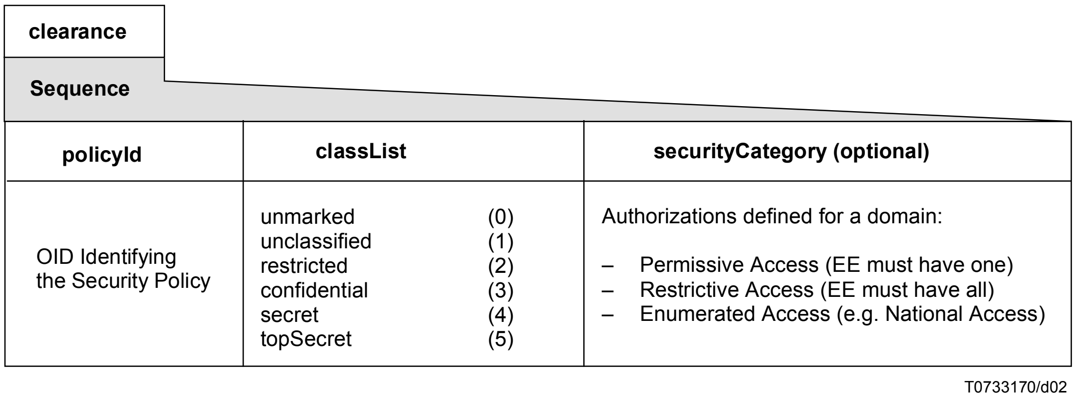


Figure 1 – Clearance Attributes Structure

The **policyId** OID identifies which optional components must be present. The **classList** component defines the user's granted and hierarchical clearances as indicated by **ClassList**, which is defined by ITU-T Rec. X.501 | ISO/IEC 9594-2. Other non-hierarchical class lists could be defined elsewhere for inclusion in other SIOs or addressed in security categories. The **securityCategory** component identifies any number of restrictive and permissive bit mapped security categories as well as restrictive and permissive enumerated security categories assigned to the user. This structure is illustrated in Figure 2.



policyId	classList	securityCategory (optional)
OID Identifying the Security Policy	unmarked (0) unclassified (1) restricted (2) confidential (3) secret (4) topSecret (5)	Authorizations defined for a domain: – Permissive Access (EE must have one) – Restrictive Access (EE must have all) – Enumerated Access (e.g. National Access)

Figure 2 – Clearance Attribute Fields

6.3.2 Definition of clearance attribute

The clearance attribute is defined:

```
clearance  ATTRIBUTE  ::=  { WITH SYNTAX    Clearance
          ID                               id-at-clearance }

id-at-clearance OBJECT IDENTIFIER  ::=  {
  joint-iso-itu-t (2) ds (5) attributeType (4) clearance (55) }

Clearance  ::=  SEQUENCE {
          policyId                         OBJECT IDENTIFIER,
          classList                        ClassList DEFAULT {unclassified},
          securityCategories               SecurityCategories  OPTIONAL}

ClassList  ::=  BIT STRING {
          unmarked                         (0),
          unclassified                     (1),
          restricted                       (2),
          confidential                     (3),
          secret                           (4),
          topSecret                        (5)    }

SecurityCategories  ::=  SET SIZE(1..MAX) OF SecurityCategory
    --  SecurityCategory is defined in the confidentiality label given in subclause 6.1.2
```

7 Security Information Object Interaction

7.1 SIO Class Structure Comparison

Figure 3 shows the structures of the confidentiality label, the ITU-T Rec. X.501 | ISO/IEC 9594-2 clearance attribute and the SPIF for comparison. Equivalent components in these structures may be examined in application software to achieve specific functionality. Achievement of access control functionality using these three structures is discussed in 7.2.

7.2 Security Information Object Interaction for Access Control

Access control includes the concept of conveying authorizations for initiators or users through the use of a clearance attribute and assigning sensitivities to target objects by means of a security label. The SPIF is used to interpret these authorization and sensitivity parameters. Application software uses the SPIF to apply sensitivities to targets, read sensitivities from labels, read and assert authorizations in certificates, and determine legitimate policy mappings across security policy domains.

The mechanisms used to convey the authorizations and sensitivities are classifications and categories. Classifications and categories are asserted in a clearance attribute embedded in a user's certificate thereby conveying the authorizations of that user. Classifications and categories are also asserted in an object's security label thereby conveying the sensitivities of that object. Access to an object is permitted when authorizations conveyed in the clearance attribute of a user are sufficient when compared to the sensitivities conveyed in the security label of the target object. Figure 4 illustrates the interactions among the SIOs defined herein to achieve access control in a data storage environment. The authorizations in the data owner's clearance attribute, contained in the certificate associated with the data owner, limit what authorizations from the SPIF the owner can assert in the label for the target data. The label is bound with the data and placed in storage. In accessing data in the storage device, the user's clearance attribute, contained in the certificate associated with the user, is compared with the label bound with the target data in the Access Control Decision function. If permissive sensitivities exist in the security label, they are checked to ensure that at least one of the sensitivities present in each permissive tag in the security label is also authorized in the certificate (permissive authorization(s)) allowing access to the target data object through the Access Control Enforcement function. A similar access control scenario for a messaging environment is shown in Figure 5.

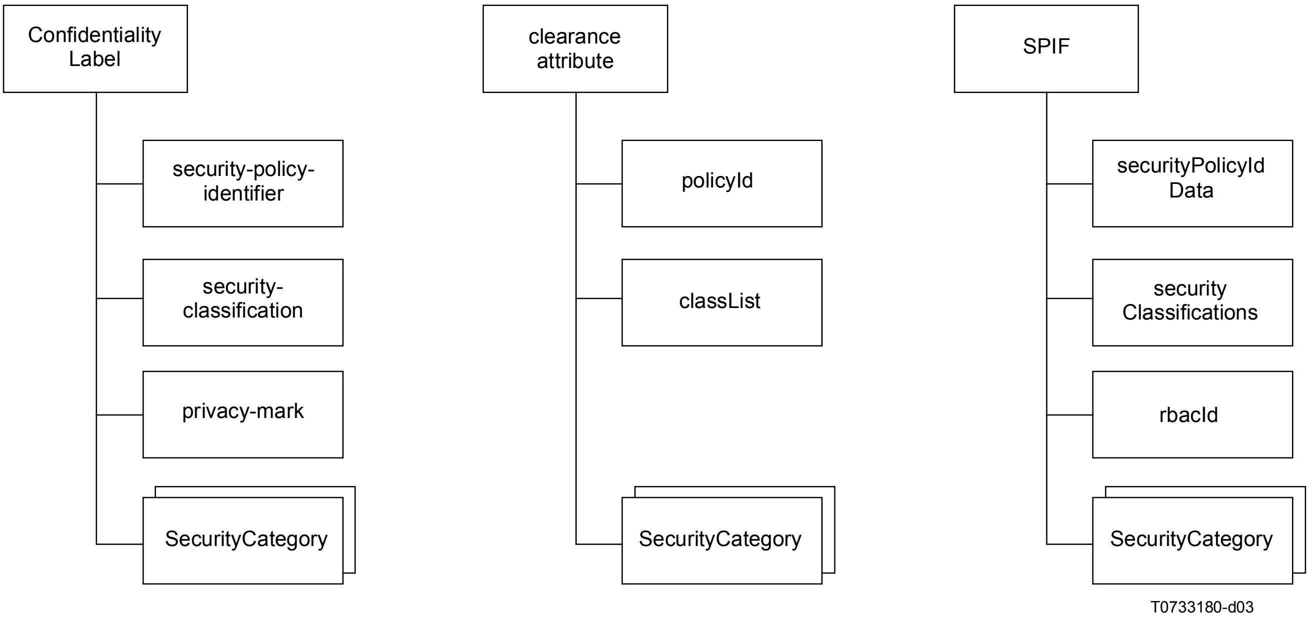


Figure 3 – Equivalent Object Class Comparison

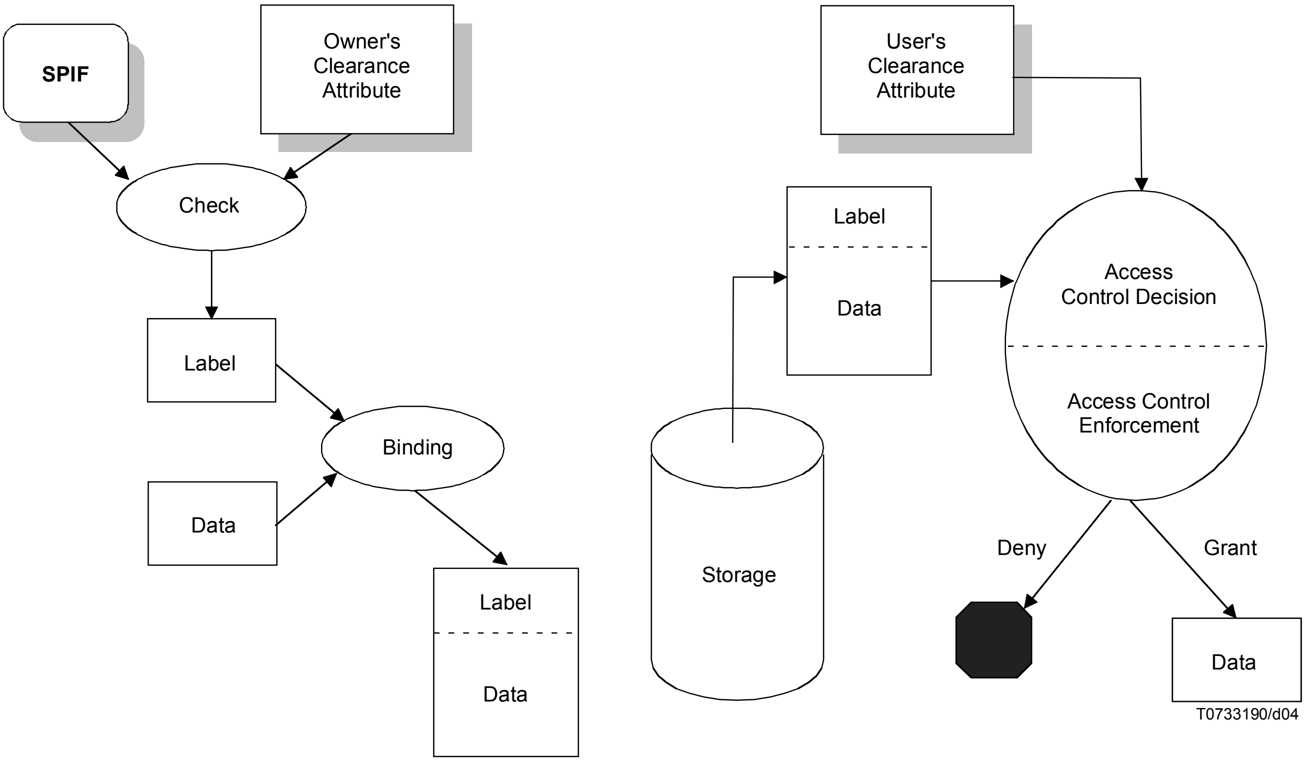


Figure 4 – Data Storage Access Control

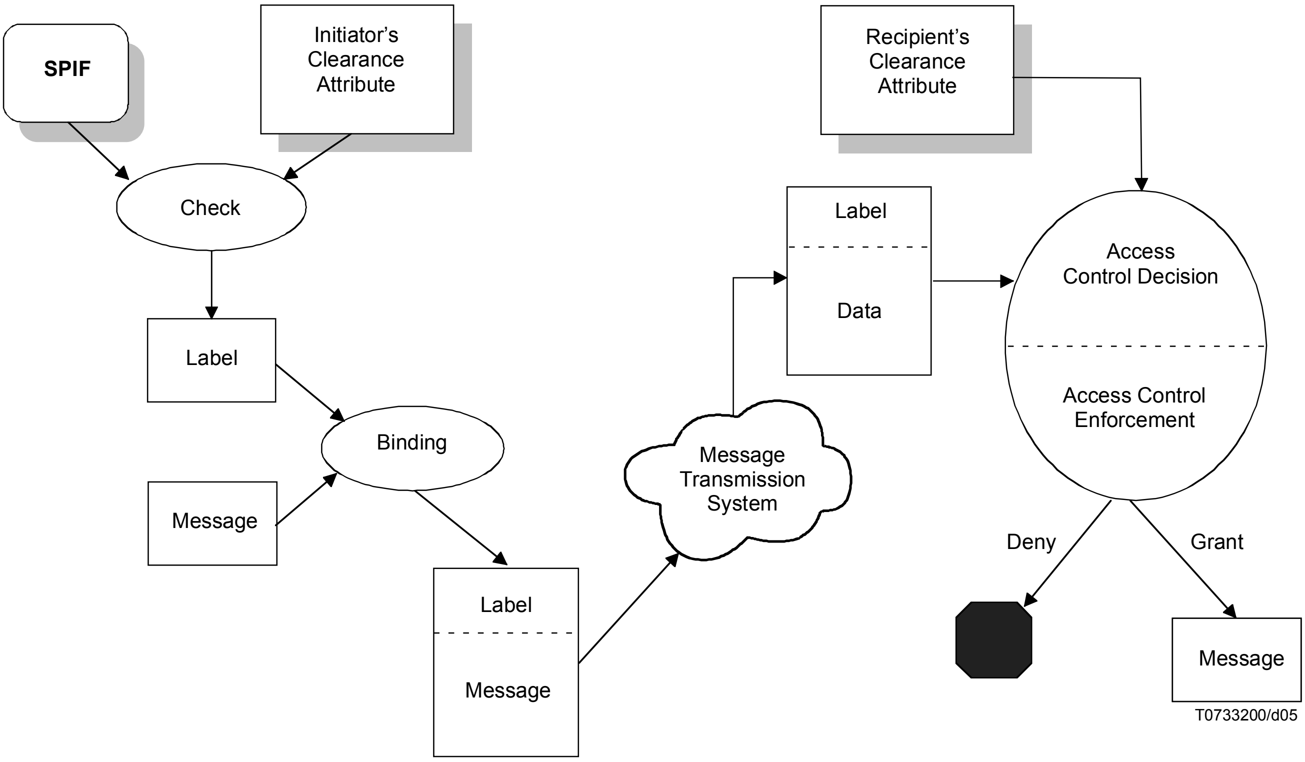

SPIF
Initiator's Clearance Attribute
Check
Label
Binding
Message
Label
Message
Message Transmission System
Label
Data
Recipient's Clearance Attribute
Access Control Decision
Access Control Enforcement
Deny
Grant
Message
T0733200/d05

Figure 5 – Messaging Scenario Access Control

Annex A

Security Information Objects for Access Control in ASN.1

(This annex forms an integral part of this Recommendation | International Standard)

This annex includes all of the ASN.1 type, value, and information object class definitions contained in this Recommendation | International Standard, in the form of an ASN.1 module.

```
SIOsAccessControl-MODULE  {
   joint-iso-itu-t sios(24) specification(0) modules(0) accessControl(0)
}
       DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS All; --

IMPORTS

   id-at-clearance
      FROM EnhancedSecurity        -- ITU-T Rec. X.501 | ISO/IEC 9594-2 --

   ATTRIBUTE, Name
      FROM InformationFramework    -- ITU-T Rec. X.501 | ISO/IEC 9594-2 --

   Extensions
      FROM CertificateExtensions   -- ITU-T Rec. X.509 | ISO/IEC 9594-8 --

   DirectoryString {}
      FROM SelectedAttributeTypes; -- ITU-T Rec. X.520 | ISO/IEC 9594-6 --

id-ConfidentialityLabel OBJECT IDENTIFIER  ::=  {joint-iso-itu-t sios(24) specification(0)
securityLabels(1) confidentiality(0)}

ConfidentialityLabel  ::=  SET {
   security-policy-identifier  SecurityPolicyIdentifier  OPTIONAL,
   security-classification     INTEGER(0..MAX)  OPTIONAL,
   privacy-mark                PrivacyMark  OPTIONAL,
   security-categories         SecurityCategories  OPTIONAL
}
 (ALL EXCEPT({-- none; at least one component shall be present --}))

SecurityPolicyIdentifier ::= OBJECT IDENTIFIER
```

```
PrivacyMark ::= CHOICE {
   pString     PrintableString (SIZE(1..ub-privacy-mark-length)),
   utf8String  UTF8String (SIZE(1..ub-privacy-mark-length))
}

ub-privacy-mark-length INTEGER ::= 128  -- as defined in X.411

SecurityCategories ::= SET SIZE(1..MAX) OF SecurityCategory

SecurityCategory ::= SEQUENCE {
   type   [0] SECURITY-CATEGORY.&id({SecurityCategoriesTable}),
   value  [1] EXPLICIT SECURITY-CATEGORY.&Type(
                             {SecurityCategoriesTable}{@type})
}

SECURITY-CATEGORY ::= TYPE-IDENTIFIER

SecurityCategoriesTable SECURITY-CATEGORY ::= {
   ...  -- objects defined as needed --
}

SecurityPolicyInformationFile ::= SIGNED { EncodedSPIF }

-- Type EncodedSPIF is an open type constrained to be a value
-- of type SPIF. This open type representation is an opaque
-- string of hexadecimal characters suitable for signature
-- and signature verification operations.

EncodedSPIF ::= TYPE-IDENTIFIER.&Type( SPIF )

SPIF ::= SEQUENCE {
   versionInformation           VersionInformationData  DEFAULT v1,
   updateInformation            UpdateInformationData,
   securityPolicyIdData         ObjectIdData,
   privilegeId                  OBJECT IDENTIFIER,
   rbacId                       OBJECT IDENTIFIER,
   securityClassifications      [0] SecurityClassifications  OPTIONAL,
   securityCategories           [1] SPIF-SecurityCategories  OPTIONAL,
   equivalentPolicies           [2] EquivalentPolicies  OPTIONAL,
   defaultSecurityPolicyIdData  [3] ObjectIdData  OPTIONAL,
   extensions                   [4] Extensions  OPTIONAL
}
```

```
VersionInformationData ::= INTEGER { v1(0) } (0..MAX)

UpdateInformationData ::= SEQUENCE {
   sPIFVersionNumber            SPIFVersionNumber,
   creationDate                 GeneralizedTime,
   originatorDistinguishedName  Name,
   keyIdentifier                OCTET STRING  OPTIONAL
}

SPIFVersionNumber ::=  INTEGER (0..MAX)

ObjectIdData ::= SEQUENCE {
   objectId      OBJECT IDENTIFIER,
   objectIdName  ObjectIdName
}

ObjectIdName ::= DirectoryString { ubObjectIdNameLength }

SecurityClassifications ::=
      SEQUENCE SIZE(0..MAX) OF SecurityClassification

SPIF-SecurityCategories ::=
      SEQUENCE SIZE(0..MAX) OF SecurityCategory

EquivalentPolicies ::=
      SEQUENCE SIZE(0..MAX) OF EquivalentPolicy

SecurityClassification ::= SEQUENCE {
   labelAndCertValue          LabelAndCertValue,
   classificationName         ClassificationName,
   equivalentClassifications  [0] EquivalentClassifications  OPTIONAL,
   hierarchyValue             INTEGER,
   markingData                [1] MarkingDataInfo  OPTIONAL,
   requiredCategory           [2] OptionalCategoryGroups  OPTIONAL,
   obsolete                   BOOLEAN  DEFAULT FALSE
}

LabelAndCertValue ::= INTEGER(0..MAX)

ClassificationName ::= DirectoryString { ubClassificationNameLength }

EquivalentClassifications ::=
     SEQUENCE SIZE(0..MAX) OF EquivalentClassification
```

```
EquivalentClassification ::= SEQUENCE {
   securityPolicyId   OBJECT IDENTIFIER,
   labelAndCertValue  LabelAndCertValue,
   applied            Applied
}

Applied ::= INTEGER {
   encrypt (0),
   decrypt (1),
   both    (2)
}
 (encrypt | decrypt | both)

MarkingDataInfo ::= SEQUENCE SIZE(1..MAX) OF MarkingData

MarkingData ::= SEQUENCE {
   markingPhrase  MarkingPhrase  OPTIONAL,
   markingCodes   MarkingCodes  OPTIONAL
}
 (ALL EXCEPT({-- none; at least one component shall be present --}))

MarkingPhrase ::= DirectoryString { ubMarkingPhraseLength }

MarkingCodes ::= SEQUENCE SIZE(1..MAX) OF MarkingCode

MarkingCode ::= INTEGER {
   pageTop           (1),
   pageBottom        (2),
   pageTopBottom     (3),
   documentEnd       (4),
   noNameDisplay     (5),
   noMarkingDisplay  (6),
   unused            (7),
   documentStart     (8),
   suppressClassName (9)
}

OptionalCategoryGroups ::=
      SEQUENCE SIZE(1..MAX) OF OptionalCategoryGroup
```

```
OptionalCategoryGroup ::= SEQUENCE {
   operation      Operation,
   categoryGroup  CategoryGroup
}

Operation ::= INTEGER {
   onlyOne   (1),
   oneOrMore (2),
   all       (3)
}
 (onlyOne | oneOrMore | all)

CategoryGroup ::= SEQUENCE SIZE(1..MAX) OF OptionalCategoryData

OptionalCategoryData ::= SEQUENCE {
   optCatDataId  OC-DATA.&id({CatData}),
   categorydata  OC-DATA.&Type({CatData}{@optCatDataId })
}

OC-DATA ::= TYPE-IDENTIFIER

CatData OC-DATA ::= {
   ...  -- defined as needed --
}

EquivalentPolicy ::= SEQUENCE {
   securityPolicyId    OBJECT IDENTIFIER,
   securityPolicyName  SecurityPolicyName  OPTIONAL
}

SecurityPolicyName ::= DirectoryString { ubObjectIdNameLength }

clearance ATTRIBUTE ::= {
   WITH SYNTAX  Clearance
   ID           id-at-clearance
}

Clearance ::= SEQUENCE {   -- Automatic tags applied
   policyId            [0] OBJECT IDENTIFIER,
   classList           [1] ClassList  DEFAULT { unclassified },
   securityCategories  [2] SecurityCategories  OPTIONAL
}
```

```
ClassList ::= BIT STRING {
   unmarked     (0),
   unclassified (1),
   restricted   (2),
   confidential (3),
   secret       (4),
   topSecret    (5)
}

-- upper bound values

ubObjectIdNameLength        INTEGER ::= 256
ubClassificationNameLength  INTEGER ::= 256
ubMarkingPhraseLength       INTEGER ::= 256

-- information object classes --

ALGORITHM ::= CLASS {
   &id    OBJECT IDENTIFIER  UNIQUE,
   &Type  OPTIONAL
}
  WITH SYNTAX { OID &id [PARMS &Type] }

-- parameterized types --

SIGNED { ToBeSigned } ::= SEQUENCE {
   toBeSigned  ToBeSigned,
   algorithm   AlgorithmIdentifier{{SignatureAlgorithms}},
   signature   BIT STRING
}

AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
   algorithm   ALGORITHM.&id({IOSet}),
   parameters  ALGORITHM.&Type({IOSet}{@algorithm}) OPTIONAL
}

SignatureAlgorithms ALGORITHM ::= {
  ... -- defined as needed --
}

END  -- SecurityInformationObjects --
```

Annex B

Expansion of the **SECURITY-CATEGORY** Syntax

(This annex does not form an integral part of this Recommendation | International Standard)

The information object class **SECURITY-CATEGORY** is defined as the built-in class **TYPE-IDENTIFIER**.

```
SECURITY-CATEGORY  ::=  TYPE-IDENTIFIER
```

This useful information object class is specified in Annex A of ITU-T Rec. X.681 | ISO/IEC 8824-2 as:

```
TYPE-IDENTIFIER ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type
}
     WITH SYNTAX { &Type IDENTIFIED BY &id }
```

The **SECURITY-CATEGORY** class has two fields named **&id** and **&Type**. The **&id** field is defined to be a value of type **OBJECT IDENTIFER**, and the **&Type** field is an open type. An open type may be any ASN.1 type.

When objects of this class are used as members of an information object set, the definition of the **&id** field requires that each object in the set must contain a unique object identifier value. The class definition also includes a WITH SYNTAX statement that specifies a notation that can be used to define information objects of class **SECURITY-CATEGORY**.

SecurityCategoriesTable is an information object set of class **SECURITY-CATEGORY**. It is defined as:

```
SecurityCategoriesTable SECURITY-CATEGORY ::= {
     ...  -- objects defined as needed --
}
```

The **SecurityCategoriesTable** set contains an extension marker, "...", but no information objects. Objects of class **SECURITY-CATEGORY** can be specified individually using the **WITH SYNTAX** notation provided in the class definition. The following example objects show that any ASN.1 type, whether simple or complex, can be used to create an information object.

```
-- Type 1 - restrictive attributes

restrictiveBitMap SECURITY-CATEGORY ::= {
     AttributeFlags IDENTIFIED BY id-restrictiveBitMap
}

AttributeFlags ::= BIT STRING

-- Type 2 - hierarchical attributes

enumeratedAttributes SECURITY-CATEGORY ::= {
     AttributeList IDENTIFIED BY id-enumeratedAttributes
}

AttributeList ::= SET SIZE(1..MAX) OF LabelAttribute

-- Type 5 - all attributes in the range(s)

rangeSet SECURITY-CATEGORY ::= {
     RangeList IDENTIFIED BY id-rangeSet
}

RangeList ::= SET SIZE(1..MAX) OF LabelAttributeRange

-- Type 6 - release attributes

permissiveBitMap SECURITY-CATEGORY ::= {
     PermissiveBitMap IDENTIFIED BY id-permissiveBitMap

}

PermissiveBitMap ::= BIT STRING

-- Type 7 – for markings with no formal access control --

freeFormField SECURITY-CATEGORY ::= {
     FreeFormField IDENTIFIED BY id-freeFormField
}
```

```
  FreeFormField ::= SEQUENCE {
     name   SECURITY-CATEGORY.&id({Fields}),
     field  SECURITY-CATEGORY.&Type({Fields}{@name})
  }

  Fields SECURITY-CATEGORY ::= {
     ...  -- defined as needed --
  }
```

Here the **&Type** fields of the objects contain ASN.1 types named **AttributeFlags**, **AttributeList**, **RangeList**, **PermissiveBitMap** and **FreeFormField**. The **&id** fields contain unique object identifier values named **id-restrictiveBitMap**, **id-enumeratedAttributes**, **id-rangeSet**, **id-permissiveBitMap** and **id-freeFormField**.

These objects can be added to an implementation version of the **SecurityCategoriesTable** by object name to form a security category set from the union of the objects:

```
SecurityCategoriesTable SECURITY-CATEGORY ::= {
       restrictiveBitMap |
       enumeratedAttributes  |
       rangeSet       |
       permissiveBitMap
       freeFormField,
       ...  -- expect other objects --
}
```

Alternatively, the object definitions can be added directly to the **SecurityCategoriesTable** information object set:

```
SecurityCategoriesTable SECURITY-CATEGORY ::= {
       --
       --    &Type                               &id
       --
       { AttributeFlags   IDENTIFIED BY id-restrictiveBitMap    } |
       { AttributeList    IDENTIFIED BY id-enumeratedAttributes } |
       { RangeList        IDENTIFIED BY id-rangeSet             } |
       { PermissiveBitMap IDENTIFIED BY id-permissiveBitMap     } |
       { FreeFormField    IDENTIFIED BY id-freeFormField       },
       ...  -- expect other objects --
  }
```

This view of the security categories set shows a table of four rows, each having two columns, one column for **&id** and another for **&Type**.

Type **SecurityCategory** is defined as a sequence of two components named type and value.

```
SecurityCategory ::= SEQUENCE {
       type   [0] SECURITY-CATEGORY.&id({SecurityCategoriesTable}),
       value  [1] EXPLICIT SECURITY-CATEGORY.&Type(
                              {SecurityCategoriesTable}{@type})
}
```

Each of these components is specified in terms of the **&id** and **&Type** fields of class **SECURITY-CATEGORY**. The **type** component is specified in terms of the **&id** field and must be a value of type **OBJECT IDENTIFIER**. The value component is specified by the **&**Type field and may be a value of any ASN.1 type.

The information object set **SecurityCategoriesTable** is used to form a table constraint on the valid values of the **type** and **value** components of **SecurityCategory**. The table constraint has two columns, one for each field of the **SECURITY-CATEGORY** class.

The unique object identifier value specified by the &id field of the type component selects a row in the table. The **@type** notation selects the **&Type** column associated with the **&id** value of the selected row. The extension marker in the **SecurityCategoriesTable** set indicates that an application should expect objects other than those explicitly specified in the set.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks and open system communications**
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Geneva, 2001