-- 12 ASN.1 DEFINITIONS

-- 12.1 DATA TYPE DEFINITION

```
IN-CS3-SCF-SDF-datatypes
        {ccitt recommendation q 1238 modules(1) in-cs3-scf-sdf-datatypes(14) version1(0) }

DEFINITIONS ::=

BEGIN

IMPORTS

ds-UsefulDefinitions, scf-sdf-classes,  ssf-scf-datatypes, ssf-scf-classes, scf-scf-datatypes,
id-soa-methodRuleUse, id-at-securityFacilityId, id-at-secretKey, id-at-identifierList, id-at-bindLevelIfOK,
id-at-lockSession, id-at-failureCounter, id-at-maxAttempts, id-at-currentList, id-at-stockId, id-at-source,
id-at-sizeOfRestocking, id-at-challengeResponse, id-aca-prescriptiveACI, id-aca-entryACI, id-aca-subentryACI,
id-avc-basicService, id-avc-lineIdentity, id-avc-assignment
    FROM IN-CS3-object-identifiers
        {ccitt recommendation q 1238 modules(1) in-cs3-object-identifiers(0) version1(0)}

informationFramework, upperBounds, directoryAbstractService, selectedAttributeTypes, basicAccessControl,
authenticationFramework
    FROM UsefulDefinitions ds-UsefulDefinitions

ATTRIBUTE, OBJECT-CLASS, CONTEXT, AttributeType, AttributeTypeAndValue, objectClass,
aliasedEntryName, SubtreeSpecification, ContextAssertion, DistinguishedName
    FROM InformationFramework informationFramework

ub-tag, ub-schema
    FROM UpperBounds upperBounds

METHOD, SupportedMethods
    FROM IN-CS3-SCF-SDF-classes scf-sdf-classes

Filter
    FROM DirectoryAbstractService  directoryAbstractService

NameAndOptionalUID, directoryStringFirstComponentMatch, DirectoryString{}
    FROM SelectedAttributeTypes selectedAttributeTypes

MaxValueCount, RestrictedValue, AuthenticationLevel, Precedence
    FROM BasicAccessControl basicAccessControl

AlgorithmIdentifier
    FROM AuthenticationFramework authenticationFramework

Digits{}
    FROM    IN-CS3-SSF-SCF-datatypes ssf-scf-datatypes

SCF-SSF-BOUNDS
    FROM    IN-CS3-SSF-SCF-Classes ssf-scf-classes

AgreementID
    FROM IN-CS3-SCF-SCF-datatypes scf-scf-datatypes
;

-- Data types
NPartsMessage{INTEGER : n} ::= SEQUENCE SIZE(2..n) OF BIT STRING

SCFCriteria ::=  SEQUENCE {
```

```
        agreement    [0]    IMPLICIT   AgreementID,
        ...}

SDFCriteria ::=  SEQUENCE {
        object               [0]    IMPLICIT   DistinguishedName,
        ...}

TFCcriteria  ::= CHOICE {
        sdf          [0]    IMPLICIT   SDFCriteria,      -- used if initiating FE is an SDF --
        scf          [1]    IMPLICIT   SCFCriteria,      -- used if initiating FE is an SCF --
        ...}

TwoPartMessage ::= NPartsMessage{2}

-- Enhancement data types for Basic Access Control
-- The following enhancements to the third edition X.500 specification of Access Control Information
-- (ACI) are required to support IN requirements on the SCF/SDF interface.
--  The remaining elements apply as described in the third edition X.500-Series of Recommendations.

ACIItem              ::=          SEQUENCE {
        identificationTag         DirectoryString { ub-tag },
        precedence                Precedence,
        authenticationLevel       AuthenticationLevel,
        itemOrUserFirst           CHOICE {
                itemFirst         [0]    SEQUENCE {
                        protectedItems    ProtectedItems,
                        itemPermissions   SET OF ItemPermission },
                userFirst         [1]    SEQUENCE {
                        userClasses       UserClasses,
                        userPermissions   SET OF UserPermission }}}

GrantsAndDenials             ::=    BIT STRING {
        -- permissions that may be used in conjunction with any component of ProtectedItems
        grantAdd                  (0),
        denyAdd                   (1),
        grantDiscloseOnError      (2),
        denyDiscloseOnError       (3),
        grantRead                 (4),
        denyRead                  (5),
        grantRemove               (6),
        denyRemove                (7),
        -- permissions that may be used only in conjunction with the entry component
        grantBrowse               (8),
        denyBrowse                (9),
        grantExport               (10),
        denyExport                (11),
        grantImport               (12),
        denyImport                (13),
        grantModify               (14),
        denyModify                (15),
        grantRename               (16),
        denyRename                (17),
        grantReturnDN             (18),
        denyReturnDN              (19),
        -- permissions that may be used in conjunction with any component, except entry, of
        -- ProtectedItems
        grantCompare              (20),
        denyCompare               (21),
        grantFilterMatch          (22),
        denyFilterMatch           (23),
        -- permissions that may be used in conjunction with entryMethod component of ProtectedItems
        grantExecuteMethod        (30),
        denyExecuteMethod         (31) }
```

```
-- grantExecuteMethod means that the user can perform the specific Methods for the Entry.
-- NOTE – It is a matter for network operators as to whether the grantExecuteMethod
-- permission bypasses the normal access control mechanisms for Entries and Attributes.
-- denyExecuteMethod means that the user cannot perform the specific Methods for the Entry

ItemPermission                  ::=     SEQUENCE {
        precedence              Precedence OPTIONAL,
                                        -- defaults to precedence in ACIItem --
        userClasses             UserClasses,
        grantsAndDenials        GrantsAndDenials }

MethodIDs ::=       METHOD.&id

ProtectedItems      ::=     SEQUENCE {
        entry                                   [0]     NULL OPTIONAL,
        allUserAttributeTypes                   [1]     NULL OPTIONAL,
        attributeType                           [2]     SET OF AttributeType OPTIONAL,
        allAttributeValues                      [3]     SET OF AttributeType OPTIONAL,
        allUserAttributeTypesAndValues          [4]     NULL OPTIONAL,
        attributeValue                          [5]     SET OF AttributeTypeAndValue OPTIONAL,
        selfValue                               [6]     SET OF AttributeType OPTIONAL,
        rangeOfValues                           [7]     Filter OPTIONAL,
        maxValueCount                           [8]     SET OF MaxValueCount OPTIONAL,
        maxImmSub                               [9]     INTEGER OPTIONAL,
        restrictedBy                            [10]    SET OF RestrictedValue OPTIONAL,
        contexts                                [11]    SET OF ContextAssertion OPTIONAL,
        entryMethods                            [30]    SET OF MethodIDs OPTIONAL}
-- entryMethods identifies the specified Methods for which the level of protection is to be applied.

UserClasses  ::=    SEQUENCE {
        allUsers        [0]     NULL OPTIONAL,
        thisEntry       [1]     NULL OPTIONAL,
        name            [2]     SET OF NameAndOptionalUID OPTIONAL,
        userGroup       [3]     SET OF NameAndOptionalUID OPTIONAL,
                                        -- dn component must be the name of an
                                        -- entry of GroupOfUniqueNames
        subtree         [4]     SET OF SubtreeSpecification OPTIONAL}

UserPermission                  ::=     SEQUENCE {
        precedence              Precedence OPTIONAL,
                                        -- defaults to precedence in ACIItem
        protectedItems          ProtectedItems,
        grantsAndDenials        GrantsAndDenials }

-- attribute data types --
--      Definition of the following information object set is deferred, perhaps to standardized
--      profiles or to protocol implementation conformance statements. The set is required to
--      specify a table constraint on the values component of Attribute, the value component
--      of AttributeTypeAndValue, and the assertion component of AttributeValueAssertion.
SupportedAttributes ATTRIBUTE ::= { objectClass | aliasedEntryName , ...}


-- Attribute definitions

methodUse ATTRIBUTE ::= {
                WITH SYNTAX                     MethodUseDescription
                EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
                USAGE                           directoryOperation
                ID                              id-soa-methodRuleUse }
-- The methodUse operational attribute is used to indicate the methods which shall be used with an
-- object-class and all of its subclasses.
```

```
MethodUseDescription    ::=    SEQUENCE {
        identifier                  OBJECT-CLASS.&id,
        name                        SET OF DirectoryString { ub-schema } OPTIONAL,
        description                 DirectoryString { ub-schema } OPTIONAL,
        obsolete                    BOOLEAN DEFAULT FALSE,
        information                 [0]    SET OF METHOD.&id }

-- The identifier component of a value of the methodUse operational attribute is the object identifier
--  of the object-class type to which it applies.  The value id-oa-allObject-classTypes indicates that it
--  applies to all object-class types.
-- The information component of a value identifies the method types associated with the object-class
--  identified by identifier.
-- Every entry in the DIT is governed by at most one methodUse operational attribute. In addition the
-- entry is also governed by all the methodUse operation attribute defined for the superclasses of its
-- structural object class.
-- NOTE - This means that before processing an execute operation the SDF shall check the methodUse
-- attributes associated with the structural object classes which belong to the inheritance chain of the
-- entry's structural object class.

-- As a methodRule attribute is associated with a structural object class, it follows that all of the entries
-- on the same structural object class will have the same Method Use Rule regardless of the DIT structure
-- rule governing their location in the DIT and of the DIT content rule governing their contents.

securityFacilityId ATTRIBUTE ::= {
        WITH SYNTAX                     SF-CODE
        EQUALITY MATCHING RULE          objectIdentifierMatch
        SINGLE VALUE                    TRUE
        ID                              id-at-securityFacilityId}
SF-CODE ::= OBJECT IDENTIFIER
-- securityFacilityId is an attribute to name the verification

secretKey  ATTRIBUTE ::= {
        WITH SYNTAX    BIT STRING(SIZE(lb-secretKey..ub-secretKey))
        SINGLE VALUE   TRUE
        ID             id-at-secretKey}
-- secretKey is an attribute which contains the secret key (to be used by the cryptographic algorithm)
-- of the user

lb-secretKey  INTEGER ::= 32
ub-secretKey  INTEGER ::= 128

identifierList ATTRIBUTE ::= {
WITH SYNTAX             SEQUENCE{
conformMethodIdentifier         [1]     MethodIdentifier, -- e.g. time window check
fillMethodIdentifier            [2]     MethodIdentifier,-- e.g. generate a random of required size
oneToOneAlgorithm               [3]     AlgorithmIdentifier,
                                -- e.g. A11 and A12, output RES from RS,RAND
oneToTwoAlgorithm               [4]     AlgorithmIdentifier }
                                -- e.g DECT algorithm output RES,SDK from RS,RAND
SINGLE VALUE                    TRUE
ID                              id-at-identifierList}
-- identifierList is an attribute that could contain four identifiers:
-- conformMethodIdentifier identifies the method used to verify that some parts of the input message
--  are conformed to some criteria as size, value matching with an attribute, greater than a counter,
-- included in a time window,
-- fillMethodIdentifier identifies the method used to fill the input message (first part of a
-- twoPartMessage or ThreePartMessage or FivePartMessage).
-- oneToOneAlgorithm (resp. oneToTwoAlgorithm) identifies the cryptographic algorithm with one
-- output (resp. two output). if KS is the secret key, IN is the input and OUT the output, it would be
-- OUT=output1of (A12(RS_size_in_bits first bits of IN,A11(RAND_size_in_bits last bits of IN,KS)))
-- (resp. (OUT1,OUT2)= (A12(RS_size_in_bits first bits of IN,A11(RAND_size_in_bits last bits of
-- IN,KS)) )
```

```
MethodIdentifier ::= SEQUENCE {
methodid            METHOD.&id ({SupportedMethods}),
inputAttributes     METHOD.&InputAttributes ({SupportedMethods}{@method}) OPTIONAL,
specific-Input      METHOD.&SpecificInput ({SupportedMethods}{@method}) OPTIONAL}
bindLevelIfOK ATTRIBUTE ::= {
WITH SYNTAX     AuthenticationLevel
SINGLE VALUE    TRUE
ID              id-at-bindLevelIfOK}
-- bindLevelIfOK is a mono-valued attribute that contains an AuthenticationLevel. It is to be used by the
-- bind operation to determine the level of privileges granted to the user. When this attribute is absent
-- and a bind operation is invoked, the bind operation returns the error provided by the method.

lockSession ATTRIBUTE ::= {
WITH SYNTAX     LockSession
SINGLE VALUE    TRUE
ID              id-at-lockSession}

LockSession  ::= SEQUENCE {
entryName   [0]     DistinguishedName,
atribute    [1]     OBJECT IDENTIFIER}
-- lockSession is a mono-valued attribute that contains the name of the entry and the mono-valued
-- attribute of type boolean of this entry used to lock a dialogue to a mono-session (the timer set as temporal
-- context on this lock attribute is the same for all the users). If this attribute is present and a bind
-- operation is at the origin of the method invocation, the method checks first that the pointed attribute is
-- FALSE before proceeding.

-- For some security facilities, it is useful to count the number of failures and if necessary to lock the
-- facility when a threshold is reached. The two following attributes are used to store this information
failureCounter ATTRIBUTE ::= {
WITH SYNTAX                     INTEGER
ORDERING MATCHING RULE          integerOrderingMatch
SINGLE VALUE                    TRUE
ID                              id-at-failureCounter}

maxAttempts ATTRIBUTE ::= {
WITH SYNTAX                     INTEGER
ORDERING MATCHING RULE          integerOrderingMatch
SINGLE VALUE                    TRUE
ID                              id-at-maxAttempts}

-- To check, the no replay of a challenge RAND drawn in another domain, it is necessary to maintain a
-- list of the random already used for the valid period indicated by RS. The currentList attribute contains
-- a list of RAND already played for the current period of time.
currentList  ATTRIBUTE ::= {
        WITH SYNTAX                     BIT STRING
        EQUALITY MATCHING RULE          bitStringMatch
        ID                              id-at-currentList}

stockId  ATTRIBUTE ::= {
WITH SYNTAX             DT-Code
EQUALITY MATCHING RULE      objectIdentifierMatch
SINGLE VALUE            TRUE
ID                      id-at-stockId}
DT-Code ::= OBJECT IDENTIFIER
-- stockId is a mono valued attribute of type DT-Code that is used as naming attribute

source ATTRIBUTE ::={
WITH SYNTAX     SourceType
SINGLE VALUE    TRUE
ID          id-at-source}

SourceType ::=DistinguishedName
```

-- In the visited network, the source attribute will be used to store the DN of the entry of class derived
-- from stockId. In the home network, the attribute will contain the DN of an entry of class
-- securityUserInfo, the token is generated using the method defined in the fillMethodIdentifier field of
-- this entry of class securityUserInfo.

sizeOfRestocking ATTRIBUTE ::= {
WITH SYNTAX **INTEGER**
ORDERING MATCHING RULE **integerOrderingMatch**
SINGLE VALUE **TRUE**
ID **id-at-sizeOfRestocking }**
-- sizeOfRestocking is a mono-valued attribute that indicates how many tokens have to be requested or
-- computed when the tokens attribute is empty.

-- The following attribute could contain the precomputed set of
-- (CHALLENGE,RES[,DCK][,NCHALLENGE,NRES]) (2, 3,4 or 5 values)
stock{INTEGER: n} ATTRIBUTE ::= {
WITH SYNTAX **NPartsMessage{n}**
ID **id-at-challengeResponse}**

prescriptiveACI **ATTRIBUTE ::=** **{**
WITH SYNTAX **ACIItem**
EQUALITY MATCHING RULE **directoryStringFirstComponentMatch**
USAGE **directoryOperation**
ID **id-aca-prescriptiveACI }**

entryACI **ATTRIBUTE ::=** **{**
WITH SYNTAX **ACIItem**
EQUALITY MATCHING RULE **directoryStringFirstComponentMatch**
USAGE **directoryOperation**
ID **id-aca-entryACI }**

subentryACI **ATTRIBUTE ::=** **{**
WITH SYNTAX **ACIItem**
EQUALITY MATCHING RULE **directoryStringFirstComponentMatch**
USAGE **directoryOperation**
ID **id-aca-subentryACI }**

-- **Attribute contexts** definitions

basicServiceContext **CONTEXT ::= {**
WITH SYNTAX **BasicService**
ID **id-avc-basicService}**
BasicService ::= INTEGER {
telephony (1),
faxGroup2-3 (2),
faxGroup4 (3),
teletexBasicAndMixed (4),
teletexBazicAndProcessable (5),
teletexBasic (6),
syntaxBasedVideotex (7),
internationalVideotex (8),
telex (9),
messageHandlingSystems (10),
osiApplication (11),
audioVisual (12)}
-- This Basic Service context associates an attribute value with a basic service for which the attribute
-- value is semantically valid. For example, the Basic Service context will be associated with an ISDN
-- address to indicate the type of basic service that could be used with it. In the UPT case, this context
-- allows the definition of registration addresses for different basic services.
-- A presented value is considered to match a stored value if the context value (i.e., a basic service value)
-- in the presented value is identical to that in the stored value.

```
lineIdentityContext {SCF-SSF-BOUNDS : b2} CONTEXT ::= {
        WITH SYNTAX     IsdnAddress {b2}
        ID              id-avc-lineIdentity}

IsdnAddress {SCF-SSF-BOUNDS : b2} ::= Digits {b2}
-- The line identity context associates an attribute value with the identity of a line for which the attribute
-- value is semantically valid.  For example, this Line Identity context will be associated with a routing
-- number to provide calling-line dependent routing.
-- Q763 Generic Digits is applied for encoding.  The bound definition is as follows:
sCFSSFBoundSetforSCFSDF SCF-SSF-BOUNDS ::=
{       MINIMUM-FOR-DIGITS1
        MAXIMUM-FOR-DIGITS      10  }
-- This is an example, and appropriate values will be defined as network specific.

assignmentContext CONTEXT ::= {
        WITH SYNTAX     DistinguishedName
        ID              id-avc-assignment }
-- The assignment context associates an attribute value with a Distinguished name (e.g. customer's
-- number or customer's name) for which the attribute value is assigned. For example, assuming that a set
-- of available resources is modelled as a multivalued attribute and customer has been designated by a
-- distinguished name, this Assignment context will be associated with the used resource to provide the
-- state of the resource (reserved) and the name of the current customer using it.

END
```

-- 12.2 OPERATIONS AND ARGUMENTS DEFINITION

```
IN-CS3-SCF-SDF-Operations
{ccitt recommendation q 1238 modules(1) in-cs3-scf-sdf-ops-args(16) version1(0)}

DEFINITIONS ::=

BEGIN

IMPORTS

ds-UsefulDefinitions, ros-InformationObjects, errortypes, operationcodes, ssf-scf-datatypes, scf-scf-classes,
scf-sdf-datatypes, scf-sdf-classes
        FROM IN-CS3-object-identifiers
                {ccitt recommendation q 1238 modules(1) in-cs3-object-identifiers(0) version1(0)}

informationFramework, enhancedSecurity, directoryAbstractService
        FROM UsefulDefinitions ds-UsefulDefinitions

Context, Name
        FROM InformationFramework informationFramework

OPERATION
        FROM Remote-Operations-Information-Objects ros-InformationObjects

OPTIONALLY-PROTECTED{}, DIRQOP
        FROM EnhancedSecurity enhancedSecurity

CommonArguments, CommonResults, attributeError, nameError, referral, securityError, serviceError,
updateError, Credentials, Versions, SecurityParameters
        FROM DirectoryAbstractService       directoryAbstractService

executionError, tfcBindError
        FROM        IN-CS3-errortypes errortypes

opcode-execute, opcode-trafficFlowControl
        FROM        IN-CS3-OperationCodes operationcodes
```

```
Duration, ControlType
        FROM        IN-CS3-SSF-SCF-datatypes ssf-scf-datatypes

SCFQOP
        FROM        IN-CS3-SCF-SCF-classes scf-scf-classes

TFCcriteria
        FROM        IN-CS3-SCF-SDF-datatypes scf-sdf-datatypes

METHOD, SupportedMethods
        FROM  IN-CS3-SCF-SDF-Classes scf-sdf-classes
;

execute OPERATION ::= {
        ARGUMENT        ExecuteArgument
        RESULT          ExecuteResult
        ERRORS          { attributeError | nameError |  serviceError | referral |
                        securityError | updateError | executionError }
        CODE            opcode-execute }
-- Direction: SCF->SDF
-- Note that the serviceError error is subtyped as described in subclause 12.5 for the IN specification.
-- Refer to the in-ServiceError definition for it.

ExecuteArgument ::=  OPTIONALLY-PROTECTED {
        SET {
                object              [0] Name,
                method-id           [1] METHOD.&id({SupportedMethods}),
                input-assertions    [2] SEQUENCE OF SEQUENCE {
                                        type
        METHOD.&InputAttributes.&id({SupportedMethods}{@method-id}),
                                        values SET OF
METHOD.&InputAttributes.&Type({SupportedMethods}{@method-id,@.type}) OPTIONAL,
                                        valuesWithContext [0] SET OF SEQUENCE {
                                                value  [0]
        METHOD.&InputAttributes.&Type({SupportedMethods}{@method-id,@.type})      OPTIONAL,
                                                contextList [1] SET OF Context
                                                } OPTIONAL
                                        } OPTIONAL,
                specific-input      [3] METHOD.&SpecificInput({SupportedMethods}{@method-id})
                                        OPTIONAL,
                COMPONENTS OF CommonArguments },
        DIRQOP.&dapModifyEntryArg-QOP{@qop} }
-- Note that CommonArguments in the ExecuteArgument is replaced with IN-CommonArguments
-- described as the subtyped definition in subclause 12.5.

ExecuteResult ::= OPTIONALLY-PROTECTED {
        SET {
                method-id           [1] METHOD.&id({SupportedMethods}),
                output-assertions   [2] SEQUENCE OF SEQUENCE {
                                        type
        METHOD.&OutputAttributes.&id({SupportedMethods}{@method-id}),
                                        values SET OF
METHOD.&OutputAttributes.&Type({SupportedMethods}{@method-id,@.type})OPTIONAL,
                                        valuesWithContext [0] SET OF SEQUENCE {
                                                value  [0]
        METHOD.&OutputAttributes.&Type({SupportedMethods}{@method-id,@.type})    OPTIONAL,
                                                contextList   [1] SET OF Context
                                                } OPTIONAL
                                        } OPTIONAL,
                specific-output     [3] METHOD.&SpecificOutput({SupportedMethods}{@method-id})
                                        OPTIONAL,
                COMPONENTS OF CommonResults },
```

```
        DIRQOP.&dapModifyEntryRes-QOP{@qop} }

-- The specific-output field contains information returned as a result of the method execution.
-- The output-assertions contains attributes values returned as a result of the method execution.

-- addEntry OPERATION
-- Direction: SCF->SDF
-- Note that the serviceError error is subtyped as described in subclause 12.5 for the IN specification.
-- Refer to the in-ServiceError definition for it.
-- Note that CommonArguments in X.511 AddEntryArgument is replaced with  IN-CommonArguments described as
-- the subtyped definition in subclause 12.5.

-- directoryBind OPERATION
-- Direction: SCF->SDF
-- Note that the directoryBindError error is subtyped as described in subclause 12.5 for the IN specification.  Refer to
-- the in-DirectoryBindError definition for it.

-- modifyEntry OPERATION
-- Direction: SCF->SDF
-- Note that the ModifyEntryArgument, ModifyEntryResult, and serviceError error are subtyped as described in
-- subclause 12.5 for the IN specification.  Refer to the IN-ModifyEntryArgument, IN-ModifyEntryResult, and in-
-- ServiceError definitions for them, respectively.

-- removeEntry OPERATION
-- Direction: SCF->SDF
-- Note that the serviceError error is subtyped as described in subclause 12.5 for the IN specification.
-- Refer to the in-ServiceError definition for it.
-- Note that CommonArguments in X.511 RemoveEntryArgument is replaced with
-- IN-CommonArguments described as the subtyped definition in subclause 12.5.

-- search OPERATION
-- Direction: SCF->SDF
-- The search operation is used to search a portion of the DIT for entries of interest.
-- The abandoned error is not supported.
-- Note that the SearchArgument, SearchResult, and serviceError error are subtyped as described in
-- subclause 12.5 for the IN specification.  Refer to the IN-SearchArgument, IN-SearchResult, and in-
-- ServiceError definitions for them, respectively.

-- inUnbind  OPERATION
-- Direction: SCF->SDF, SDF->SCF (TFC case)
-- This operation is described in Q.1238.1.

trafficFlowControl OPERATION ::= {
        ARGUMENT                TFCArgument
        RETURN RESULT           FALSE
        ALWAYS RESPONDS         FALSE
        CODE                    opcode-trafficFlowControl }
-- Direction: SDF->SCF

TFCArgument ::=  OPTIONALLY-PROTECTED{
        SEQUENCE {
                tfcCriteria             [0]     IMPLICIT   TFCcriteria,
                duration                [1]     IMPLICIT   Duration,
                controlType             [2]     IMPLICIT   ControlType,
                security                [3]     IMPLICIT   SecurityParameters,
                wasChained              [4]     IMPLICIT   BOOLEAN  DEFAULT FALSE,
                ...,
                ...},
        SCFQOP.&scfArgumentQOP(@dirqop)}

tfcBind OPERATION ::= {
        ARGUMENT        TFCBindArgument
```

```
        RESULT          TFCBindResult
        ERRORS          {tfcBindError }
        }
-- Direction: SDF->SCF

TFCBindArgument::=  SET {
        credentials  [0]  IMPLICIT  Credentials OPTIONAL,
        versions     [1]  IMPLICIT  Versions DEFAULT {v1},
        ...,
        ...}
-- Note: the tfcBind.credentials.strong.bind-token.dirqop field should be set to a value which corresponds
--       to a SCFQOP object.
--       The absence of this field in the tfcBind argument implies the non-use of protection facilities.

TFCBindResult    ::=  TFCBindArgument

END

-- The table below lists operation timer value range. The definitive timer value may be network
-- specific and has to be defined by the network operator.
--
--          Table 2/Q.1238.4 – Operation timer value
--          Operation                       Timer value
--          - - - - - - - - -- - - - - - - - - - - - - - - - - - - - -
--          IN-Search                       medium
--          IN-ModifyEntry                  medium
--          IN-AddEntry                     medium
--          IN-RemoveEntry                  medium
--          Execute                         medium
--          TrafficFlowControl              medium
--                              medium : 1 - 60 second

-- 12.3  CONTRACTS, PACKAGES, AND ABSTRACT SYNTAXES

IN-CS3-SCF-SDF-Protocol
        {ccitt recommendation q 1238 modules(1) in-cs3-scf-sdf-pkgs-contracts-acs(17) version1(0)}

DEFINITIONS ::=

BEGIN

IMPORTS

ds-UsefulDefinitions, ros-InformationObjects, ros-genericPDUs, tc-Messages, tc-NotationExtensions,
sese-APDUs,
common-classes,
scf-sdf-Operations,
id-ac-indirectoryAccessAC, id-ac-inExtendedDirectoryAccessAC, id-contract-dap, id-contract-dapExecute,
id-package-dapConnection, id-package-search, id-package-modify, id-package-execute,
id-as-indirectoryOperationsAS, id-as-inExtendedDirectoryOperationsAS, id-as-indirectoryBindingAS,
id-as-inSESEAS, id-ac-inExtendedDirectoryAccessWith3seAC, id-ac-indirectoryAccessWith3seAC,
id-ac-trafficFlowControlAC, id-contract-tfc, id-package-tfcConnection, id-package-tfcOperations,
id-as-tfcOperationsAS, id-as-tfcBindingAS
        FROM IN-CS3-object-identifiers
                {ccitt recommendation q 1238 modules(1) in-cs3-object-identifiers(0) version1(0)}

directoryAbstractService, protocolObjectIdentifiers
        FROM UsefulDefinitions ds-UsefulDefinitions

directoryBind, search, addEntry, removeEntry, modifyEntry
```

```
        FROM DirectoryAbstractService directoryAbstractService

spkmThreeWay
        FROM DirectorySecurityExchanges {joint-iso-ccitt ds(5) module(1) dir-se(29) 1}

id-se-threewayse
        FROM ProtocolObjectIdentifiers protocolObjectIdentifiers

CONTRACT, OPERATION-PACKAGE, CONNECTION-PACKAGE, OPERATION
        FROM Remote-Operations-Information-Objects ros-InformationObjects

Bind{}, Unbind{}
        FROM Remote-Operations-Generic-ROS-PDUs ros-genericPDUs

TCMessage {}
        FROM  TCAPMessages tc-Messages

APPLICATION-CONTEXT, dialogue-abstract-syntax
        FROM TC-Notation-Extensions tc-NotationExtensions

SESEapdus{}, NoInvocationId
        FROM SeseAPDUs sese-APDUs

InUnbind, EmptyReturnable
        FROM IN-CS3-common-classes common-classes
execute, tfcBind, trafficFlowControl
        FROM IN-CS3-SCF-SDF-Operations scf-sdf-Operations
;
-- application contexts --
-- Though new application context IDs are assigned in all the following APPLICATION-CONTEXT
-- definition, note that the functionality are identical to IN CS-2.

iNdirectoryAccessAC APPLICATION-CONTEXT ::= {
        CONTRACT                        dapContract
        DIALOGUE MODE                   structured
        TERMINATION                     basic
        ABSTRACT SYNTAXES               {dialogue-abstract-syntax |
                                        inDirectoryOperationsAbstractSyntax |
                                        inDirectoryBindingAbstractSyntax}
        APPLICATION CONTEXT NAME        id-ac-indirectoryAccessAC}
-- The iNdirectoryAccessAC is used for the realization of dapContract.

iNdirectoryAccessWith3seAC APPLICATION-CONTEXT ::= {
        CONTRACT                        dapContract
        DIALOGUE MODE                   structured
        TERMINATION                     basic
        ADDITIONAL ASEs                 {id-se-threewayse}
        ABSTRACT SYNTAXES               {dialogue-abstract-syntax |
                                        inDirectoryOperationsAbstractSyntax |
                                        inDirectoryBindingAbstractSyntax |
                                        inSESEAbstractSyntax }
        APPLICATION CONTEXT NAME        id-ac-indirectoryAccessWith3seAC}
-- The iNdirectoryAccessWith3seAC is used for the realization of dapContract when 3-way
-- authentication is required.

inExtendedDirectoryAccessAC APPLICATION-CONTEXT ::= {
        CONTRACT                        dapExecuteContract
        DIALOGUE MODE                   structured
        TERMINATION                     basic
        ABSTRACT SYNTAXES               {dialogue-abstract-syntax |
                                        inExtendedDirectoryOperationsAbstractSyntax |
                                        inDirectoryBindingAbstractSyntax}
```

```
        APPLICATION CONTEXT NAME     id-ac-inExtendedDirectoryAccessAC}
-- The inExtendedDirectoryAccessAC is used for the realization of dapExecuteContract.

inExtendedDirectoryAccessWith3seAC APPLICATION-CONTEXT ::=  {
        CONTRACT                     dapExecuteContract
        DIALOGUE MODE                structured
        TERMINATION                  basic
        ADDITIONAL ASEs              {id-se-threewayse}
        ABSTRACT SYNTAXES            {dialogue-abstract-syntax |
                                     inExtendedDirectoryOperationsAbstractSyntax |
                                     inDirectoryBindingAbstractSyntax |
                                     inSESEAbstractSyntax }
        APPLICATION CONTEXT NAME     id-ac-inExtendedDirectoryAccessWith3seAC}
-- The inExtendedDirectoryAccessWith3seAC is used for the realization of dapExecuteContract when
-- 3-way authentication is required.

trafficFlowControlAC APPLICATION-CONTEXT ::=  {
        CONTRACT                     tfcContract
        DIALOGUE MODE                structured
        TERMINATION                  basic
        ABSTRACT SYNTAXES            {dialogue-abstract-syntax |
                                      tfcAbstractSyntax |
                                      tfcBindingAbstractSyntax}
        APPLICATION CONTEXT NAME     id-ac-trafficFlowControlAC}
-- The trafficFlowControlAC is used for the realization of tfcContract.

-- contracts --
dapContract CONTRACT ::= {
        CONNECTION                   dapConnectionPackage
        INITIATOR CONSUMER OF        {searchPackage | modifyPackage}
        ID                           id-contract-dap}

dapExecuteContract CONTRACT ::= {
        CONNECTION                   dapConnectionPackage
        INITIATOR CONSUMER OF        {searchPackage | modifyPackage | executePackage}
        ID                           id-contract-dapExecute}

tfcContract CONTRACT ::= {
        CONNECTION                   tfcConnectionPackage
        INITIATOR CONSUMER OF        {tfcPackage}
        ID                           id-contract-tfc}

-- packages --
dapConnectionPackage CONNECTION-PACKAGE ::= {
        BIND        directoryBind
        UNBIND      inUnbind
        ID          id-package-dapConnection}

tfcConnectionPackage CONNECTION-PACKAGE ::= {
        BIND        tfcBind
        UNBIND      inUnbind
        ID          id-package-tfcConnection}

searchPackage OPERATION-PACKAGE ::= {
        CONSUMER INVOKES  {search}
        ID                id-package-search}

modifyPackage OPERATION-PACKAGE ::= {
        CONSUMER INVOKES  {addEntry | removeEntry | modifyEntry}
        ID                id-package-modify}

executePackage OPERATION-PACKAGE ::= {
        CONSUMER INVOKES  {execute}
```

```
        ID                          id-package-execute}

tfcPackage OPERATION-PACKAGE ::= {
        CONSUMER INVOKES  {trafficFlowControl}
        ID                          id-package-tfcOperations}

-- abstract-syntaxes --
inDirectoryOperationsAbstractSyntax ABSTRACT-SYNTAX ::= {
        BasicDAP-PDUs
        IDENTIFIED BY    id-as-indirectoryOperationsAS}

BasicDAP-PDUs ::= TCMessage {{DAP-Invokable},{DAP-Returnable}}

DAP-Invokable  OPERATION ::= {search | addEntry | removeEntry | modifyEntry}
DAP-Returnable OPERATION ::= {search | addEntry | removeEntry | modifyEntry}

inExtendedDirectoryOperationsAbstractSyntax ABSTRACT-SYNTAX ::= {
        Extended-BasicDAP-PDUs
        IDENTIFIED BY    id-as-inExtendedDirectoryOperationsAS}

Extended-BasicDAP-PDUs ::= TCMessage {{Extended-DAP-Invokable},{Extended-DAP-Returnable}}

Extended-DAP-Invokable  OPERATION ::= {search | addEntry | removeEntry | modifyEntry | execute}
Extended-DAP-Returnable OPERATION ::= {search | addEntry | removeEntry | modifyEntry | execute}

inDirectoryBindingAbstractSyntax       ABSTRACT-SYNTAX ::= {
        DAPBinding-PDUs
        IDENTIFIED BY    id-as-indirectoryBindingAS}

DAPBinding-PDUs ::= CHOICE {
        bind            Bind {directoryBind},
        unbind          Unbind {inUnbind}}

inSESEAbstractSyntax     ABSTRACT-SYNTAX ::= {
        SESEapdus {{spkmThreeWay},NoInvocationId}
        IDENTIFIED BY    id-as-inSESEAS}

tfcAbstractSyntax ABSTRACT-SYNTAX ::= {
        BasicTFC-PDUs
        IDENTIFIED BY    id-as-tfcOperationsAS}

BasicTFC-PDUs ::= TCMessage {{TFC-Invokable},{EmptyReturnable}}

TFC-Invokable  OPERATION ::= {trafficFlowControl}

tfcBindingAbstractSyntax ABSTRACT-SYNTAX ::= {
        TFCBinding-PDUs
        IDENTIFIED BY    id-as-tfcBindingAS}

TFCBinding-PDUs ::=     CHOICE      {
        bind            Bind{tfcBind},
        unbind          Unbind {inUnbind}}

END
```

-- 12.4 CLASS DEFINITION

```
IN-CS3-SCF-SDF-Classes
                {ccitt recommendation q 1238 modules(1) in-cs3-scf-sdf-classes(15) version1(0) }

DEFINITIONS ::=
```

```
BEGIN

IMPORTS

ds-UsefulDefinitions, scf-sdf-datatypes,
id-mt-verifyCredentials, id-mt-conformCredentials, id-mt-provideTokens, id-mt-fillSecurityTokens,
id-oc-securityUserInfo, id-oc-securityUserInfo, id-oc-tokensStock
        FROM IN-CS3-object-identifiers
                {ccitt recommendation q 1238 modules(1) in-cs3-object-identifiers(0) version1(0)}

informationFramework
        FROM UsefulDefinitions ds-UsefulDefinitions

ATTRIBUTE, OBJECT-CLASS
        FROM InformationFramework informationFramework

NPartsMessage{}, TwoPartMessage,
securityFacilityId, secretKey, identifierList, bindLevelIfOK, currentList, failureCounter, lockSession,
maxAttempts, stockId, stock{}, source, sizeOfRestocking
        FROM IN-CS3-SCF-SDF-datatypes scf-sdf-datatypes
;

-- METHOD information object class specification --
METHOD ::= CLASS {
        &InputAttributes        ATTRIBUTE OPTIONAL,
        &SpecificInput          OPTIONAL,
        &OutputAttributes       ATTRIBUTE OPTIONAL,
        &SpecificOutput         OPTIONAL,
        &description            PrintableString OPTIONAL,
        &id                     OBJECT IDENTIFIER UNIQUE}
WITH SYNTAX {
        [INPUT ATTRIBUTES           &InputAttributes ]
        [SPECIFIC-INPUT             &SpecificInput ]
        [OUTPUT ATTRIBUTES          &OutputAttributes ]
        [SPECIFIC-OUTPUT            &SpecificOutput ]
        [BEHAVIOUR                  &description]
        ID                          &id}
-- The &InputAttributes field identifies the attributes which may be submitted as input
-- to the method execution.
-- The &OutputAttributes field identifies the attributes which may be returned as output
-- of the method execution.
-- The &SpecificInput field provides that syntax of additional information
-- which may be used as input to the method execution.
-- The &SpecificOutput field provides that syntax of additional information
-- which may be used as output to the method execution.
-- The &id field uniquely identifies the method.
-- method --

verifyCredentials METHOD ::={
SPECIFIC-INPUT  TwoPartMessage
-- see the definition of this type below
SPECIFIC-OUTPUT       BOOLEAN -- to indicate the success of the verification
BEHAVIOUR       "This method performs the following actions on the entry identified by the execute
argument, this entry would be of class genericSecurityUserInfo:
        1) if maxattempts is present, verify that failureCounter is inferior to its value
        2) read the value of identifierList attribute
        (return bad format entry if failure)
        3) if conformMethodIdentifier is NULL go to step 5)
        4) run conformMethodIdentifier method on TwopartMessage provided as specific input
        (return a badconformance error if the execution fails or if the result is false)
        5) run the oneToOneAlgorithm on the messageData bit string to get an expected
        certificationCode bit string
        6) return TRUE if the expected and provided certificationCode values matched,
```

7) otherwise if failureCounter is present, increment it
8) return FALSE
"
ID id-mt-verifyCredentials}

conformCredentials METHOD ::={
SPECIFIC-INPUT TwoPartMessage
-- see the definition of this type below
SPECIFIC-OUTPUT BOOLEAN *-- to indicate the success of the verification*
BEHAVIOUR "This method performs the following actions on the entry identified by the execute argument, this entry would be of class genericSecurityUserInfo:
- verify with an embedded conformance algorithm that messageData value of TwoPartMessage is no replay (RAND is in the current time window and the associated RS is not in the list of the current time windows currentList).
- add RAND to time windows list currentList.
- return TRUE if no replay,
- otherwise return FALSE
"
ID id-mt-conformCredentials}

provideTokens METHOD ::={
SPECIFIC-INPUT SEQUENCE{
numOfRequestedTokens INTEGER, *-- how many tokens are requested (NOfRT)*
oidOfAttribute OBJECT IDENTIFIER} *-- oid of the attribute (tokens)*
SPECIFIC-OUTPUT ATTRIBUTE *--attribute selected as input (tokens)*
BEHAVIOUR "This method performs the following actions on the entry (thisEntry would be a variable with the DN value of this entry) identified by the execute argument:
1) if the attribute sizeOfRestocking does not exist in the entry, define a variable MAXNT with a default value (e.g. 10) else MAXNT takes the value of the attribute sizeOfRestocking
2) verify that NofRT is inferior or egal to MAXNT
(return an execute error if NofRT value is superior to MAXNT)
3) read the attribute of the entry which has the selected oid and count the number of values (0 if empty) and put the result in a variable N
(return execute error if the attribute does not exist)
4) read the source attribute in the entry
(return execute error error if source does not exist)
5) if N is inferior to NOfRT and the DN of source indicates an entry of class or subclass tokenStock:
5a) bind anonymous with the DSA which contains the entry defined by the address field of source
5b) execute the method provideTokens on the entry with MAXNT as value of the specific-input
5c) if none error is returned, modify the tokens attribute by adding the resulted values
6) if N is inferior to NOfRT and the DN of source indicates an entry of class or subclass securityUserInformation
6a) execute the method defined by fillMethodIdentifier field value on the entry defined by the DN with MAXNT as specific input
6b) if none error is returned, modify the tokens attribute by adding the resulted values
7) read the tokens attribute
8) define a variable toBeReturned with NofRT values of tokens attribute and a variable toBeKept with remainder values
9) remove tokens attribute
10) modify tokens attribute by adding the toBeKept values
11) return the toBeReturned values
"
ID id-mt-provideTokens}

fillSecurityTokens {INTEGER:n} METHOD ::={
SPECIFIC-INPUT INTEGER *-- X number of value to be computed*
SPECIFIC-OUTPUT SEQUENCE OF NPartsMessage{n}
BEHAVIOUR "This method performs the following actions on the entry identified by the execute argument, this entry shall be of object class (or subclass) genericSecurityUserInfo:
- read the secretKey attribute and Algorithms attribute
- repeat X times
- fill the first BIT STRING field with a random value
- apply cryptographic algorithms to compute

```
                the other  BIT STRING fields of the NPARTMESSAGE.
-       return X NPartMessage values
"
ID              id-mt-fillSecurityTokens}

SupportedMethods METHOD ::= { ... }

-- The SupportedMethods set contains all of the defined methods for the interface. Its exact contents are
-- a matter for local determination as it will depend on the service and network provider agreements
-- being supported.

-- A DIT METHOD Use is a specification provided by the subschema administrative authority to specify
-- the METHOD types that may be used on entries of a particular object-class.
DITMethodUse    ::=     SEQUENCE {
        objectClass             OBJECT-CLASS.&id,
        methods                 [1]     SET OF METHOD.&id }
-- a) the objectClass component identifies the object-class to which the DIT METHOD Use applies;
-- b) the methods component specifies types that shall be associated with the object-class whenever
-- entries of that object-class are stored;
-- The DITMethodUse definition for a particular object-class also applies to any subclass which may be
-- subsequently defined.

-- The METHOD-USE-RULE information object class is provided to facilitate the documentation of the
--  DIT METHOD Use rules:
METHOD-USE-RULE ::= CLASS {
        &objectClassType                OBJECT-CLASS.&id        UNIQUE,
        &Mandatory                      METHOD }
WITH SYNTAX {
        OBJECT-CLASS TYPE               &objectClassType
        METHODS                         &Mandatory }
-- The METHOD-USE-RULE definition for a particular object-class also applies to any subclass
-- which may be subsequently defined.

-- method-use-rule --
securityUserInfoRule{INTEGER:n} METHOD-USE-RULE ::= {
OBJECT-CLASS TYPE               id-oc-securityUserInfo
METHODS         {verifyCredentials| fillSecurityTokens{n}|conformCredentials}}

-- object class --
-- The following object class could be used to store the necessary information about the user security.
-- For each UPT user, an entry of the securityUserInfo object class may be created, subordinated to each
-- entry of class uptUser.
securityUserInfo OBJECT-CLASS ::={
MUST CONTAIN            {securityFacilityId|
                        secretKey|
                        identifierList}
MAY CONTAIN             {bindLevelIfOK|
                        currentList|
                        failureCounter|
                        lockSession|
                        maxAttempts}
ID                      id-oc-securityUserInfo }
-- The object class SecurityUserInfo supports the method verifyCredentials
tokensStock {INTEGER: n}  OBJECT-CLASS ::={
KIND                    abstract
MUST CONTAIN            {stockId | stock{n}}
MAY CONTAIN             {source|
                        sizeOfRestocking}
ID                      id-oc-tokensStock}
-- This object class is used to represent a set of information that is common to all disposable tokens
-- (stock identifier, source, and size of the set). Disposable tokens could be, for example authentication
-- tokens pairs, triplets.
```

```
END
```

-- 12.5 INAP DEFINED BY SUBTYPED X.500 RELATED DEFINITION
--
-- This clause describes ASN.1 definition defined by subtyping X.500 related definitions
-- or by using them. The following module is only for information, and the syntax is not assured.

```
IN-CS3-SCF-SDF-Additional-Definitions
        { ccitt recommendation q 1238 modules(1) in-cs3-scf-sdf-additional-definitions(28) version1(0)}

DEFINITIONS ::=

BEGIN

IMPORTS

ds-UsefulDefinitions, operationcodes, errortypes, scf-sdf-Operations
        FROM IN-CS3-object-identifiers
                {ccitt recommendation q 1238 modules(1) in-cs3-object-identifiers(0) version1(0)}

directoryAbstractService, dap
        FROM UsefulDefinitions ds-UsefulDefinitions

CommonArguments, ServiceControls, EntryInformationSelection, EntryInformation,
AddEntryArgument, AddEntryResult, DirectoryBindArgument, DirectoryBindResult, RemoveEntryArgument,
RemoveEntryResult, SearchArgument, SearchResult, ModifyEntryArgument, ModifyEntryResult,
PartialOutcomeQualifier, attributeError, directoryBindError, nameError, referral, securityError,
serviceError, updateError
        FROM DirectoryAbstractService  directoryAbstractService

id-opcode-search, id-opcode-addEntry, id-opcode-removeEntry, id-opcode-modifyEntry
        FROM        DirectoryAccessProtocol dap

opcode-execute
        FROM        IN-CS3-OperationCodes operationcodes

executionError
        FROM        IN-CS3-errortypes errortypes

execute, ExecuteArgument, ExecuteResult
        FROM IN-CS3-SCF-SDF-Operations scf-sdf-Operations
;
-- Information types and common procedures
IN-CommonArguments ::= CommonArguments (
        WITH COMPONENTS {
                ...,
                serviceControls             (IN-ServiceControls),
                aliasedRDNs                 ABSENT })

IN-ServiceControls ::= ServiceControls
        (WITH COMPONENTS {
                ...,
                timeLimit               ABSENT,
                sizeLimit               ABSENT,
                scopeOfReferral         ABSENT,
                attributeSizeLimit      ABSENT})

IN-EntryInformationSelection ::= EntryInformationSelection
        (WITH COMPONENTS {
                ...,
                infoTypes               (attributeTypesAndValues) })
```

```
IN-EntryInformation ::= EntryInformation
        (WITH COMPONENTS {
                ...,
                fromEntry           (TRUE),
                information         (WITH COMPONENTS {
                                        ...,
                                        attributeType               ABSENT}) OPTIONAL})

-- Operations, Arguments and Results definition

execute OPERATION ::= {
        ARGUMENT        IN-ExecuteArgument
        RESULT          ExecuteResult
        ERRORS          { attributeError | nameError |  in-ServiceError | referral |
                        securityError | updateError | executionError }
        CODE            opcode-execute }
-- Direction: SCF->SDF

IN-ExecuteArgument ::= ExecuteArgument
        (WITH COMPONENTS {
                ...,
                COMPONENTS OF       IN-CommonArguments })
-- Note that CommonArguments in ExecuteArgument in subclause 12.2 is replaced with IN-CommonArguments.

in-AddEntry  OPERATION ::= {
        ARGUMENT        IN-AddEntryArgument
        RESULT          AddEntryResult
        ERRORS          {nameError | in-ServiceError | securityError | attributeError | updateError
                        | referral}
        CODE            id-opcode-addEntry}
-- Direction: SCF->SDF

IN-AddEntryArgument ::= AddEntryArgument
        (WITH COMPONENTS {
                ...,
                COMPONENTS OF       IN-CommonArguments })
-- Note that CommonArguments in X.511 AddEntryArgument is replaced with IN-CommonArguments.

in-DirectoryBind  OPERATION ::= {
        ARGUMENT        DirectoryBindArgument
        RESULT          DirectoryBindResult
        ERRORS          {in-DirectoryBindError} }
-- Direction: SCF->SDF

in-ModifyEntry  OPERATION ::= {
        ARGUMENT        ModifyEntryArgument
        RESULT          ModifyEntryResult
        ERRORS          {nameError | in-ServiceError | securityError | attributeError | updateError
                        | referral}
        CODE            id-opcode-modifyEntry}
-- Direction: SCF->SDF
IN-ModifyEntryArgument ::= ModifyEntryArgument
        (WITH COMPONENTS {
                ...,
                selection           (IN-EntryInformationSelection),
                COMPONENTS OF       IN-CommonArguments })
-- Note that CommonArguments in X.511 ModifyEntryArgument is replaced with IN-CommonArguments.

IN-ModifyEntryResult ::= ModifyEntryResult
        (WITH COMPONENTS {
                ...,
                information   (WITH COMPONENTS {
                                        ...,
```

```
                        entry  (IN-EntryInformation)})}
-- The information is to be returned in the entry component of the information result.

in-RemoveEntry  OPERATION ::= {
        ARGUMENT        IN-RemoveEntryArgument
        RESULT          RemoveEntryResult
        ERRORS          {nameError | in-ServiceError | securityError | updateError | referral}
        CODE            id-opcode-removeEntry}
-- Direction: SCF->SDF

IN-RemoveEntryArgument ::= RemoveEntryArgument
        (WITH COMPONENTS {
                ...,
                COMPONENTS OF       IN-CommonArguments })
-- Note that CommonArguments in X.511 RemoveEntryArgument is replaced with IN-CommonArguments.

in-Search  OPERATION ::= {
        ARGUMENT        IN-SearchArgument
        RESULT          IN-SearchResult
        ERRORS          {nameError | in-ServiceError | securityError | attributeError | referral}
        CODE            id-opcode-search}
-- Direction: SCF->SDF

IN-SearchArgument ::= SearchArgument(
        WITH COMPONENTS {
                ...,
                searchAliases           (TRUE),
                selection               (IN-EntryInformationSelection),
                pagedResults            ABSENT,
                extendedFilter          ABSENT,
                COMPONENTS OF           IN-CommonArguments } )
-- Note that CommonArguments in X.511 SearchArgument is replaced with IN-CommonArguments.

IN-SearchResult ::= SearchResult
        (WITH COMPONENTS {
                ...,
                searchInfo          (WITH COMPONENTS {
                ...,
                entries         (WITH COMPONENT (IN-EntryInformation)),
                        partialOutcomeQualifier   (PartialOutcomeQualifier
                                (WITH COMPONENTS {
                                ...,
                                queryReference          ABSENT}))OPTIONAL})})

-- Errors definition

in-DirectoryBindError::= directoryBindError (WITH COMPONENTS{
        …,
        error (WITH COMPONENTS{
                serviceError(ServiceProblem(2)),
                securityError(SecurityProblem(1|2|7|10)) })})
-- SecurityProblem 10 indicates that the supplied SPKM token was found to be valid.
-- In reception, all the possible errors should be supported to understand a Bind error

in-ServiceError::=serviceError (WITH COMPONENTS{
        …,
        problem (ServiceProblem(1|2|3|4|5|6|8|9|10|11|12))})
--invalidQueryReference should not be sent because it is linked to the use of paged results.
-- The error code is the same as defined in X.511.

END
```