PHS MoU Document
B-IF4.28-02-TS

Title: **Public Personal Handy-phone System : Network-Network Interface for SCP Exchange Intelligent Network Inter-network Interface**

Version: 02

Date: May 15, 1998

PHS MoU Classification: Public

List of contents:

Number of pages: 94

PHS MoU Group

c/o Association of Radio Industries and Businesses (ARIB)
14F, Nittochi Bldg., 4-1, Kasumigaseki 1-choume, Chiyoda-ku, Tokyo 100, Japan
TEL +81-3-5510-8599 FAX +81-3-3592-1103

PHS MoU Document
B-IF4.28-02-TS

History of Revised Versions

Version	Date	Outline
01		Established
02	May 15,1998	

B-IF4.28-02-TS

Public Personal Handy-Phone System: Network - Network interface for SCP Exchange Intelligent Network Inter-network Interface

< Summary>

1. Relationship with International Standards

This Specifications specifies interface between public PHS networks based on the latest study on Q.1224 and Q1228 in ITU-T related to IN Capability Set 2 (CS-2), which was approved at ITU-T SG11
general meeting held in September 1997 and TTC Standards of the IN capability set (JT-Q1218), which is called as “CS-1(Refinement)” considering a consistency with the part of IN CS-2 studied in ITU-T.

2. Differences to/from International Recommendations

The description of this portion is not provided because the study results of IN-CS2 in ITU-T SG11 have not been officially published yet.

3. Others

3.1 References to ITU-T Recommendations

I.130 ,
Q.775,
Q.1200, Q.1201, Q.1204, Q.1205, Q.1208, Q.1211, Q.1214, Q.1218,
Q.1290,
X.25, X.200,
X.500, X.501, X.509, X.511), X.518, X.519, X.525 ,
X.680, X. 681, X.682, X.683, X.690, X.880

3.2 References to TTC Standards

JT-Q701, JT-Q711, JT-Q712, JT-Q713, JT-Q714,
JT-Q762, JT-Q763,
JT-Q771, JT-Q772, JT-Q773, JT-Q774
JT-Q932,
JT-Q1218
JT-X500

3.3 References to ISO Standards

IS 9545

4. Items for Further Study

None

(2) **B-IF4.28-01-TS**

Public Handy-Phone System:
Network-Network Interface for SCP Exchange
Intelligent Network Internetwork Interface

Contents

0. Introduction

This specification defines the internetwork INAP (Intelligent Network Application Protocol) required for support internetwork services. It supports interactions between the following two functional entities (FE's) (between SDF's only), as defined in the IN functional model. (Note: The names of two FEs below may be changed at internetwork INAP, for example, SCF (local), SDF (remote), or so.)

- Service Control Function (SCF)
- Service Data Function (SDF)

0.1 Definition methodology

The definition of the protocol can be split into three sections:

- the definition of the SACF/MACF rules for the protocol;
- the definition of the operations transferred between entities;
- the definition of the actions taken at each entity.

The SACF/MACF rules are defined in prose. The operation definitions are in Abstract Syntax Notation 1 (ASN.1, see ITU-T Recommendations X.208, X.680), and the actions are defined in terms of state transition diagrams.

The INAP is a ROSE user protocol (see ITU-T Recommendations X.219 and 229). The ROSE protocol is contained within the Component Sublayer of TCAP (see B-IF4.21 to B-IF4.25). At present, B-IF2.02 REGISTER, FACILITY and Call Control messages in DSS 1. Other supporting protocols may be added later.

The INAP (as a ROSE user) and the ROSE protocol have been specified using ASN.1(see ITU-T Recommendation X.680). The encoding of the resulting PDU's should use the Basic Encoding Rules (see ITU-T Recommendation X.690).

0.2 Example Physical Scenarios

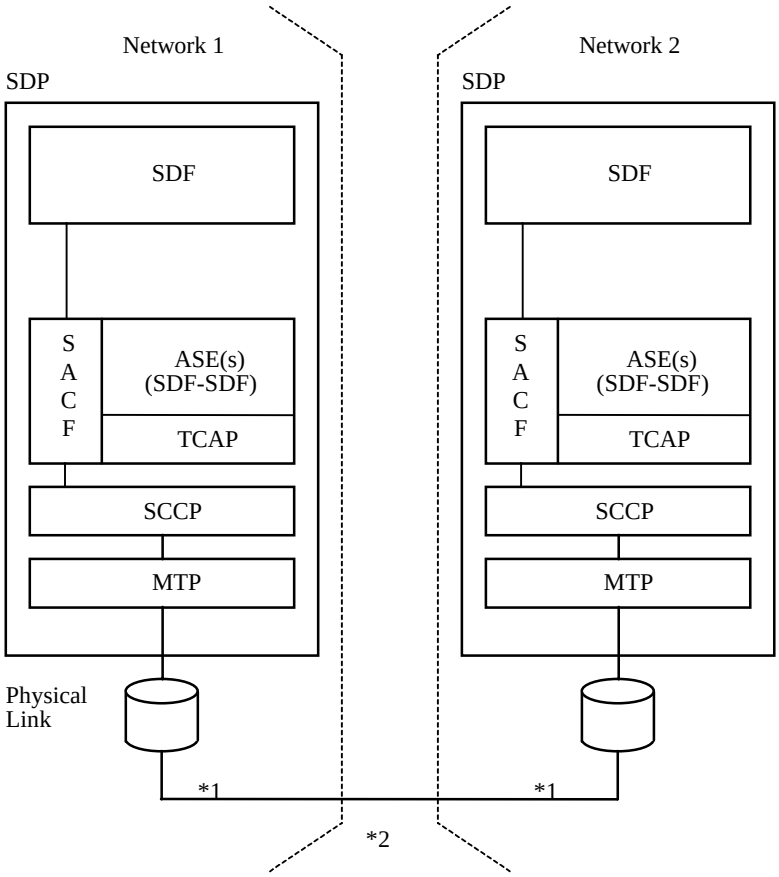
The protocol will support any mapping of functional to physical entities (PE's). It is the responsibility of network operators and equipment manufacturers to decide how to co-locate FE's to the best possible advantage as this may vary between manufacturers and between network operators. Therefore, the protocol is defined assuming the maximum distribution (i.e., one PE per FE).

The figures depicted in this subclause show how the INAP would be supported in an SS No. 7 network environment. This does not imply that only SS No. 7 may be used as the network protocol to support internetwork INAP.

The interface between SDF's will be INAP using TCAP which in turn, uses the services of the connectionless SCCP and MTP (see Figure 0-1 / B-IF4.28). The SDF is responsible for any interworking to other protocols to access other types of networks.

The TCAP, if it appears in the following figures, should be recognized to represent the TCAP function corresponding to a single interaction and transaction (rather than a single TCAP entity).

If the INAP message segmentation and reassembling are required on the inter-SDF interface (and on another interface if necessary) due to the message length, the segmentation and reassembling procedures for SCCP connectionless messages defined by B-IF4.14 should be used.



*1: In each network, messages may be transferred by using SS No. 7 network.
*2: In internetwork interaction, it may have a common SS No. 7 network.

Figure 0-1 / B-IF4.28 (ITU-T Q.1218)•
Physical Interface Between the SDP and SDP (for Internetwork INAP)

Terminology of Figure 0-1 / B-IF4.28 and Figure 0-2 / B-IF4.28:

SACF : Single association control function
SAO : Single association object
ASE : Application service element
INAP : Intelligent network application protocol

Note: Internetwork INAP is the collection of specifications of ASE's corresponding to inter-SDF interface of all IN ASE's.

0.3 INAP protocol architecture

Many of the terms used in this clause are based on the OSI Application Layer Structure as defined in ISO IS-9545.

The internetwork INAP protocol architecture can be illustrated as shown in Figure 0-2 / B-IF4.28.

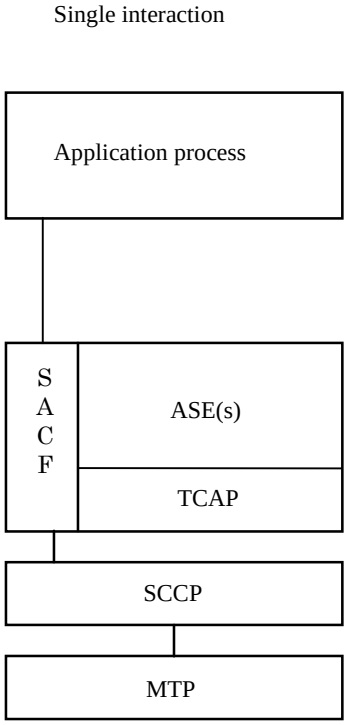


Figure 0-2 / B-IF4.28 (ITU-T Q.1218) INAP protocol Architecture

SACF provides the co-ordination function in using ASE's, which includes the ordering of operations supported by ASE(s) (based on the order of received primitives). The SAO represents the SACF plus a set of ASE's to be used over a single interaction between a pair of PE's.

Each ASE supports one or more operations. Description of each operation is tied with the action of corresponding FE modeling. Each operation is specified using the OPERATION macro described in Figure 0-3 / B-IF4.28.

The use of application context negotiation mechanism (as defined in B-IF4.21 - B-IF4.25) allows the two communicating entities to identify exactly what their capabilities are and also what the capabilities required on the interface should be. This should be used to allow evolution through capability set.

If the indication of a specific application context is not supported by a pair of communicating FE's, some mechanism to pre-arrange the context must be supported.

This specification specifies dialogue control method and application context negotiation mechanism for internetwork INAP. The implementation of inter-SDF interaction control in TCAP (including

application context negotiation) is specified in Clause 1.

0.3.1 INAP signaling congestion control for SS No. 7

The same type of procedure as defined for ISDN User Part signaling congestion control, defined by 2.11./ITU-T Q.767, may be used to the INAP procedures for signalling congestion control, i.e. on receipt of N-PCSTATE indication primitive with the information "signalling point congested" from SCCP, the INAP shall reduce the traffic load into the affected direction in several steps.

The above procedure may only apply to traffic which uses MTP Point Code addressing in the affected direction.

0.4 Internetwork INAP addressing

SCCP Global Title and MTP Point Code addressing (see B-IF4.1x and B-IF4.0x series) ensure that PDU's reach their physical destination (i.e., the correct point code) regardless of which network it is in.

Within a node, it is the choice of the network operator / manufacturer as to which SSN (Subsystem number) or SSNs are assigned to INAP.

Regardless of the above, any addressing scheme supported by the SCCP may be used.

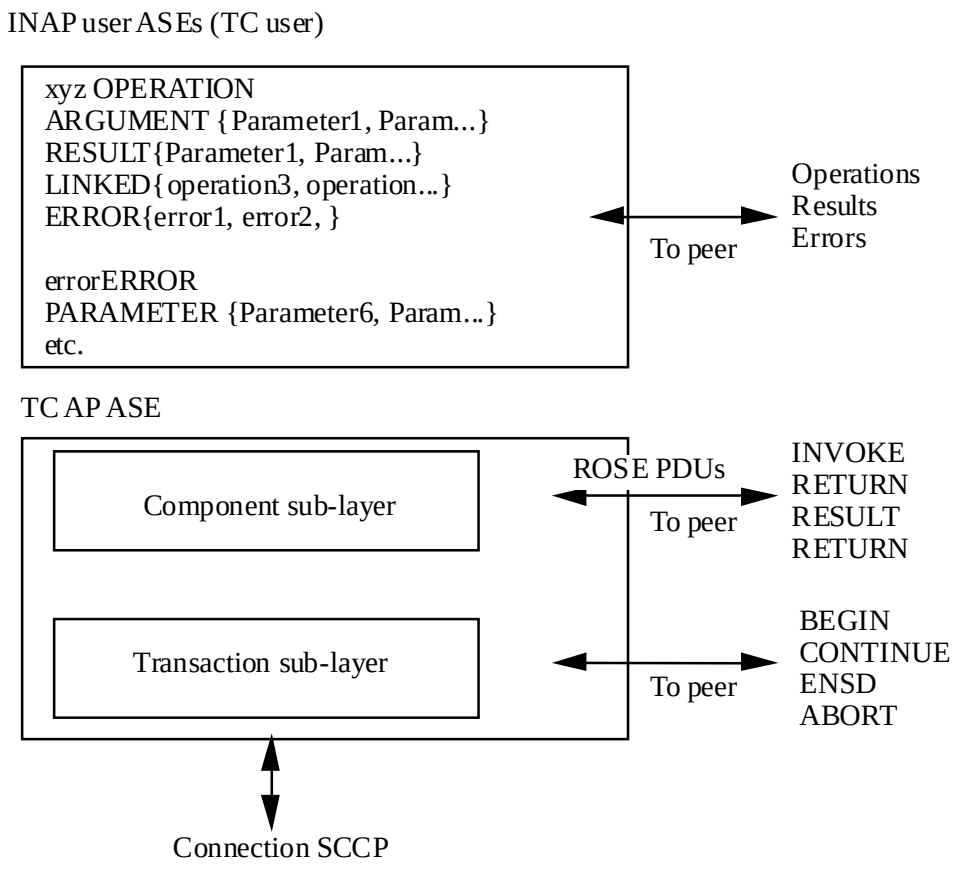


Figure 0-3 / B-IF4.28 (ITU-T Q.1218) Operation description (for TCAP)

0.5 Compatibility mechanism used for INAP

0.5.1 Introduction

This subclause specifies the compatibility mechanisms that shall be used to ensure consistent future versions of INAP.

There’re three categories of compatibility:

- Minor changes to INAP in future standardized versions:

  A minor change can be defined as a change of a functionality which is not essential for the requested IN service. In case it is a modification of an existing function, it is acceptable that the addressed function is executed in either the older or the modified variant. If the change is purely additional, it is acceptable that it is not executed at all and that the peer Application Entity (AE) need not know about the effect of the change. For minor changes, a new AC is not required.

  Major changes to INAP in future standardized versions:

  A major change can be defined as a change of a functionality which is essential for the requested IN service. In case it is a modification of an existing function, both application entities shall have a shared knowledge about the addressed functional variant. If the change is purely additional, the requested IN service will not be provided if one of the application entities does not support the additional functionality. For major changes, a new AC is required.

- Network-specific changes to INAP:

  These additions may be of either the major or minor type for a service. No new AC is expected to be defined for this type of change. At the time of definition, the additions would not be expected to be included in identical form in future versions of ITU-T Recommendations.

0.5.2 Definition of INAP compatibility mechanisms

0.5.2.1 Procedures for major additions to INAP

In order to support the introduction of major functional changes, the protocol allows a synchronization between the two applications with regard to which functionality is to be performed. This synchronization takes place before the new function is invoked in either application entity, in order to avoid complicated fall-back procedures. The solution chosen to achieve such a synchronization is to use the AC negotiation procedures provided in ITU-T Recommendation Q.773.

0.5.2.2 Procedures for minor additions to INAP

The extension mechanism marker shall be used for future standardized minor additions to INAP. This mechanism implements extensions differently by including an “extensions maker” in the type

definition. The extensions are expressed by optional fields that are placed after the maker. When an entity receives unrecognized parameters that occur after the maker, they are ignored (see ITU-T recommendation X.68x).

0.5.2.3 Procedures for inclusion of network specific additions to INAP

This mechanism is based on the ability to explicitly declare fields of any type via the Macro facility in ASN.1 at the outermost level of a type definition. It works by defining an “Extension field” that is placed at the end of the type definition. This extension field is defined as a set of extensions, where an extension can contain any type. Each extension is associated with a value that defines whether the terminating node should ignore the field if unrecognized, or reject the message, similar to the comprehension required mechanism described in the previous subclause. Refer to ITU-T Recommendation Q.1400 for a definition of this mechanism.

0.6 SACF/MACF rules

0.6.1 Reflection of TCAP AC

TCAP application context negotiation rules require that the proposed AC, if acceptable, is reflected in the first backwards message.

If the AC is not acceptable and the TC-user does not wish to continue the dialogue, it may provide an alternate AC to the initiator which can be used to start a new dialogue.

TCAP AC negotiation applies only to the SCF interfaces.

Refer to the B-IF4.2x Series for a more detailed description of the TCAP AC negotiation mechanism.

0.6.2 Sequential / parallel execution of operations

In some cases it may be necessary to distinguish whether operations should be performed sequentially or in parallel (synchronized). Operations which may be synchronized are:

- charging operations may be synchronized with any other operation.

The method of indication that operations are to be synchronized is to include them in the same message. Where one of the operations identified above must not be executed until some other operation has progressed to some extent or finished, the sending PE (usually) SCP) can control this by sending the operations in two separate messages.

This method does not imply that all operations sent in the same message should be executed simultaneously, but simply that where it could make sense to do so (in the situations identified above) the operations should be synchronized.

In case of inconsistency between the above-mentioned generic rules and the FE-specific riles, as specified in clause 1, the FE-specific rules take precedence over the generic rules.

1. Relationships between FEs

1.1 General

This clause describes the Information Flows (IFs) between Functional Entities (FEs) relating to inter-network environment.

1.2 Information Flows between FEs

Information flows between two FEs either consist of a request/response pair or of a request alone. Note that information flows may not map one to one on to signaling messages between the corresponding physical entities in the physical plane. The complete set of IFs between two FEs defines the relationship between those FEs. Where necessary, specific information flows have been identifies to cancel the effect of other information flows.

Note that IFs relating to error conditions are not described.

1.3 SDF - SDF relationship

1.3.1 General

The purpose of the SDF-SDF relationship is twofold. It should provide both data location transparency and efficiency of data distribution.

These two usages of the SDF-SDF relationship correspond to two types of operations. First, when hiding the data distribution, an operation sent to an SDF could be forwarded to another SDF which holds the relevant data by using the SDF-SDF interface. In this case, the procedures over the SDF-SDF interface used are for the chaining of SCF-SDF procedures. When the SDF-SDF relationship is used to manage the data distribution and to improve the overall efficiency of the data procedures, the SDF-SDF interface is used to transfer data from a SDF to another SDF. In this case, the procedures are transfer and copy procedures.

1.3.2 Information flows between the SDF and SDF

1.3.2.1 Authenticate

a. FE Relationship: SDF to SDF
b. Synopsis:

This procedure is used to have identification and authentication of two SDFs involved in an SDF-SDF relationship. This procedure is prior to any procedure on the SDF-SDF interface. It is used to enforce access control policy between databases.

c. Information Elements

Authentication Information	(M)
Authorized Relationship ID	(M)

d. Mapping to FE Model(s)

The initiating SDF sends this information flow to an SDF with which it is interacting in order to perform authentication.

1.3.2.2 Authenticate Result

a. FE Relationship: SDF to SDF
b. Synopsis

This procedure is used to confirm the result of authentication by the interacting SDF.

c. Information Elements

Authentication Information	(M)
Authorized Relationship ID	(M)

d. Mapping to FE Model(s)

The responding SDF sends this information flow to an SDF with which it is interacting in order to perform authentication.

1.3.2.3 Chaining Request

a. FE Relationship: SDF to SDF
b. Synopsis:

This procedure is used to pass from the invoking SDF to the responding SDF a database request that cannot be fulfilled by the invoking SDF. The responding SDF is considered by the invoking SDF as the SDF holding the data needed to perform the database request. Indications contained in the database of the invoking SDF motivate the sending of the chaining procedure to the responding SDF.

c. Information Elements

Authorized Relationship ID	(M)
Chained Argument	(M)
Security Parameters	(M)

d. Mapping to FE Model(s)

The initiating SDF sends this information flow to an SDF with which it is interacting in order to perform the chained operation. This would include indication of the manner in which authentication has been carried out between the SDFs.

1.3.2.4 Chaining Result

a. FE Relationship: SDF to SDF
b. Synopsis:

This procedure is used to pass from the responding SDF to the invoking SDF the results of the request for performing a database request that cannot be fulfilled by the invoking SDF.

c. Information Elements

Authorized Relationship ID	(M)
Chained Result	(M)
Security Parameters	(M)

d. Mapping to FE Model(s)

The responding SDF sends this information flow to the initiating SDF with which it is interacting in order to perform the chained operation.

1.3.2.5 Copy Request

a. FE Relationship: SDF to SDF
b. Synopsis:

This procedure is used to copy data contained in the responding SDF to the invoking SDF. The copy (or a part of it) can be refreshed regularly or not at all. This maintenance of the copy is done through the use of the Update Copy procedure. The need for a Copy procedure originates from network events or service events. This procedure puts indications in the invoking SDF about the location of the master copy.

c. Information Elements

Authorized Relationship ID	(M)
Maintained Part	(M)
Master	(M)
Replication Area	(M)
Update Mode	(M)
Update Strategy	(M)

d. Mapping to FE Model(s)

The invoking SDF sends this information flow to the responding SDF in order to request a copy.

1.3.2.6 Copy Result

a. FE Relationship: SDF to SDF
b. Synopsis:

This procedure is used to provide the copy data contained in the responding SDF to the invoking SDF.

c. Information Elements

Authorized Relationship ID (M)
Replicated Data (M)

d. Mapping to FE Model(s)

The responding SDF sends this information flow to the invoking SDF in order to provide a copy.

1.3.2.7 End Authenticate

a. FE Relationship: SDF to SDF
b. Synopsis:

This IF is issued by the SDF to end an authenticated relationship between two SDFs.

c. Information Elements

Authorized Relationship ID (M)

d. Mapping to FE Model(s)

The invoking SDF sends the information flow to an SDF to end a previously authenticated relationship.

1.3.2.8 Update Copy

a. FE Relationship: SDF to SDF
b. Synopsis:

This IF is used to maintain a copy contained in the SDF to which a copy was originally provided because the selected update mode indicates that an update of the copy should be sent (e.g. modification of the copy in the responding network). The format of the information to be sent follows the indications contained in the information element update strategy of the Copy procedure.

c. Information Elements

Authorized Relationship ID (M)
Refreshed Information (M)

d. Mapping to FE Model(s)

The invoking SDF sends this information flow to an SDF to which a copy was previously provided in order to update the copy.

1.3.2.9 Update Copy Result

a. FE Relationship: SDF to SDF
b. Synopsis:

This IF confirms results of the Update Copy Request.

c. Information Elements

Authorized Relationship ID (M)

d. Mapping to FE Model(s)

The SDF to which a copy update was provided sends this IF to confirm results of the UpdateCopy.

1.3.3 IE Description for the SDF-SDF information flows

1.3.3.1 Authentication Information

This IE, if being used with Authenticate IF, contains information needed to perform the required type of authentication. No information might be needed.

This IE, if being used with Authenticate Result IF, contains information used in providing results of the authentication procedure, as previously defined.

1.3.3.2 Authorized Relationship ID

Identifies the established Authorized Relationship between two FEs through which operations can be applied. At the physical plane for IN CS-2, it is mapped on to a TCAP transaction identity.

1.3.3.3 Chained Argument

This IE contains the argument of a database operation of the SCF-SCF interface as defined in CS-1. That is the argument of the forwarded operation.

1.3.3.4 Chained Result

This IE gives the result of the chained request.

1.3.3.5 Maintained Part

This IE describes the part of the copy that should be regularly maintained.

1.3.3.6 Master

This IE gives the name of the master. It is used to know the origin of the copy.

1.3.3.7 Refreshed Information

This IE contains the update of the copy according to the format specified in the information element update strategy of the Copy procedure.

1.3.3.8 Replication Area

This IE describes the part of the database of the responding SDF that should be replicated in the database of the invoking SDF.

1.3.3.9 Replicated Data

This IE contains the copy that should be placed in the invoking network.

1.3.3.10 Security Parameters

This IE, if being used with Chaining Request govern the operation of various security features associated with the directory operation.

This IE, if being used with Chaining Result govern the operation of various security features associated with the directory operation. This would include indication of the manner in which authentication has been carried out between the SDFs.

1.3.3.11 Update Mode

This IE indicates when the copy should be updated (on changes or periodically) and who will take the initiative of updating it.

1.3.3.12 Update Strategy

This IE indicates whether the full copy should be returned or only the changes made to it.

2. SDF/SDF Interface

2.1 Introduction to the IN X.500 DSP and DISP subset

The purpose of the SDF/SDF interface is to allow the transfer of copies of service profiles from one SDF to another and to manage the copies within the database network. The X.500 functionalities cover more than the functionalities needed to fulfill the CS2 requirements. This clause tries to indicate which aspects of the DSP and DISP should be considered and supported and which should be left out or ignored. Profiling is used as a means to present the status of the different parameters.

It is important to mention that the number of parameters carried in a message should be minimized, to reduce the load on the signalling traffic and processing time. This is the reason why the parameters are removed unless they are absolutely necessary when they are sent. On reception removed parameters should not be treated but should be understood by the receiving entity. This allows the extension of the profile in the future according to its actual description in the 1993 edition of the Directory.

For convenience and clarity this profile is defined using ASN.1 subtyping facilities however these definitions do not form a protocol specification. This simply indicates which parameters an implementation should not send. It does not change the behavior of the receiving entity which shall still be capable of decoding values which conform to the original definition of the DSP and DISP. Nevertheless elements that are excluded by subtyping should be understood but not treated.

2.2 Working Assumptions

Several assumptions were used to design the DSP and DISP for IN CS2. They are as follows:

- Assumption 1: The agreements between network operators concerning the transfer of data are defined off-line (e.g. management operations). The establishOperationalBinding operation is only used to activate an agreement.

- Assumption 2: The agreements cannot be modified by an on-line operation.

- Assumption 3: The terminateOperationalBinding operation is used to end an agreement between two network operators. This means that the copy held by the shadow-consumer is no longer maintained. It should not be used and should be deleted. However the agreement could be required for future associations between the two networks, therefore this information should be retained.

- Assumption 4: The shadow updates are initiated by the shadow supplier who holds the master copy. Therefore modifications of the copies are not performed on the shadowed copies but only on the master copy. The modification requests are passed to the master copy by using a chained operation. Copies are updated on changes.

- Assumption 5: Only direct references are used in DSAs. Operations can only be chained once. If the operation cannot be fulfilled after one chaining, a referral should be sent back.

- Assumption 6: It is not possible to make a copy of a copy. One should refer to the master copy to get

a copy.

- Assumption 7: The shadowing mechanism is initiated by a specific DAP operation or by a management operation. The management operation is for further study.

- Assumption 8: The time when a shadowing agreement is terminated depends on the type of service. In most cases it will be based on the number of copies. Once the maximum number of copies is reached for a part of a DIT, then the oldest copy has to be deleted and its agreement de-activated. The maximum number of copies can be equal to one.

- Assumption 9: An SDF-SDF operation cannot be abandoned. If an operation takes too much time, its timer expires and there is no need to abandon it.

2.3 The SDF Information Model

The ITU-T Recommendation X.501 provides a generic information model that is needed to support the service provided by the Directory. In the context of IN, the generic information model should conform to the section 1 to section 7 of the ITU-T Recommendation X.501. However certain aspects of this recommendation need not to be supported. This includes the DIT content rules whose use is a local matter.

Some other points are outside the scope of this recommendation. This concerns the items associated with capabilities not covered by IN CS-2. Therefore the following parts of ITU-T Recommendation X.501 are not applicable:

paragraphs f), h) and i) in 16.2.3/ITU-T Recommendation X.501.

paragraph a) in 16.2.4/ ITU-T Recommendation X.501. The compare operation is not used, the search operation is used instead. Therefore the FilterMatch permission replaces the Compare permission.

2.3.1 Information Framework

The IN defines a number of extensions to the ITU-T Recommendation X.501 information framework in order to meet IN service requirements. Only the enhancements are defined in these clauses. Unless stated the definition of other elements are the same as for ITU-T Recommendation X.501 version 3.

2.3.1.1 METHOD

Each method represents a sequence of DAP operations which are performed under the control of the DSA. The DUA is responsible for providing all necessary information in order for the DSA to complete the method. The DSA is responsible for collecting all information to be returned to the DUA.

For documentation purposes, it is suggested to add a description field to the class definition.

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 ::= 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}
```

2.4 The IN X.500 DISP subset

2.4.1 Shadowing Agreement Specification

The Shadowing agreement is specified as:

```
IN-ShadowingAgreementInfo ::= ShadowingAgreementInfo
        (WITH COMPONENTS {
                ...,
                master                  ABSENT,
                secondaryShadows        ABSENT})
```

shadowSubject specifies the subtree, entries and attributes to shadow. The components of UnitOfReplication are defined in 9.2/ITU-T Recommendation X.525.

updateMode specifies when updates of a shadowed area are scheduled to occur. The components of UpdateMode are defined in 9.3/ITU-T Recommendation X.525.

master contains the access point of the DSA containing the mastered area. "As this information is already known by the DSA it is not required for IN."

secondaryShadows permits secondary shadow information to be subsequently supplied to the shadow

supplier. The secondary shadows are ignored in the IN context (assumption 5), then this component should not be included.

2.4.2 DSA Shadow Bind

A dSAShadowBind operation is used at the beginning of a period of providing shadows.

in-dSAShadowBind OPERATION ::= in-DirectoryBind

IN CS2 uses the dSAShadowBind operation defined by an in-DirectoryBind operation. The in-DirectoryBind operation is specified as follows.

```
in-DirectoryBind  OPERATION ::= {
        ARGUMENT    DirectoryBindArgument
        RESULT   DirectoryBindResult
        ERRORS  {in-DirectoryBindError}
        CODE      in-opcode-in-bind}
```

The DirectoryBind and DirectoryUnbind operations are used at the beginning and end of a particular period of accessing the Directory.

The status of each parameter of DirectoryBindArgument is as follows:

- the credentials parameter is used to establish the identity of the user. It is necessary for services that requires authentication.

- the versions parameter identifies the version of the DirectoryService which is to be used. Since only one version is supported by IN CS-2, this parameter needs not to be sent. When it is received with the v1 not supported, a service error should be returned

The same considerations apply to the DirectoryBindResult.

```
IN-DirectoryBindError ::= DirectoryBindError
    (WITH COMPONENTS {
        ...,
        error   (WITH COMPONENTS {
                                securityError
(SecurityProblem (1|2|7|10)),
                                serviceError
(ServiceProblem (2))})})
```

SecurityProblem 10 indicates that the supplied SPKM token was found to be invalid.
In reception, all the possible errors should be supported to understand a Bind error.

2.4.3 IN-DSA Shadow Unbind

The **i**n-DSAShadowUnbind operation replaces the ITU-T Recommendation X.525

dSAShadowUnbind operation to provide class 4 operation behaviour for unbind procedures.

```
in-DSAShadowUnbind  OPERATION ::= inEmptyUnbind

inEmptyUnbind OPERATION ::= {
    RETURN RESULT   FALSE
    ALWAYS RESPONDS FALSE }
```

2.4.4 Coordinate Shadow Update

The inCoordinateShadowUpdate operation is used by the shadow supplier to indicate the shadowing agreement for which it intends to send updates.

```
inCoordinateShadowUpdate  OPERATION ::= {
        ARGUMENT            IN-CoordinateShadowUpdateArgument
        RESULT              IN-CoordinateShadowUpdateResult
        ERRORS              {shadowError}
        CODE                id-opcode-coordinateShadowUpdate}

IN-CoordinateShadowUpdateArgument ::= CoordinateShadowUpdateArgument (
        WITH COMPONENTS {
                ...,
                updateStrategy  (standard:{total | incremental})})

IN-CoordinateShadowUpdateResult ::=  CoordinateShadowUpdateResult(
    WITH COMPONENTS {
            ...,
            null            PRESENT})
```

The various parameters have the meanings defined below:

a) The agreementID argument identifies the shadowing agreement.

b) The lastUpdate argument indicates the shadow supplier's understanding of the time at which the last update for this agreement was sent and is the time as provided by the shadow supplier DSA. This argument may only be omitted in the first instance of either an inCoordinateShadowUpdate operation or an inRequestShadowUpdate operation for a particular shadowing agreement

c) The updateStrategy argument identifies the update strategy the shadow supplier intends to use for this update. For IN CS2, a total or incremental replacement strategy should be used. The "NoChanges" option will not be used.

d) The securityParameters argument is defined in 7.10/ITU-T Rec. X.511 | ISO/IEC 9594-3.

2.4.5 Update Shadow

An inUpdateShadow operation is invoked by the shadow supplier to send updates to the shadow consumer for a unit of replication. Prior to this operation being initiated, an

inCoordinateShadowUpdate operation or an inRequestShadowUpdate operation must have been successfully completed for the identified shadowing agreement.

```
inUpdateShadow  OPERATION ::= {
        ARGUMENT            IN-UpdateShadowArgument
        RESULT              IN-UpdateShadowResult
        ERRORS              {shadowError}
        CODE                id-opcode-updateShadow}

IN-UpdateShadowArgument ::= UpdateShadowArgument (
        WITH COMPONENTS {
                ...,
                updatedInfo             (IN-RefreshInformation)})

IN-UpdateShadowResult ::=   UpdateShadowUpdateResult(
WITH COMPONENTS {
        ...,
        null            PRESENT})
```

The various parameters have the meanings as defined below:

a) The agreementID argument identifies the shadowing agreement that has been established.

b) The updateTime argument is supplied by the shadow supplier. This time is used during the next inCoordinateShadowUpdate or inRequestShadowUpdate to ensure that the shadow supplier and shadow consumer have a common view of the shadowed information.

c) The updateWindow argument, when present, indicates the next window during which the shadow supplier expects to send an update.

d) The updatedInfo argument provides the information required by the shadow consumer to update its shadowed information. The semantics of the information conveyed in this parameter shall result in the shadow consumer reflecting the changes supplied.

e) The securityParameters argument is defined in 7.10/ITU-T Rec. X.511 | ISO/IEC 9594-3.

```
IN-RefreshInformation ::= RefreshInformation (
        WITH COMPONENTS {
        ...,
        otherStrategy   ABSENT})
```

The various parameters have the meanings as defined below:

a) noRefresh indicates that there have been no changes to the shadowed information from the previous instance to the present. This may be used where an updateShadow operation must be supplied at a certain interval defined in the shadowing agreement (updateMode), but no modification has actually occurred.

b) total provides a new instance of the shadowed information. The incremental strategy should be preferably used because it saves signalling.

c) incremental provides, instead of a complete replacement of the shadowed information, only the changes which have occurred to that shadowed information between lastUpdate in the most recent inCoordinateShadowUpdate (or inRequestShadowUpdate) request and updateTime in the current inUpdateShadow request (or requestShadowUpdate response).

d) otherStrategy provides the ability to send updates by mechanisms outside the scope of the Directory Specification. For IN CS2, either a total or incremental strategy should be used.

2.4.6 Request Shadow Update

An inRequestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier.

```
inRequestShadowUpdate  OPERATION ::= {
        ARGUMENT        IN-RequestShadowUpdateArgument
        RESULT          IN-RequestShadowUpdateResult
        ERRORS          {shadowError}
        CODE            id-opcode-requestShadowUpdate}

IN-RequestShadowUpdateArgument ::= RequestShadowUpdateArgument (
        WITH COMPONENTS {
                ...,
                requestStrategy (standard:{incremental | total})})

    IN-RequestShadowUpdateResult ::= RequestShadowUpdate Result(
    WITH COMPONENTS {
        ...,
        null    PRESENT})
```

The various parameters have the meanings as defined below:

a) The agreementID argument identifies the shadowing agreement.

b) The lastUpdate argument is the time provided by the shadow supplier in the most recent successful update. This argument may only be omitted in the first instance of either an inCoordinateShadowUpdate operation or an inRequestShadowUpdate operation for a particular shadowing agreement.

c) The requestedStrategy argument identifies the type of update being requested by the shadow consumer. The shadow consumer may request either an incremental or a total update from the shadow supplier.

d) The securityParameters argument is defined in 7.10/ITU-T Rec. X.511 | ISO/IEC 9594-3.

2.5 The IN X.500 DSP subset

2.5.1 Information Types and Common Procedures

2.5.1.1 Chaining Arguments

The ChainingArguments are present in each chained operation, to convey to a DSA the information needed to successfully perform its part of the overall task:

```
IN-ChainingArguments ::= ChainingArguments (
        WITH COMPONENTS {
                ...,
                aliasDereferenced           ABSENT,
                aliasedRDNs                 ABSENT,
                returnCrossRefs             ABSENT,
                info                        ABSENT,
                timeLimit                   ABSENT,
                entryOnly                   ABSENT,
                exclusions                  ABSENT,
                excludeShadows              ABSENT,
                nameResolveOnMaster         ABSENT})
```

The various components have the meanings as defined below:

a) The originator component conveys the name of the originator of the request unless already specified in the security parameters. If requester is present in CommonArguments, this argument may be omitted.

b) The targetObject component conveys the name of the object whose directory entry is being routed to. The role of this object depends on the particular operation concerned: it may be the object whose entry is to be operated on, or which is to be the base object for a request or sub request involving multiple objects (e.g., ChainedModify). This component can be omitted only if it has the same value as the object or base object parameter in the chained operation, in which case its implied value is that value.

c) The operationProgress component is used to inform the DSA of the progress of the operation, and hence of the role which it is expected to play in its overall performance. Even though direct knowledge references are assumed, this parameter is deemed applicable for IN CS2 since an SDF to which an operation is chained can still respond with a continuation reference in the chained operation dsaReferral error.

d) The traceInformation component is used to prevent looping among DSAs when chaining is in operation. A DSA adds a new element to trace information prior to chaining an operation to another DSA. On being requested to perform an operation, a DSA checks, by examination of the trace information, that the operation has not formed a loop.

e) The aliasDereferenced component is a boolean value which is used to indicate whether or not one or more alias entries have so far been encountered and dereferenced during the course of

distributed name resolution. Since alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed, there is no need for this indicator.

f) The aliasedRDNs component indicates how many of the RDNs in the targetObject name have been generated from the aliasedEntryName attributes of one (or more) alias entries. The integer value is set whenever an alias entry is encountered and dereferenced. Since alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed, there is no need for this indicator.

g) The returnCrossRefs component is a Boolean value which indicates whether or not knowledge references, used during the course of performing a distributed operation, are requested to be passed back to the initial DSA as cross references, along with a result or referral. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

h) The referenceType component indicates, to the DSA being asked to perform the operation, what type of knowledge was used to route the request to it. The DSA may therefore be able to detect errors in the knowledge held by the invoker. If such an error is detected it shall be indicated by a ServiceError with the invalidReference problem. ReferenceType is described fully in 2.5.1.3.

i) The info component is used to convey DMD-specific (Directory Management Domain) information among DSAs which are involved in the processing of a common request. As the management protocols are not addressed in CS2, this parameter is deemed to be not applicable.

j) The timeLimit component, if present, indicates the time by which the operation is to be completed. It is redundant with operation timers of TCAP and is therefore not needed.

k) The securityParameters component is specified in ITU-T Rec. X.511 | ISO/IEC 9594-3.

l) The entryOnly component is set to TRUE if the original operation was a search, with the subset argument set to oneLevel and an alias entry was encountered as an immediate subordinate of the baseObject. The DSA which successfully performs name resolution on the targetObject name, shall perform object evaluation on only the named entry. The only chained operations for CS1(refinement) are ModifyEntry operations and therefore this parameter is not needed.

m) The uniqueIdentifier component is optionally supplied when it is required to confirm the originator name (the originator is the SDF forwarding the request). The uniqueIdentifier element is described in ITU-T Rec. X.501 | ISO/IEC 9594-2.

n) The authenticationLevel component is optionally supplied when it is required to indicate the manner in which authentication has been carried out between the SDFs. The AuthenticationLevel element is described in ITU-T Rec. X.501 | ISO/IEC 9594-2.

o) The exclusions component has significance only for Search operations; it indicates, if present, which subtrees of entries subordinate to the targetObject shall be excluded from the result of the Search operation. The only chained operations for CS2 are ModifyEntry operations and therefore this parameter is not needed.

p) The excludeShadows component has significance only for Search and List operations; it indicates that the search shall be applied to entries and not to entry copies. This optional component may be used by a DSA as one way to avoid the receipt of duplicate results. Since direct knowledge references are assumed, this parameter is deemed not applicable for CS2.

q) The nameResolveOnMaster component only has significance during name resolution, and is only set if NSSRs (non-specific knowledge references) have been encountered. If set to TRUE, it signals that subsequent name resolution, i.e., matching the remaining RDNs from nextRDNToBeResolved, shall not employ entry copy information; subsequent resolution of each remaining RDN shall be done in the master DSA for the entry identified by that RDN. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS-2.

2.5.1.2 Chaining Results

The ChainingResults are present in the result of each operation and provide feedback to the DSA which invoked the operation.

```
IN-ChainingResults ::= ChainingResults (
        WITH COMPONENTS {
                ...,
                info                ABSENT,
                crossReferences     ABSENT,
                alreadySearched     ABSENT})
```

The various components have the meanings as defined below:

a) The info component is used to convey DMD-specific information among DSAs which are involved in the processing of a common request. As the management protocols are not addressed in CS2, this parameter is deemed to be not applicable.

b) The crossReferences component is not present in the ChainingResults unless the returnCrossRefs Tcomponent of the corresponding request had the value TRUE. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

c) The securityParameters component is specified in ITU-T Rec. X.511 | ISO/IEC 9594-3. Its absence is deemed equivalent to there being an empty set of security parameters.

d) The alreadySearched component, if present, indicates which subordinate RDNs immediately subordinate to the targetObject have been processed as a part of a chained Search operation and therefore shall be excluded in a subsequent request. The only chained operations for CS1(refinement) are ModifyEntry operations and therefore this parameter is not needed.

2.5.1.3 Reference Type

A ReferenceType value indicates one of the various kinds of reference defined in ITU-T Rec. X.501 | ISO/IEC 9594-2.

```
IN-ReferenceType ::= ReferenceType (1|2|4|5|6|7|8)
```

Value (3)(cross reference) is not applicable for IN CS2 as direct references are assumed.

2.5.1.4 Access Point Information

There are three types of access points:

a) An AccessPoint value identifies a particular point at which access to the Directory, specifically to a DSA, can occur. The access point has a Name, that of the DSA concerned, and a PresentationAddress, to be used in SS7 signalling to that DSA.

```
IN-AccessPoint ::= AccessPoint (
        WITH COMPONENTS {
                ...,
                protocolInformation      ABSENT})
```

The address contains the network address of the DSA in the SS7.

b) A MasterOrShadowAccessPoint value identifies an access point to the Directory. The category, either master or shadow, of the access point is dependent upon whether it points to a naming context or commonly usable replicated area.

```
IN-MasterOrShadowAccessPoint ::= MasterOrShadowAccessPoint (
        WITH COMPONENTS {
                ...,
                COMPONENTS OF IN-AccessPoint})
```

c) A MasterAndShadowAccessPoints value identifies a set of access points to the Directory, i.e., a set of related DSAs. These access points share the property that each refers to a DSA holding entry information from a common naming context or a common set of naming contexts mastered in one DSA when the value is a value of the nonSpecificKnowledge attribute. A MasterAndShadowAccessPoints value indicates the category of each AccessPoint value it contains. The access point of the master DSA of the naming context need not be included in the set.

```
IN-MasterAndShadowAccessPoints ::= SET OF MasterOrShadowAccessPoint
```

An AccessPointInformation value identifies one or more access points to the Directory.

```
IN-AccessPointInformation ::= AccessPointInformation (
        WITH COMPONENTS {
                ...,
                COMPONENTS OF IN-MasterOrShadowAccessPoint})
```

2.5.1.5 Continuation Reference

A ContinuationReference describes how the performance of all or part of an operation can be continued at a different DSA or DSAs. It is typically returned as a referral when the DSA involved is unable or unwilling to propagate the request itself.

```
IN-ContinuationReference ::= ContinuationReference (
        WITH COMPONENTS {
                ...,
                aliasedRDNs                ABSENT,
                rdnsResolved               ABSENT,
                referenceType              (IN-ReferenceType),
                accessPoints               SET OF (IN-AccessPoint),

                )
```

The various components have the meanings as defined below:

a) The targetObject name indicates the name which is proposed to be used in continuing the operation. This might be different from the targetObject name received on the incoming request if, for example, an alias has been dereferenced, or the base object in a search has been located.

b) The aliasedRDNs component indicates how many (if any) of the RDNs in the target object name have been produced by dereferencing an alias. Since alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed, there is no need for this indicator.

c) The operationProgress indicates the amount of name resolution which has been achieved, and which will govern the further performance of the operation by the DSAs named, should the DSA or DUA receiving the ContinuationReference wish to follow it up.

d) The rdnsResolved component value (which need only be present if some of the RDNs in the name have not been the subject of full name resolution, but have been assumed to be correct from a cross reference) indicates how many RDNs have actually been resolved, using internal references only. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

e) The referenceType component indicates what type of knowledge was used in generating this continuation.

f) The accessPoints component indicates the access points which are to be contacted to achieve this continuation. Only where non-specific subordinate references are involved can there be more than one AccessPointInformation item.

g) The entryOnly component is set to TRUE if the original operation was a search, with the subset argument set to oneLevel, and an alias entry was encountered as an immediate subordinate of the baseObject. The DSA which successfully performs name resolution on the targetObject name, shall perform object evaluation on only the named entry. Since alias entries in IN are just a means to provide an alternative name for an object and therefore should be dereferenced when needed, there is no need for this indicator.

h) The exclusions component identifies a set of subordinate naming contexts that should not be explored by the receiving DSA.

i) The returnToDUA element is optionally supplied when the DSA creating the continuation reference wishes to indicate that it is unwilling to return information via an intermediate DSA (e.g., for security reasons), and wishes to indicate that information may be directly available via an operation over DAP between the originating DUA and the DSA. When returnToDUA is set to TRUE, referenceType may be set to self. This element may be used in IN for support of the shadowing agreement established between network operators (e.g., SDFv to SDFh Modify may fail based upon access control restrictions).

j) The nameResolveOnMaster element is optionally supplied when the DSA creating the continuation reference has encountered NSSRs. Since direct knowledge references are assumed, this parameter is deemed not applicable for IN CS2.

2.5.2 DSA Bind

A dSABind operation is used to begin of a period of cooperation between two DSAs providing the Directory service.

```
dSABind OPERATION ::= in-DirectoryBind
```

IN CS2 uses the dSABind operation as specified in subclause 2.4.2.

2.5.3 IN DSA Unbind

The in-DSAUnbind operation replaces the ITU-T Rec. X.518 dSAUnbind operation to provide class 4 operation behaviour for unbind procedures.

```
in-DSAUnbind  OPERATION ::= inEmptyUnbind
```

2.5.4 Chained operations

A DSA, having received an operation from a DUA, may elect to construct a chained form of that operation to propagate to another DSA. For IN CS2 a DSA, having received a chained form of an operation, may not elect to chain it to another DSA.

The DSA invoking a chained form of an operation may optionally sign the argument of the operation; the DSA performing the operation, if so requested, may sign the result of the operation.

The chained form of an operation is specified using the parameterized type IN-chained {}.

```
IN-chained {OPERATION : operation} OPERATION ::= {
        ARGUMENT   OPTIONALLY-PROTECTED {SET{
                IN-chainedArgument              (IN-ChainingArguments),
                argument                [0]     operation.&ArgumentType}
                DIRQOP.&dspChainedOp-QOP@dirqop }
        RESULT      OPTIONALLY-PROTECTED {SET{
                IN-chainedResult                ABSENT,
                result                  [0]     operation.&ResultType},
                DIRQOP.&dspChainedOp-QOP@dirqop }
```

```
ERRORS      {operation.&Errors EXCEPT (referral | dsaReferral)}
CODE        operation.&code }
```

a) IN-chainedArgument. This is a value of ChainingArguments which contains that information, over and above the original DUA-supplied argument, which is needed in order for the performing DSA to carry out the operation.

b) argument. This is a value operation.&Argument and consists of the original DUA-supplied argument.

Should the request succeed, the result of the derived operation has the components:

a) IN-chainedResult. This is a value of IN-ChainingResults which contains that information, over and above that to be supplied to the originating DUA, which may be needed by the previous DSAs in a chain. For IN CS2, it is assumed that chains are not greater than length one, therefore the need of this parameter is not needed.

b) result. This is a value operation.&Result and consists of the result which is being returned by the performer of this operation, and which is intended to be passed back in the result to the originating DUA. This information is as specified in the appropriate clause of ITU-T Rec. X.511 | ISO/IEC 9594-3.

Should the request fail, one of the errors of the set operation.&Errors will be returned, except that dsaReferral is returned instead of referral.

2.5.5 Chained errors

The dsaReferral error is generated by a DSA when, for whatever reason, it doesn't wish to continue performing an operation by chaining the operation to another DSA. For IN CS2, DSAs do not chain operations incoming from another DSA unless the DSA is in another netwok.

```
IN-dsaReferral  ERROR ::= dsaReferral (
WITH COMPONENTS {
        ...,
        reference           (IN-ContinuationReference),
        contextPrefix       ABSENT})
```

The various parameters have the meanings as described below:

a) The IN-ContinuationReference contains the information needed by the invoker to propagate an appropriate further request, perhaps to another DSA.

b) If the returnCrossRefs component of the ChainingArguments for this operation had the value TRUE, and the referral is being based upon a subordinate or cross-reference, then the contextPrefix parameter may optionally be included. The administrative authority of any DSA will decide which knowledge references, if any, can be returned in this manner (the others, for example, may be confidential to that DSA). Since direct knowledge references are assumed for IN CS2, this parameter is not applicable.

2.6 Protocol overview

2.6.1 ROS-Objects and Contracts

The interactions between DSAs generally required to provide the Directory Abstract Service in the presence of a distributed DIB are defined as an indspContract. A DSA that participates in this contract is defined as a ROS-object of class dsp-dsa. the contract is referred in this specifications as the DSA Abstract Service

```
dsp-dsa  ROS-OBJECT-CLASS ::= {
        BOTH        {indspContract}
        ID          id-rosObject-dspDSA}
```

The Shadow Abstract Service specifies the shadowing of information between a shadow supplier and a shadow consumer DSA. This service is manifested in two forms and therefore is defined as two distinct contracts. They are specified as a ROS-based information objects in subclause 2.6. 2.

The shadowConsumerContract expresses the form of the service in which the shadow consumer, a ROS-object of class initiating-consumer-dsa, initiates the contract. A ROS-object of class responding-supplier-dsa, responds in this contract.

```
initiating-consumer-dsa  ROS-OBJECT-CLASS ::= {
        INITIATES   {shadowConsumerContract}
        ID          id-rosObject-initiatingConsumerDSA}
responding-supplier-dsa  ROS-OBJECT-CLASS ::= {
        RESPONDS    {shadowConsumerContract}
        ID          id-rosObject-respondingSupplierDSA}
```

The shadowSupplierContract expresses the form of the service in which the shadow supplier, a ROS-object of class initiating-supplier-dsa, initiates the contract. A ROS-object of class responding-consumer-dsa, responds in this contract.

```
initiating-supplier-dsa  ROS-OBJECT-CLASS ::= {
        INITIATES   {shadowSupplierContract}
        ID          id-rosObject-initiatingSupplierDSA }
responding-consumer-dsa  ROS-OBJECT-CLASS ::= {
        RESPONDS    {shadowSupplierContract}
        ID          id-rosObject-respondingConsumerDSA }
```

2.6.2 DSP Contract and Packages

The indspContract is defined as an information object of class CONTRACT.

```
indspContract  CONTRACT ::= {
        CONNECTION                  dspConnectionPackage
        INITIATOR CONSUMER OF       {inchainedModifyPackage |
                                    inchainedSerchPackage | chainedExecutePackage}
        ID                          id-contract-indsp}
```

When a pair of DSAs from different open systems interact, this association contract is realized as an SS7 application layer protocol, referred to as the IN Directory System Protocol (DSP). The definition of this protocol in terms of an SS7 application context is provided in subclause 2.7.

The indspContract is composed of a connection package, indspConnectionPackage and two operation packages, inchainedModifyPackage, inchainedSearchPackage and chainedExecutePackage

The connection package, indspConnectionPackage, is defined as an information object of class CONNECTION-PACKAGE. It is identical to the connection package, indapConnectionPackage.

```
dspConnectionPackage  CONNECTION-PACKAGE ::= {
        BIND          dSABind
        UNBIND        in-DSAUnbind
        ID            id-package-dspConnection}
```

The operation packages inchainedModifyPackage and inchainedSearchPackage are defined as information objects of class OPERATION-PACKAGE. The operations of these package are defined in ITU-T Rec. X.518.

```
inchainedModifyPackage  OPERATION-PACKAGE ::= {
        CONSUMER    INVOKES {chainedAddEntry | chainedRemoveEntry |
                             chainedModifyEntry}
        ID                   id-package-inchainedModify}
inchainedSearchPackage OPERATION-PACKAGE::={
        CONSUMER INVOKES {chainedSearch}
        ID               id-package-inchainedSearch}
```

The operation packages chainedExecutePackage is defined an informatiion objects of class OPERATION-PACKAGE

```
chainedExecutePackage OPERATION-PACKAGE::={
        CONSUMER INVOKES {chainedExecute}
        ID               id-package-inchainedExecute}
chainedExecute OPERATION::=chained{execute}
```

Excute operation is defined in Annex A in this specification.

In the indspContract either DSA may assume the role of initiator and invoke the operations of the

contract.

2.6.3 emptyConnectionPackage

```
emptyConnectionPackage   CONNECTION-PACKAGE ::= {
   BIND                  emptyBind
   UNBIND                inEemptyUnbind
   RESPONDER UNBIND      TRUE
   ID                    id-package-emptyConnection
   }
id-package-emptyConnection   OBJECT IDENTIFIER ::= {
id-package 60}
```

2.6.4 DISP Contract and Packages

The shadowConsumerContract and shadowSupplierContract are defined as information objects of class CONTRACT.

```
shadowConsumerContract  CONTRACT ::= {
        CONNECTION            dispConnectionPackage
        INITIATOR CONSUMER OF    {shadowConsumerPackage}
        ID                       id-contract-shadowConsumer}
shadowSupplierContract  CONTRACT ::= {
        CONNECTION            dispConnectionPackage
        RESPONDER CONSUMER OF  {shadowSupplierPackage}
        ID                       id-contract-shadowSupplier}
```

The SS7 realisation of the two forms of Shadow Abstract Service, referred to as the IN Directory Information Shadowing Protocol (DISP) are defined in terms of several SS7 application contexts provided in subclause 2.7.

The shadowConsumerContract and shadowSupplierContract are composed of a common connection package, dispConnectionPackage and one operation package, either shadowConsumerPackage in the first case or shadowSupplierPackage in the second.

The connection package, dispConnectionPackage, is defined as an information object of class CONNECTION-PACKAGE. It is identical to the connection package, dapConnectionPackage.

```
dispConnectionPackage  CONNECTION-PACKAGE ::= {
        BIND          dSAShadowBind
        UNBIND        in-DSAShadowUnbind
        ID            id-package-dispConnection}
```

The operation packages shadowConsumerPackage and shadowSupplierPackage are defined as information objects of class OPERATION-PACKAGE. The operations of these packages are defined in ITU-T Rec. X.525.

```
shadowConsumerPackage  OPERATION-PACKAGE ::= {
```

```
        CONSUMER INVOKES        {requestShadowUpdate}
        SUPPLIER INVOKES        {updateShadow}
        ID                      id-package-shadowConsumer}
shadowSupplierPackage  OPERATION-PACKAGE ::= {
        SUPPLIER INVOKES        {coordinateShadowUpdate | updateShadow}
        ID                      id-package-shadowSupplier}
```

Since the shadow consumer is the initiator of the shadowConsumerContract, it assumes the role of consumer of the shadowConsumerPackage. This means that the shadow consumer invokes the requestShadowUpdate operation and that the shadow supplier invokes the updateShadow operation.

Since the shadow supplier is the initiator of the shadowSupplierContract, it assumes the role of supplier of the shadowSupplierPackage. This means that the shadow supplier invokes the operations of the contract.

2.7 Protocol Abstract Syntax

2.7.1 DSP Abstract Syntax

The Directory ASEs that realize the operation packages specified in subclause 2.6.2 share a single abstract syntax, inDirectorySystemAbstractSyntax. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
inDirectorySystemAbstractSyntax  ABSTRACT-SYNTAX ::= {
        BasicDSP-PDUs
        IDENTIFIED BY       id-as-indirectorySystemAS}
BasicDSP-PDUs ::= TCMessage {{DSP-Invokable},{DSP-Returnable}}

DSP-Invokable OPERATION ::= {chainedAddEntry | chainedRemoveEntry |
                        chainedModifyEntry | chainedSearch | chainedExecute}

DSP-Returnable OPERATION ::= {chainedAddEntry | chainedRemoveEntry |
                                chainedModifyEntry| chainedSearch | chainedExecute}
```

The realisation of the connection package specified in subclause 2.6.2 uses a separate abstract syntax, inDirectoryDSABindingAbstractSyntax. This is specified as an information object of the class ABSTRACT-SYNTAX.

```
inDirectoryDSABindingAbstractSyntax  ABSTRACT-SYNTAX ::= {
        DSABinding-PDUs
        IDENTIFIED BY       id-as-indirectoryDSABindingAS}

DSABinding-PDUs ::= CHOICE {
        bind     Bind {dSABind},
        unbind  Unbind {in-DSAUnbind}}
```

2.7.2 DISP Abstract Syntax

The Directory ASEs that realize the operation packages specified in subclause 2.6.4 share the abstract syntax inDirectoryShadowAbstractSyntax. This abstract syntax is specified as an information object of the class ABSTRACT-SYNTAX.

```
inDirectoryShadowAbstractSyntax ABSTRACT-SYNTAX ::= {
        BasicDISP-PDUs
        IDENTIFIED BY        id-as-indirectoryShadowAS}

BasicDISP-PDUs ::= TCMessage {{DISP-Invokable},{DISP-Returnable}}

DISP-Invokable OPERATION :: ={requestShadowUpdate | updateShadow |
                              coordinateShadowUpdate}
DISP-ReturnableOPERATION ::={requestShadowUpdate | updateShadow |
                              coordinateShadowUpdate}
```

The realisation of the connection package specified above uses a separate abstract syntax, inDirectoryDSAShadowBindingAbstractSyntax. This is specified as an information object of class ABSTRACT-SYNTAX.

```
inDirectoryDSAShadowBindingAbstractSyntax  ABSTRACT-SYNTAX ::= {
        DISPBinding-PDUs
        IDENTIFIED BY        id-as-indsaShadowBindingAS}

DISPBinding-PDUs ::= CHOICE {
        bind     Bind {dSAShadowBind},
        unbind  Unbind {in-DSAShadowUnbind}}
```

2.7.3 Directory System Application Context

The indspContract is realized as the inDirectorySystemAC. This application context is specified as an information object of the class APPLICATION-CONTEXT.

```
inDirectorySystemAC  APPLICATION-CONTEXT ::= {
        CONTRACT                    dspContract
        DIALOGUE MODE               structured
        TERMINATION                 basic
        ABSTRACT SYNTAXES           {dialogue-abstract-syntax |
                                    inDirectorySystemAbstractSyntax |
                                    inDirectoryDSABindingAbstractSyntax}
        APPLICATION CONTEXT NAME    id-ac-indirectorySystemAC}
```

If 3-way authentication is required then the indspContract is realised as the inDirectorySystemWith3seAC. This application context is specified as an information object of the class APPLICATION-CONTEXT.

```
inDirectorySystemWith3seAC APPLICATION-CONTEXT::= {
```

```
        CONTRACT                    dspContract
        DIALOGUE MODE               structured
        TERMINATION                 basic
        ADDITIONAL ASE              {id-se-threewayse}
        ABSTRACT SYNTAXES           {dialogue-abstract-syntax |
                                    inDirectorySystemAbstractSyntax |
                                    inDirectoryDSABindingAbstractSyntax |
                                    inSESEAbstractSyntax}
        APPLICATION CONTEXT NAME    id-ac-indirectorySystemWith3seAC}
```

2.7.4 Directory Shadow Application Context

The inshadowSupplierContract is realized as the inshadowSupplierInitiatedAC. This application context is specified as an information object of the class APPLICATION-CONTEXT.

```
inshadowSupplierInitiatedAC  APPLICATION-CONTEXT ::= {
        CONTRACT                    shadowSupplierContract
        DIALOGUE MODE               structured
        TERMINATION                 basic
        ABSTRACT SYNTAXES           {dialogue-abstract-syntax |
                                    inDirectoryShadowAbstractSyntax |
                                    inDirectoryDSAShadowBindingAbstractSyntax}
        APPLICATION CONTEXT NAME  id-ac-inShadowSupplierInitiatedAC}
```

If 3-way authentication is required then the inshadowSupplierContract is realised as the inshadowSupplierInitiatedWith3seAC. This application context is specified as an information object of the class APPLICATION-CONTEXT.

```
inshadowSupplierInitiatedWith3seAC APPLICATION-CONTEXT::= {
        CONTRACT                    shadowSupplierContract
        DIALOGUE MODE               structured
        TERMINATION                 basic
        ADDITIONAL ASE              {id-se-threewayse}
        ABSTRACT SYNTAXES           {dialogue-abstract-syntax |
                                    inDirectoryShadowAbstractSyntax |
                                    inDirectoryDSAShadowBindingAbstractSyntax |
                                    inSESEAbstractSyntax}
        APPLICATION CONTEXT NAME    id-ac-
                                    inShadowSupplierInitiatedWith3seAC}
```

The inshadowConsumerContract is realized as the inshadowConsumerInitiatedAC. This application context is specified as an information object of the class APPLICATION-CONTEXT.

```
inshadowConsumerInitiatedAC  APPLICATION-CONTEXT ::= {
        CONTRACT                    shadowConsumerContract
        DIALOGUE MODE               structured
        TERMINATION                 basic
        ABSTRACT SYNTAXES           {dialogue-abstract-syntax |
```

inDirectoryShadowAbstractSyntax |
inDirectoryDSAShadowBindingAbstractSyntax}
APPLICATION CONTEXT NAME id-ac-inShadowConsumerInitiatedAC}

If 3-way authentication is required then the inshadowConsumerContract is realised as the inshadowConsumerInitiatedWithAC. This application context is specified as an information object of the class APPLICATION-CONTEXT.

inshadowConsumerInitiatedWith3seAC APPLICATION-CONTEXT::= {
CONTRACT shadowConsumerContract
DIALOGUE MODE structured
TERMINATION basic
ADDITIONAL ASE {id-se-threewayse}
ABSTRACT SYNTAXES {dialogue-abstract-syntax |
inDirectoryShadowAbstractSyntax |
inDirectoryDSAShadowBindingAbstractSyntax |
inSESEAbstractSyntax}
APPLICATION CONTEXT NAME id-ac-
inShadowConsumerInitiatedWith3seAC}

2.8 Mapping onto Services

The DSP and DISP can be mapped onto TC services. This subclause defines the mapping of the DSABind, DSAUnbind, DSAShadowBind and DSAShadowUnbind services onto the services of the TC dialogue handling services defined in B-IF4.21 document.

The DirectoryBind service in mapped onto TC-services as follows:

a) The TC-BEGIN service is used to invoke the DSAShadowBind and DSABind operations.

b) The TC-CONTINUE service is used to report the success of the DSAShadowBind and DSABind operations.

c) The TC-U-ABORT service is used to report the failure of the DSAShadowBind and DSABind operations.

The mapping of the parameters onto the TCDialogue services is as follows:

The use of parameters of the TC-BEGIN service is defined with the following qualifications.

- The Application Context Name parameter of the TC-BEGIN service shall take the value of the application-context-name field of the inDirectorySystemAC or shadowSupplierInitiatedAC or shadowConsumerInitiatedAC object.

- The User Information parameter of the TC-BEGIN service shall contain a value of type EXTERNAL which depends on the value of the Application Context being used .

If the Application Context being used is inDirectorySystemAC then the User Information parameter

shall contain an EXTERNAL Type with the direct-reference field, if present, set to id-as-indirectoryDSABindingAS and the value to be encoded shall be of type DSABinding-PDUs.bind.bind-invoke.

If the Application Context being used is shadowSupplierInitiatedAC or shadowConsumerInitiatedAC then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to id-as-indsaShadowBindingAS and the value to be encoded shall be of type DISPBinding-PDUs.bind.bind-invoke.

The use of parameters of the TC-CONTINUE service is defined with the following qualifications.

- The Dialogue Id parameter of the TC-CONTINUE service shall be used as specified in B-IF4.21 document. The Authorized Relationship Id can be mapped onto the TCAP Dialogue Id.

- The User Information parameter of the first TC-CONTINUE service shall contain a value of type EXTERNAL which depends on the value of the Application Context being used .

If the Application Context being used is inDirectorySystemAC then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to id-as-indirectoryDSABindingAS and the value to be encoded shall be of type DSABinding-PDUs.bind.bind-result.

If the Application Context being used is shadowSupplierInitiatedAC or shadowConsumerInitiatedAC then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to id-as-indsaShadowBindingAS and the value to be encoded shall be of type DISPBinding-PDUs.bind.bind-result.

The use of parameters of the TC-U-ABORT service is defined with the following qualifications.

- The Dialogue Id parameter of the TC-U-ABORT service shall be used as specified in B-IF4.21 document. The Authorized Relationship Id can be mapped onto the TCAP Dialogue Id.

- The Application-Context-Name parameter of the TC-U-ABORT service shall be used as specified in B-IF4.21 document. When the SDF refuses a dialogue because the application-context-name it receives is not supported, this parameter shall have the value of the application-context-name field of the inDirectorySystemAC or shadowSupplierInitiatedAC or shadowConsumerInitiatedAC object.

- When the TC-U-ABORT is reporting the failure of a bind operation, i.e. the abort reason parameter has the value "dialogue-refused", the User Information parameter of the TC-U-ABORT service shall contain a value of type EXTERNAL which depends on the value of the Application Context being used .

If the Application Context being used is inDirectorySystemAC then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to id-as-indirectoryDSABindingAS and the value to be encoded shall be of type DSABinding-PDUs.bind.bind-error.

If the Application Context being used is shadowSupplierInitiatedAC or shadowConsumerInitiatedAC

then the User Information parameter shall contain an EXTERNAL Type with the direct-reference field, if present, set to id-as-indsaShadowBindingAS and the value to be encoded shall be of type DISPBinding-PDUs.bind.bind-error.

Otherwise it shall be absent.

The inDSAUnbind and inDSAShadowUnbind services are mapped onto the TC-END service. The use of the parameters of the TC-END service is defined in the following qualifications .

- The Quality of Service parameter shall be used as specified in B-IF4.21 document. No specific constraints are imposed by this document.

- The Application-Context-Name parameter is not used in this phase of a dialogue.

- The Dialogue Id parameter shall be used as specified in B-IF4.21 document.

- The User Information parameter shall be empty.

- The Component Present parameter shall be used as specified in B-IF4.21 document.

The SETransfer service is mapped onto TC-services is the same as for the other services except:

- If the SESE is included in the application context, the User Information parameter of the TC-CONTINUE service shall contain a value of type seItem.

The Directory ASE services are mapped onto the TC component handling services. The mapping of operations and errors onto TC services is defined in B-IF4.24 document.

The timeout parameter of the TC-INVOKE-Req. primitives is set according to the following table:

Table 2-1/B-IF4.28
TC timer values of DSP/DISP operations

Operation	Timeout
ChainedAddEntry	medium
ChainedRemoveEntry	medium
ChainedModifyEntry	medium
ChainedExecute	medium
ChainedSearch	medium
UpdateShadow	medium
CoordinateShadowUpdate	medium
RequestShadowUpdate	medium

2.9 Conformance

For the conformance of SDFs, the following statements should be added to the list of already existing statements.

2.9.1 Conformance by SDFs

2.9.1.1 Statement requirements

The following shall be stated:

a) the application-context for which conformance is claimed. The present version of this document requires conformance to the inDirectorySytemAC application-context;

   Note: An application context shall not be divided except as stated herein; in particular, conformance shall not be claimed to particular operations.

b) if conformance is claimed to the inDirectorySystemAC application-context, whether or not the chained mode of operation is supported, as defined in ITU-T Rec. X.518;

c) the security-level(s) for which conformance is claimed (none, simple, strong);

d) the attribute types for which conformance is claimed and whether for attributes based on the syntax DirectoryString, conformance is claimed for the UNIVERSAL STRING choice;

e) the object classes, for which conformance is claimed;

f) whether conformance is claimed for collective attributes as defined in 8.8/ITU-T Rec. X.501 and 7.6, 7.8.2 and 9.2.2/ITU-T Rec. X.511;

g) whether conformance is claimed for hierarchical attributes as defined in 7.6, 7.8.2 and 9.2.2/ITU-T Rec. X.511;

h) the operational attribute types defined in ITU-T Rec. X.501 and any other operational attribute types for which conformance is claimed;

i) whether conformance is claimed for return of alias names as described in 7.7.1/ITU-T Rec. X.511;

j) whether conformance is claimed for indicating that returned entry information is complete, as described in 7.7.6/ITU-T Rec. X.511;

k) whether conformance is claimed for modifying the object class attribute to add and/or remove values identifying auxiliary object classes, as described in 11.3.2/ITU-T Rec. X.511;

l) whether conformance is claimed to Basic Access Control;

m) whether conformance is claimed to Simplified Access Control;

n) the name bindings for which conformance is claimed;

o) whether the SDF is capable of administering collective attributes, as defined in ITU-T Rec. X.501;

p) whether conformance is claimed for attribute contexts.

2.9.1.2 Static requirements

An SDF shall:

a) have the capability of supporting the application-contexts for which conformance is claimed as defined by their abstract syntax in 2.7;

b) have the capability of supporting the information framework defined by its abstract syntax in ITU-T Rec. X.501;

c) conform to the minimal knowledge requirements defined in ITU-T Rec. X.518;

d) have the capability of supporting the attribute types for which conformance is claimed; as defined by their abstract syntaxes;

e) have the capability of supporting the object classes for which conformance is claimed, as defined by their abstract syntaxes;

f) conform to the extensions for which conformance was claimed in the previous clause;

g) if conformance is claimed for collective attributes, have the capability of performing the related procedures defined in 7.6, 7.8.2 and 9.2.2/ITU-T Rec. X.511;

h) if conformance is claimed for hierarchical attributes, have the capability of performing the related procedures defined in 7.6, 7.8.2 and 9.2.2/ITU-T Rec. X.511;

i) have the capability of supporting the operational attribute types for which conformance is claimed;

j) if conformance is claimed to Basic Access Control, have the capability of holding ACI items that conform to the definitions of Basic Access Control;

k) if conformance is claimed to Simplified Access Control, have the capability of holding ACI items that conform to the definitions of Simplified Access Control.

2.9.1.3 Dynamic requirements

An SDF shall:

a) conform to the mapping onto used services defined in subclause 2.8;

b) conform to the procedures for distributed operations of the Directory related to referrals, as defined in ITU-T Rec. X.518;

c) if conformance to the directorySystemAC application-context, conform to the referral mode of interaction as defined in ITU-T Rec. X.518;

d) if conformance is claimed for the chained mode of interaction, conformance to the chained mode of interaction as defined in ITU-T Rec. X.518;

Note: Only in this case it is necessary for a DSA to be capable of invoking operations of the directorySystemAC.

e) conform to the rules of extensibility procedures will be provided in ITU-T;

f) if conformance is claimed to Basic Access Control, have the capability of protecting information within the SDF in accordance with the procedures of Basic Access Control;

g) if conformance is claimed to Simplified Access Control, have the capability of protecting information within the SDF in accordance with the procedures of Simplified Access Control;

2.9.2 Conformance by a shadow supplier

A SDF implementation claiming conformance to this Directory Specification in the role of shadow supplier shall satisfy the requirements specified below.

2.9.2.1 Statement requirements

The following shall be stated:

a) the application context(s) for which conformance is claimed as a shadow supplier: inShadowSupplierInitiatedAC and inShadowConsumerInitiatedAC;

b) the security-level(s) for which conformance is claimed (none, simple, strong);

c) to which degree the UnitOfReplication is supported. Specifically, which (if any) of the following optional features are supported:

- Entry filtering on ObjectClass;

- Selection/Exclusion of attributes via AttributeSelection;

- The inclusion of subordinate knowledge in the replicated area;

- The inclusion of extended knowledge in addition to subordinate knowledge.

2.9.2.2 Static requirements

A SDF shall:

a) have the capability of supporting the application-context(s) for which conformance is claimed as defined in their abstract syntax above;

b) provide support for modifyTimestamp and createTimestamp operational attributes.

2.9.2.3 Dynamic requirements

A SDF shall:

a) conform to the mapping onto used services defined in 2.8;

b) conform to the procedures of ITU-T Rec. X.525 | ISO/IEC 9594-9 as they relate to the DISP.

2.9.3 Conformance by a shadow consumer

A SDF implementation claiming conformance to this Directory Specification as a shadow consumer shall satisfy the requirements specified below:

2.9.3.1 Statement requirements

The following shall be stated:

a) the application context(s) for which conformance is claimed as a shadow supplier: inShadowSupplierInitiatedAC and inShadowConsumerInitiatedAC;

b) the security-level(s) for which conformance is claimed (none, simple, strong);

c) whether the SDF supports shadowing of overlapping units of replication.

2.9.3.2 Static requirements

A SDF shall:

a) have the capability of supporting the application-context(s) for which conformance is claimed as defined in their abstract syntax in 2.7;

b) provide support for modifyTimestamp and createTimestamp operational attributes if overlapping units of replication is supported;

c) provide support for the copyShallDo service control.

2.9.3.3 Dynamic requirements

A SDF shall:

a) conform to the mapping onto used services defined in 2.8;

b) conform to the procedures of ITU-T Rec. X.525 | ISO/IEC 9594-9 as they relate to the DISP.

2.10 X.500 Recommendations Profiles

2.10.1 X.501 Profile

To support shadowing 8, 9/ITU-T Recommendation X.501 need to be considered. They describe how

the DIT can be distributed over several SDFs and this includes the description of the knowledge references and all the information needed to do the splitting of a data base into several data bases.

2.10.2 X.518 Profile

Sections 1 to 4 should be imported as well as section 5 up to part 19.1.

2.10.3 X.525 Profile

From the X.525 Recommendation every part should be imported except parts 6.3.2, 8, 10.2 and 11.2.

3. Procedures

3.1 Definition of procedures and entities

3.1.1 SDF application entity procedures

3.1.1.1 General

This subclause provides the definition of the SDF application entity (AE) procedures related to the SDF-SDF interface. The procedures are based on the use of Signalling System No. 7 (SS No.7).

Other capabilities may be supported in an implementation-dependent manner in the SCP, SDP, SSP, AD or SN.

The AE, following the architecture defined in ITU-T Recommendation Q.700, B-IF4.21 and ITU-T Recommendation Q.1400, includes TCAP (transaction capabilities application part) and one or more ASEs called TC-users, which are based on the Directory (X.500 series of Recommendations). The following subclauses define the TC-user ASE and SACF and MACF rules, which interface with TCAP using the primitives specified in B-IF4.21.

The procedure may equally be used with other signalling message transport systems supporting the application layer structures defined.

In case interpretations for the application entity procedures defined in the following differ from detailed procedures and the rules for using of TCAP service, the statements and rules contained in the detailed subclause 3.3 shall be followed.

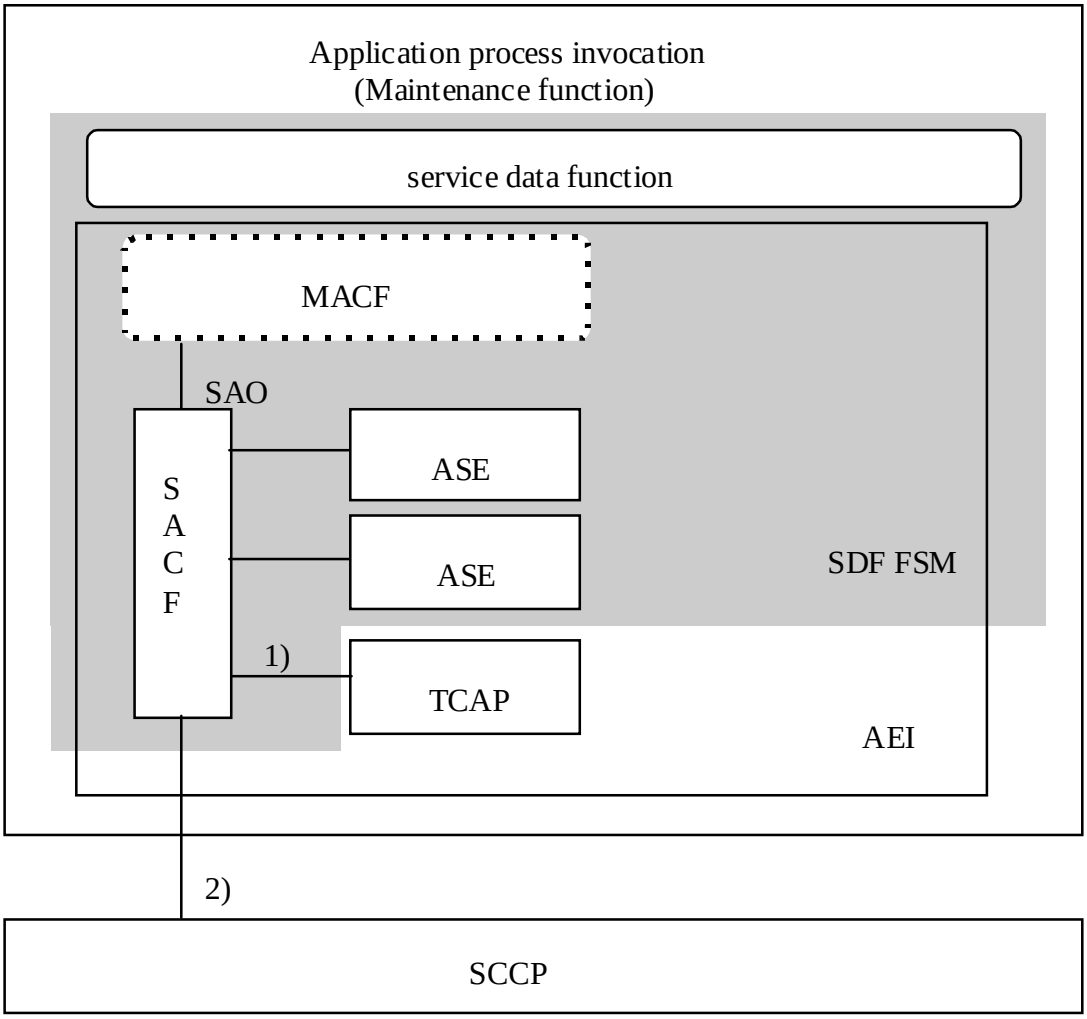
3.1.1.2 Model and interfaces

The functional model of the AE-SDF is shown in Figure 3-1/B-IF4.28; the ASEs interface to TCAP to communicate with other SDFs, and interface to the maintenance functions. The scope of this Recommendation is limited to the shaded area in Figure 3-1/B-IF4.28.

The interfaces shown in Figure 3-1/B-IF4.28 use the TC-user ASE primitives specified in B-IF4.21 [interface 1)] and N-Primitives specified in B-IF4.11 [interface 2)]. The operations and parameters of Intelligent Network Application Protocol (INAP) are defined in clause 2 of this specification.

An instance of a specific SDF FSM may be created if an SDF invokes or responds to a chaining operation to/from another SDF, or an SDF invokes or responds to a shadowing operation to/from another SDF.

The SDF FSM handles the interaction with another SDF FSM.



AEI Application entity invocation
SDF Service data function
FSM Finite state machine
SACF Single association control function
SAO Single association object
MACF Multiple association control function

1) TC-primitives.
2) N-primitives.

NOTE-The SDF FSM includes several finite state machines.

Figure 3-1/B-IF4.28 Functional model of SDF

3.1.1.3 SDF FSM structure

The structure of the SDF FSMs is illustrated in Figure 3-2/B-IF4.28

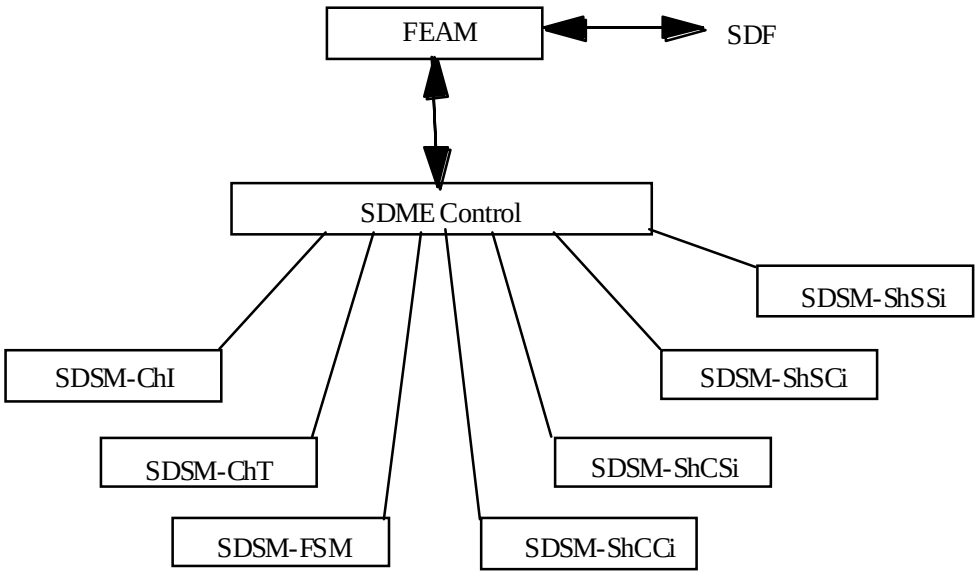


Figure 3-2/B-IF4.28 SDF Interface

The SDSM-ChI and SDSM-ChT handle the interactions between SDFs for chaining initiation and termination. The SDSM-ShSSi,•SDSM-ShSCi,•SDSM-ShCSi and SDSM-ShCCi handle the interactions between SDFs for shadowing master control and copy control. The SDME-FSM handles the interaction between the SDF and the SDF management functions.

The management functions related to the execution of operations received from the cooperating SDF are executed by the SDF Management Entity (SDME). The SDME is comprised of an SDME control and several instances of SDME FSMs. The SDME control interfaces the different SDF FSMs (e.g., SDSM-ChI) and SDME-FSMs respectively as well as the Functional Entity Access Manager (FEAM).

The FEAM provides the low level interface maintenance functions including the following:

1) Establishing and maintaining interfaces to the cooperating SDFs;
2) Passing and queuing (when necessary) the messages received from the cooperating SDF to the SDME Control;
3) Formatting, queuing (when necessary), and sending the messages received from the SDME Control to the cooperating SDF.

The SDME Control maintains the associations with the cooperating SDFs on behalf of all instances of the SDF FSMs (e.g., SDSM-ChI). These instances of the SDF FSMs occur concurrently and asynchronously as SDF related events occur, which explains the need for a single entity that performs the task of creation, invocation and maintenance of the SDF FSMs. In particular, the SDME Control performs the following tasks:

1) Interprets the input messages from other FEs and translates them into corresponding SDF FSM events;
2) Translates the SDF FSM outputs into corresponding messages to other FEs;
3) Captures asynchronous activities related to management or supervisory functions in the SDF and

creates an instance of an SDME-FSM. For example, management invocation of a shadowing procedure between network operators. In this case, the SDME Control will create an instance of the SDME-FSM to handle this management related operation.

3.1.1.4 SDF state transition models

3.1.1.4.1 SDF state transition model for SDF related states

The SDF's job relating to interactions with other SDFs is to act upon shadowing and chaining operations. States related to SDF/SDF interactions for shadowing are given in subclause 3.1.1.4.1.1. States related to SDF/SDF interactions for chaining are given in subclause 3.1.1.4.1.2.

3.1.1.4.1.1 SDF state transition models for shadowing

As for the shadowing procedure, an SDF can play the role of a copy supplier and a copy consumer. Moreover, the shadowing procedure can be initiated by a copy consumer as well as a copy supplier. Therefore, there can be in total four FSMs as described below.
The four different SDF FSMs could be gathered into one more complex FSM, but to clearly show the different roles played by an SDF, it was thought better to have four separate FSMs.

In the following FSMs, the possibility of sending the DSAShadowBind operation together with other DISP operations in one TC message is taken into considerations.

Shadow supplier-initiated supplier state machine (SDSM-ShSSi)

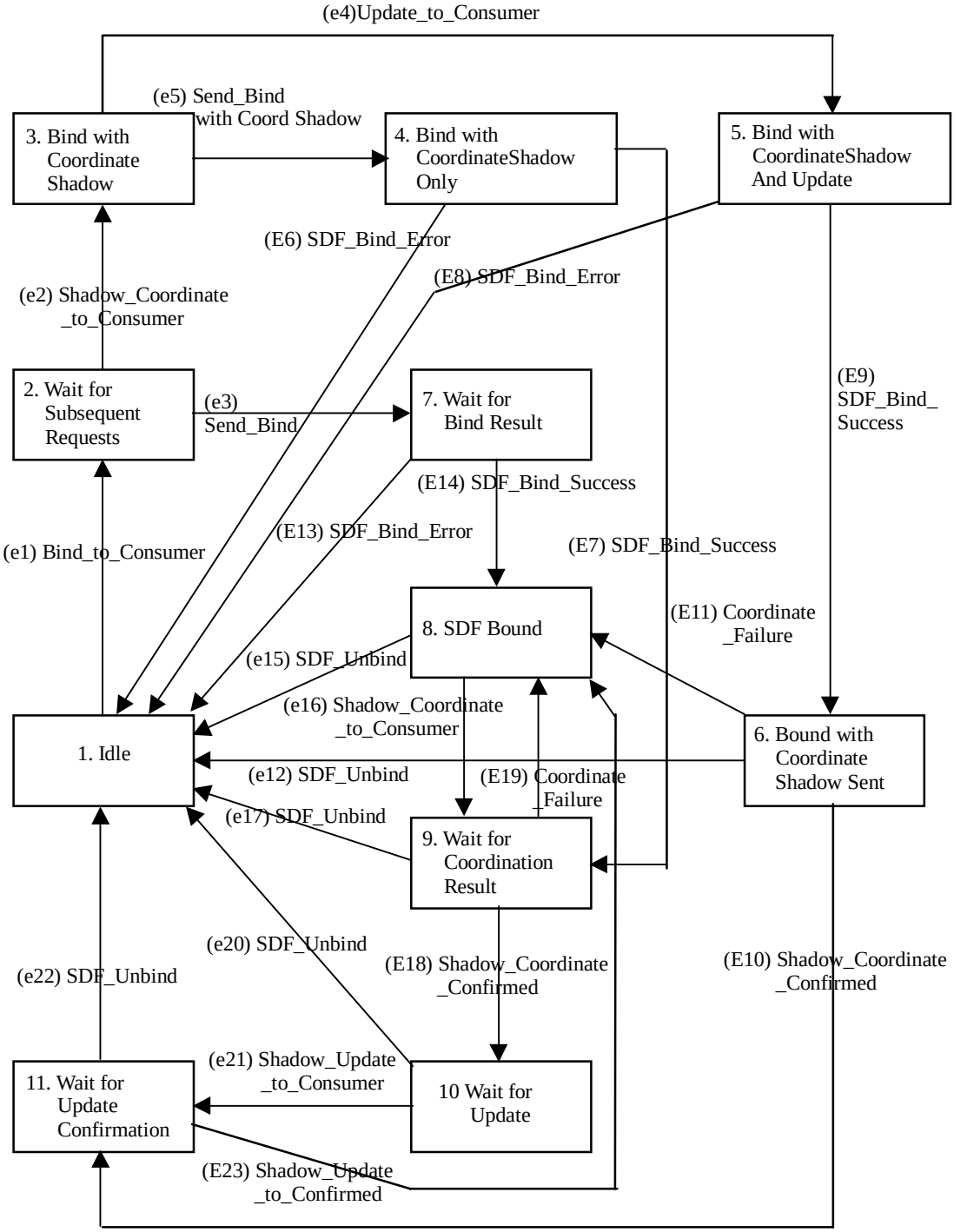


Figure 3-3/B-IF4.28 SDF FSM for a copy supplier in case of supplier-initiated (SDSM-ShSSi)

State 1 : "Idle"
There is only one event accepted in this state:

- (e1) Bind_to_Consumer: This is an internal event caused by the sending of a DSAShadowBind operation. This causes a transition out of this state to State 2 Wait for Subsequent Requests;

State 2 : “Wait for Subsequent Requests”

In this state, a CoordinateShadowUpdate operation to be sent with the DSAShadowBind operation (in the same message) to the consumer is expected. The following two events are considered in this state:

- (e2) Shadow_Coordinate _to_Consumer: This is an internal event caused by the reception of a CoordinateShadowUpdate operation. The operation is buffered until the reception of a delimiter (or a timer expiration). This event causes a transition out of this state to State 3 Bind with Coordinate Shadow;
- (e3) Send_Bind: This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent sent or the expiration of a timer. Once the internal event is received, a TCAP message containing DSAShadowBind operation is sent to the consumer SDF. This event causes a transition out of this state to State 7 Wait for Bind Results.

State 3 : " Bind with Coordinate Shadow "

In this state, an UpdateShadow operation to be sent with the DSAShadowBind and CoordinateShadowUpdate operations, or a delimiter is expected. Two events are considered in this state:

- (e4) Update_to_Consumer: This is an internal event, caused by the reception of an UpdateShadow. This event causes a TCAP message containing the DSAShadowBind, CoordinateShadowUpdate and UpdateShadow operations to be sent to the consumer SDF. This event causes a transition out of this state to State 5 Bind with Coordinate Shadow and Update;
- (e5) Send_Bind_with_CoordShadow: This is an internal event, caused by the reception of a delimiter that indicates the reception of the last operation to be sent or the expiration of a timer. Once the internal event is received, a TCAP message containing the DSAShadowBind and CoordinateShadowUpdate operations is sent to the consumer SDF. This event causes a transition out of this state to State 4 Bind with Coordinate Shadow Only;

State 4 : " Bind with Coordinate Shadow Only "

In this state, a DSAShadowBind result is expected from the consumer SDF. Two events are considered in this state:

- (E6) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 Idle.
- (E7) SDF_Bind_Success: This is an external event, caused by the reception of a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to State 9 Wait for Coordination Result;

State 5 : " Bind with Coordinate Shadow and Update "

In this state, a DSAShadowBind result is•expected from the consumer SDF. Two events are considered in this state:

- (E8) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 Idle;
- (E9) SDF_Bind_Success: This is an external event, caused by the reception of the a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to State 6 Bound with Coordinate Shadow Sent;

State 6 : “Bound with Coordinate Shadow Sent”

In this state, a Coordinate ShadowUpdate result is expected from the consumer. Three events are considered in this state:

- (E10) Shadow_Coordinate_Confirmed: This is an external event, caused by the reception of a CoordinateShadowUpdate result. This indicates the successful completion of the CoordinateShadowUpdate operation previously issued to the consumer SDF. This event causes a transition out of this state to State 11 Wait for Update Confirmation;
- (E11) Coordinate_Failure: This is an external event, caused by the reception of an error to the previously issued CoordinateShadowUpdate operation. This event causes a transition out of this state to State 8 SDF Bound.
- (e12) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;

State 7 : "Wait for Bind Result "

In this state, a DSAShadowBind result is expected from the consumer. Two events are considered in this state:

- (E13) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the consumer SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 Idle;
- (E14) SDF_Bind_Success: This is an external event caused by the successful completion of the DSAShadowBind operation previously issued to the consumer SDF. This event causes a transition out of this state to State 8 SDF Bound.

State 8 : " SDF Bound "

In this state, the SDF has established an "authenticated association" to the consumer and is ready to send a CoordinateShadowUpdate operation to it. Two events are considered in this state:

- (e15) SDF_Bind_Error: SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure) or causing the issuing of the in-DSAShadowUnbind operation. This event causes a transition out of this state to State 1 Idle;
- (e16) Shadow_Coordinate_to_Consumer: This is an internal event, that causes the sending of a request to a consumer SDF to coordinate the shadow to have it later updated. This event causes a transition out of this state to State 5 Wait for Coordination Result.

State 9 : "Wait for Bind Result "

In this state, the supplier SDF has sent a CoordinateShadowUpdate request and waits for the answer from the consumer SDF. Three events are considered in this state:

- (e17) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (E18) Shadow_Coordinate_Confirmed: This is an external event, caused by the reception of the response to the previously issued CoordinateShadowUpdate operation. This event causes a transition out of this state to State 6 Wait for Update;
- (E19) Coordinate_Failure: This is an external event caused by the reception of an error to the previously issued CoordinateShadowUpdate request operation. This event causes a transition back to State 8 SDF Bound.

State 10 : "Wait for Update "

In this state, the supplier SDF has received a confirmation to the previously issued Coordinate ShadowUpdate request and is ready to send an UpdateShadow request to the consumer SDF. Two events are considered in this state:

- (e20) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated

association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 Idle;

- (e21) Shadow_Update_to_Consumer: This is an internal event, that causes the sending of a request to the consumer SDF to update the shadow. This event causes a transition out of this state to State 11 Wait for Update Confirmation.

State 11 : "Wait for Update Confirmation "

In this state, the supplier SDF has sent a UpdateShadow request and waits for the answer from the consumer SDF. Two events are considered in this state:

- (e22) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g. during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (E23) Shadow_Update_Confirmed: This is an external event caused by the reception of the response to the previously issued UpdateShadow operation. This event causes a transition out of this state to State 8 SDF Bound.

Shadow consumer-initiated supplier state machine (SDSM-ShSCi)

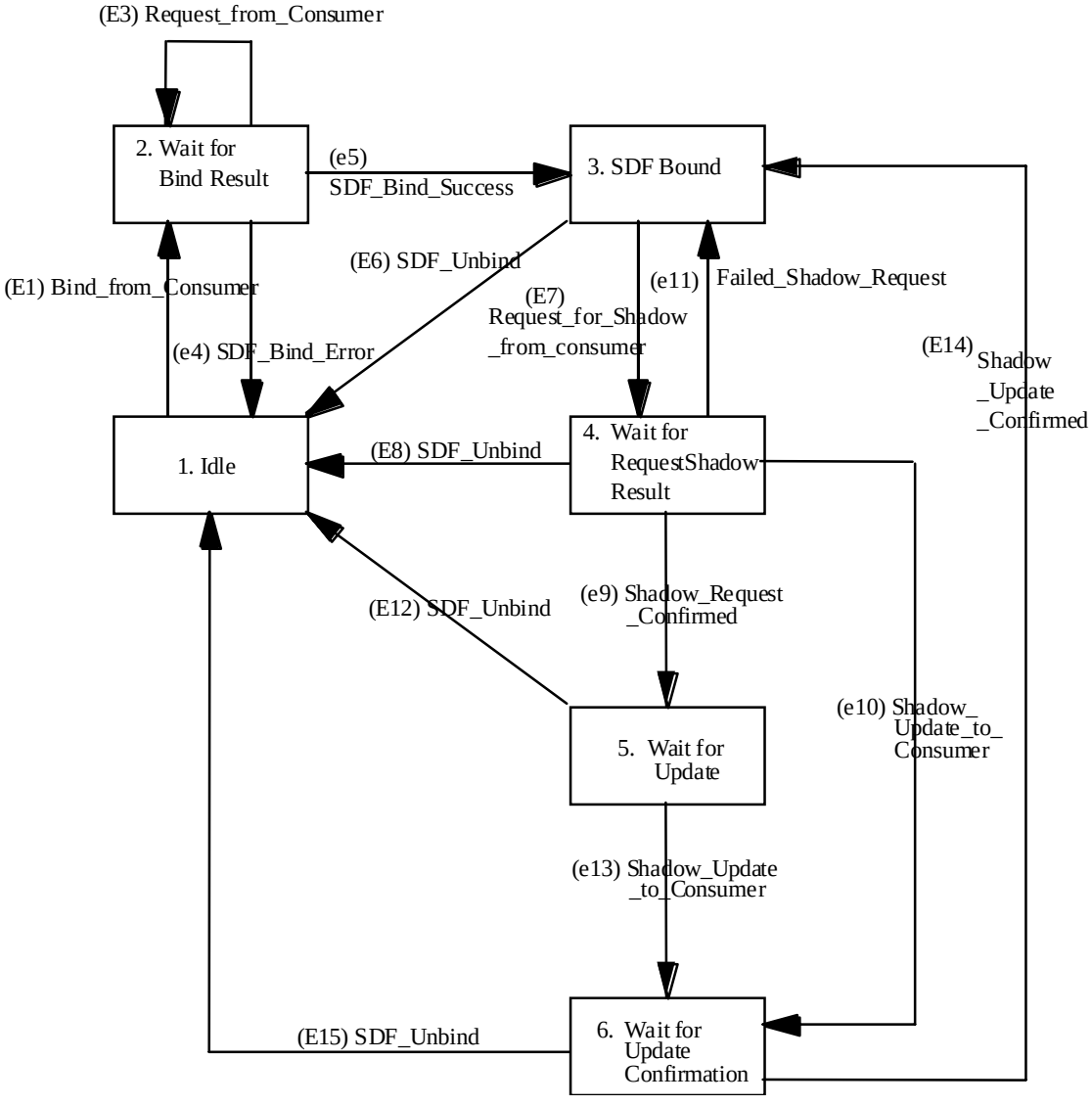


Figure 3-4/B-IF4.28 SDF FSM for a copy supplier in case of consumer-initiated (SDSM-ShSCi)

State 1 : "Idle"
There is only one event accepted in this state:

- (E1) Bind_from_Consumer: This is an external event caused by the reception of a DSAShadowBind operation. This causes a transition out of this state to State 2 Wait for Bind Result.

State 2 : "Wait for Bind Result"
In this state, a DSAShadowBind operation is being performed. Three events are considered in this state:

- (E3) Request_from_Consumer: This is an external event, caused by the reception of operations before the result from the DSAShadowBind operation is determined. This occurs when the RequestShadowUpdate is sent in the same TCAP message as the bind request. The RequestShadowUpdate message is stored and a transition occurs to the same state. When a transition occurs to the next state, the RequestShadowUpdate is re-examined as if it occurred in that state;;
- (e4) SDF_Bind_Error: This is an internal event, caused by the failure of the DSAShadowBind operation previously issued from the consumer. A DSAShadowBind error is returned. This event causes a transition out of this state to State 1 Idle;
- (e5) SDF_Bind_Success: This is an internal event, caused by the successful completion of the DSAShadowBind operation previously issued from the consumer. This event causes a transition out of this state to State 3 SDF Bound.

State 3 : "SDF Bound"
In this state, the supplier SDF is expecting a RequestShadowUpdate operation from the consumer SDF. Two events are considered in this state:

- (E6) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g., during a user's release procedure) or by the reception of the in-DSAShadowUnbind operation. This event causes a transition out of this state to State 1 Idle;
- (E7) Request_for_Shadow_from_Consumer: This is an external event, caused by the reception of a RequestShadowUpdate operation from the consumer SDF. This event causes a transition out of this state to State 4 Wait for RequestShadow Result.

State 4 : "Wait for RequestShadow Result"
In this state, the supplier SDF has received a RequestShadowUpdate operation from the consumer SDF. Four events are considered in this state:

- (E8) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (e9) Shadow_Request_Confirmed: This is an internal event, that signals that the update agreement is acceptable. This causes the sending of the response to a previously received RequestShadowUpdate operation by the supplier SDF. This event causes a transition out of this state to State 5 Wait for Update;
- (e10) Shadow_Update_to_Consumer: This is an internal event, that signals that the update agreement is acceptable, and an UpdateShadow message is ready to be sent. This causes the sending of both a RequestShadowUpdate result and an UpdateShadow operation in the same TCAP message. This event causes a transition out of this state to State 6 Wait for Update Confirmation;
- (e11) Failed_Shadow_Request: This is an internal event, caused by the sending of an error to a previously received RequestShadowUpdate operation. This event causes a transition out

of this state to State 3 SDF Bound.

State 5 : “Wait for Update”

In this state, the supplier SDF has sent a response to the previously received RequestShadowUpdate operation and is ready to send an UpdateShadow operation to the consumer. Two events are considered in this state:

- (E12) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (e11) Shadow_Update_to_Consumer: This is an internal event that causes the sending of a request to the consumer SDF to update the shadow. This event causes a transition out of this state to State 6 Wait for Update Confirmation.

State 6 : "Wait for Update Confirmation"

In this state, the supplier SDF has sent an UpdateShadow request and waits for the answer from the consumer SDF. Two events are considered in this state:

- (E14) Shadow_Update_Confirmed: This is an external event caused by the reception of the response to the previously issued UpdateShadow operation. This event causes a transition out of this state to State 5 Wait for Update;
- (E15) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle.

Shadow supplier-initiated consumer state machine (SDSM-ShCSi)

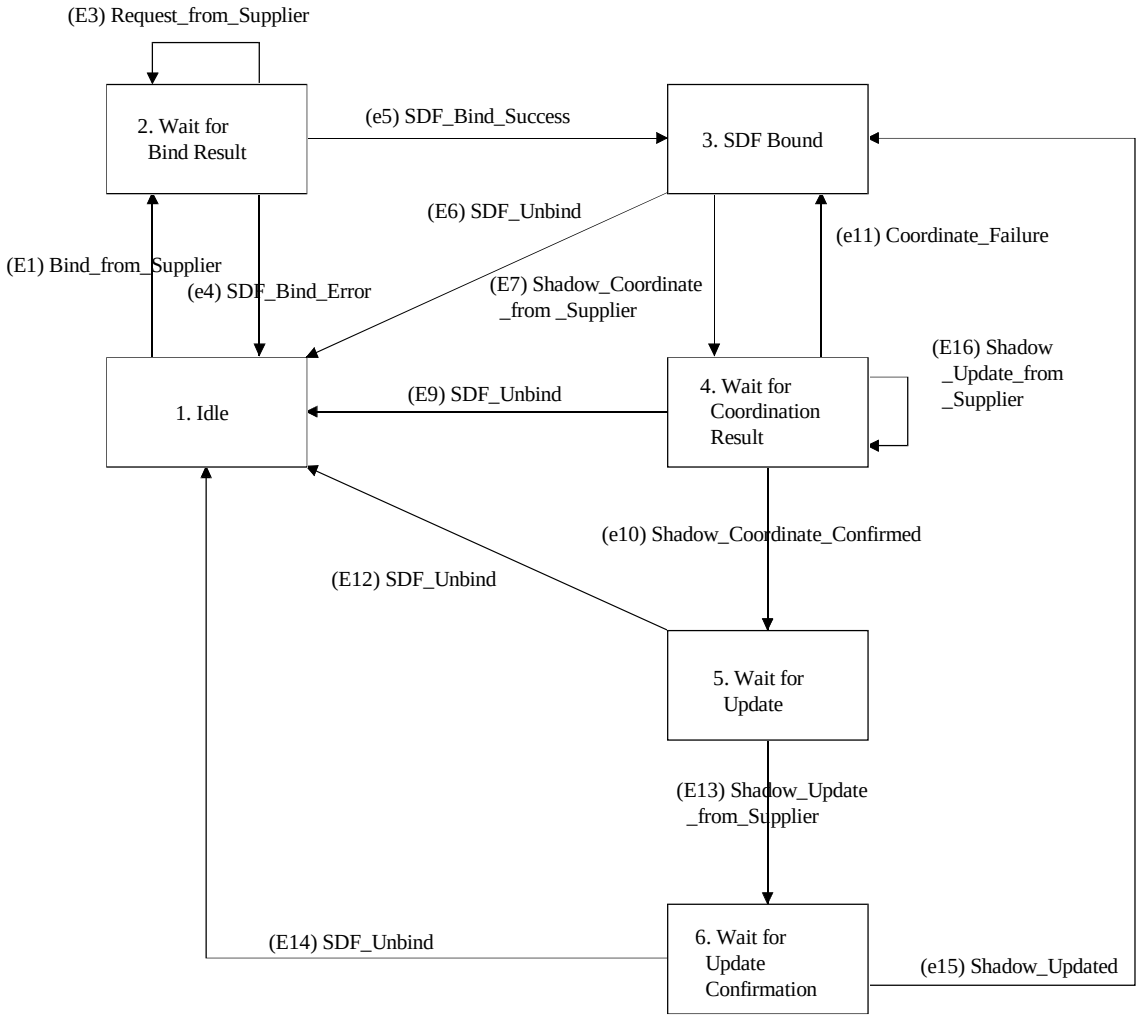


Figure 3-5/B-IF4.28 SDF FSM for a copy consumer in case of supplier-initiated (SDSM-ShCSi)

State 1 : "Idle"
There is only one event accepted in this state:
- (E1) Bind_from_Supplier: This is an external event caused by the reception of a DSAShadowBind operation. This causes a transition out of this state to State 2 Wait for Bind Result.State 2 : "Wait for Bind Result"

In this state, the consumer has received a DSAShadowBind operation and is answering that operation
Three events are considered in this state:

- (E3) Request_from_Supplier: This is an external event, caused by the reception of operations before the result from the DSAShadowBind operation is determined. This occurs when the CoordinateShadowUpdate or UpdateShadow are sent in the same TCAP messages as the bind request. These operations are stored and a transition occurs to the same state. When a transition occurs to the following states, these operations are re-examined as if they occurred in that state;
- (e4) SDF_Bind_Error: This is an internal event, caused by the failure of the DSAShadowBind operation previously issued from the supplier SDF. A DSAShadowBind error is returned.

This event causes a transition out of this state to State 1 Idle;

- (e5) SDF_Bind_Success: This is an internal event, caused by the successful completion of the DSAShadowBind operation previously issued from the supplier SDF. This event causes a transition out of this state to State 3 SDF Bound.

State 3 : "SDF Bound"

In this state, the consumer SDF is expecting a CoordinateShadowUpdate operation from the supplier SDF. Two events are considered in this state:

- (E6) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g., during a user's release procedure) or by the reception of the in-DSAShadowUnbind operation. This event causes a transition out of this state to State 1 Idle;
- (E7) Shadow_Coordinate_from_Supplier: This is an external event, caused by the reception of a CoordinateShadowUpdate operation from the supplier SDF. This event causes a transition out of this state to State 4 Wait for Coordination Result.

State 4 : "Wait for Coordination Result"

In this state, the consumer SDF has received a CoordinateShadowUpdate operation from the supplier SDF and is processing it. Four events are considered in this state:

- (E9) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (e10) Shadow_Coordinate_Confirmed: This is an internal event, caused by the successful completion of a CoordinateShadowUpdate operation from the supplier SDF. This event causes a transition out of this state to State 5 Wait for Update;
- (e11) Coordinate_Failure: This is an internal event, caused by the sending of an error to a previously received CoordinateShadowUpdate operation. This event causes a transition out of this state to State 3 SDF Bound.
- (E16) Shadow_Update_from_Supplier: This is an external event, caused by the reception of a UpdateShadow operation in the same TCAP message as a CoordinateShadowBind and CoordinateShadowUpdate. The UpdateShadow message is stored and a transition occurs to the same state. When a transition occurs to the next state, the UpdateShadow is re-examined as if it had occurred in that state.

State 5 : "Wait for Update"

In this state, the SDF is expecting an UpdateShadow operation. Two events are considered in this state:

- (E12) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (E13) Shadow_Update_from_Supplier: This is an external event, caused by the reception of the UpdateShadow operation issued from the supplier SDF. This event causes a transition out of this state to State 6 Wait for Update Confirmation.

State 6 : “Wait for Update Confirmation”

In this state, the consumer SDF has received an UpdateShadow operation from the supplier SDF and processes to update the copy. Two events are considered in this state:

- (E14) SDF_Unbind: This is an external event, caused by the cancellation of the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (e15) Shadow_Updated: This is an internal event, caused by the completion of an UpdateShadow operation and the sending of the response to it. This event causes a

transition out of this state to State 3 SDF Bound.

Shadow consumer-initiated consumer state machine (SDSM-ShCCi)

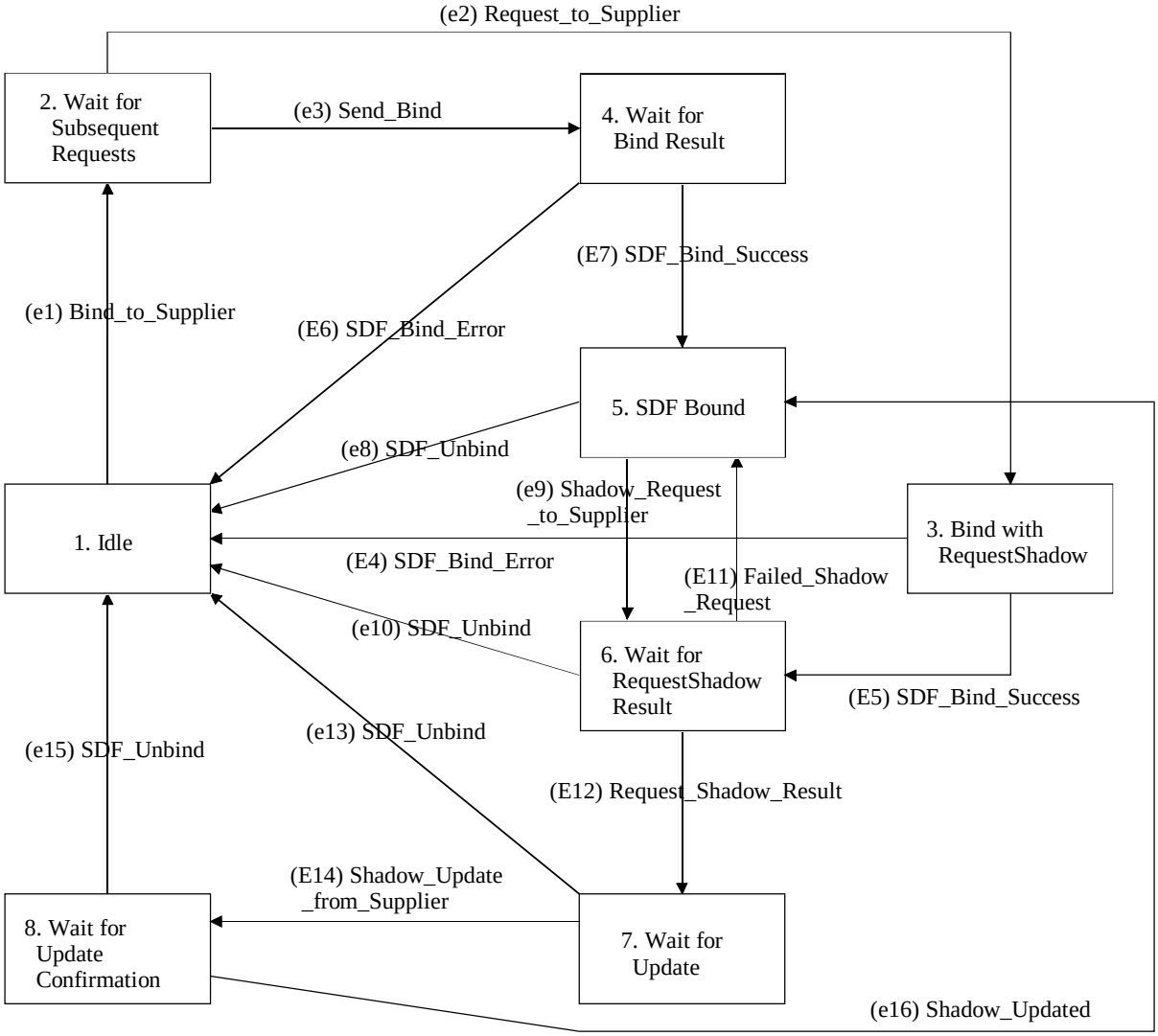


Figure 3-6/B-IF4.28 SDF FSM for a copy consumer in case of consumer-initiated (SDSM-ShCCi)

State 1 : "Idle"
There is only one event accepted in this state:

- (e1) Bind_to_Supplier: This is an internal event caused by the request to execute a DSAShadowBind operation. This event causes a transition out of this state to State 2 Wait for Subsequent Requests.

State 2 : “Wait for Subsequent Requests”
In this state, a RequestShadowUpdate operation to be sent with the DSAShadowBind operation (in the same message) to the supplier is expected. The following two events are considered in this state:

- (e2) Request_to_Supplier: This is an internal event caused by the reception of a

RequestShadowUpdate operation. This event causes a TCAP message containing the DSAShadowBind and RequestShadowUpdate operations to be sent to the supplier SDF. This event causes a transition out of this state to State 3 Bind with RequestShadow;

- (e3) Send_Bind: This is an internal event caused by the reception of a delimiter that indicates the reception of the last operation to be sent or the expiration of a timer. Once the internal event is received, a TCAP message containing the DSAShadowBind operation is sent to the supplier SDF. This event causes a transition out of this state to State 4 Wait for Bind Results.

State 3: "Bind with RequestShadow"

In this state, a DSAShadowBind result is expected from the supplier SDF. Two events are considered in this state:

- (E4) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the supplier SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 Idle;
- (E5) SDF_Bind_Success: This is an external event, caused by the reception of a DSAShadowBind result. This indicates a successful completion of the DSAShadowBind operation previously issued to the supplier SDF. This event causes a transition out of this state to State 6 Wait for RequestShadow Result.

State 4 : "Wait for Bind Result"

In this state, a DSAShadowBind result is expected from the supplier SDF. Two events are considered in this state:

- (E6) SDF_Bind_Error: This is an external event, caused by the failure of the DSAShadowBind operation previously issued to the supplier SDF. A DSAShadowBind error has been returned. This event causes a transition out of this state to State 1 Idle;
- (E7) SDF_Bind_Success: This is an external event, caused by the successful completion of the DSAShadowBind operation previously issued to the supplier SDF. This event causes a transition out of this state to State 5 SDF Bound.

State 5 : "SDF Bound"

In this state, the consumer SDF is ready to send a RequestShadowUpdate operation to the supplier SDF. Two events are considered in this state:

- (e8) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g., during a user's release procedure) or causing the issuing of the in-DSAShadowUnbind operation. This event causes a transition out of this state to State 1 Idle;
- (e9) Shadow_Request_to_Supplier: This is an internal event, caused by the sending of a RequestShadowUpdate operation to the supplier SDF. This event causes a transition out of this state to State 6 Wait for RequestShadow Result.

State 6 : "Wait for RequestShadow Result"

In this state, the consumer SDF has sent a RequestShadowUpdate operation and waits for the answer from the supplier SDF. Three events are considered in this state:

- (e10) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (E11) Failed_Shadow_Request: This is an external event, caused by the reception of an error to the previously issued RequestShadowUpdate operation. This event causes a transition out of this state to State 5 SDF Bound;
- (E12) Request_Shadow_Result: This is an external event, caused by the reception of the response to the previously issued RequestShadowUpdate operation from the supplier SDF.

This event causes a transition out of this state to State 7 Wait for Update.

State 7 : “Wait for Update”

In this state, the consumer SDF has received a RequestShadowUpdate result and waits for an UpdateShadow operation from the supplier SDF. Two events are considered in this state:

- (e13) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (E14) Shadow_Update_from_Supplier: This is an external event caused by the reception of an UpdateShadow operation issued from the supplier SDF. This event causes a transition out of this state to State 8 Wait for Update Confirmation.

State 8 : "Wait for Update Confirmation"

In this state, the consumer SDF has received an UpdateShadow operation from the supplier SDF and processes to update the copy. Two events are considered in this state:

- (e15) SDF_Unbind: This is an internal event, caused by the need to cancel the "authenticated association" established between the two SDFs (e.g., during a user's release procedure). This event causes a transition out of this state to State 1 Idle;
- (e16) Shadow_Updated: This is an internal event, caused by the completion of an UpdateShadow operation and the sending of the response to it. This event causes a transition out of this state to State 5 SDF Bound.

3.1.1.4.1.2 DF state transition models for chaining

As for the chaining procedure, an SDF can act as a chaining initiator and as a chaining terminator. Therefore, there are two FSMs as described below.

In the following FSMs, the possibility of sending the DSABind operation together with other DSP operations in one TC message is taken into considerations.

SDF state transition models for chaining initiation (SDSM-ChI)

The Finite State Machine for an SDFs interaction with another SDF when acting as a chaining initiator is depicted in Figure 3-7/B-IF4.28.

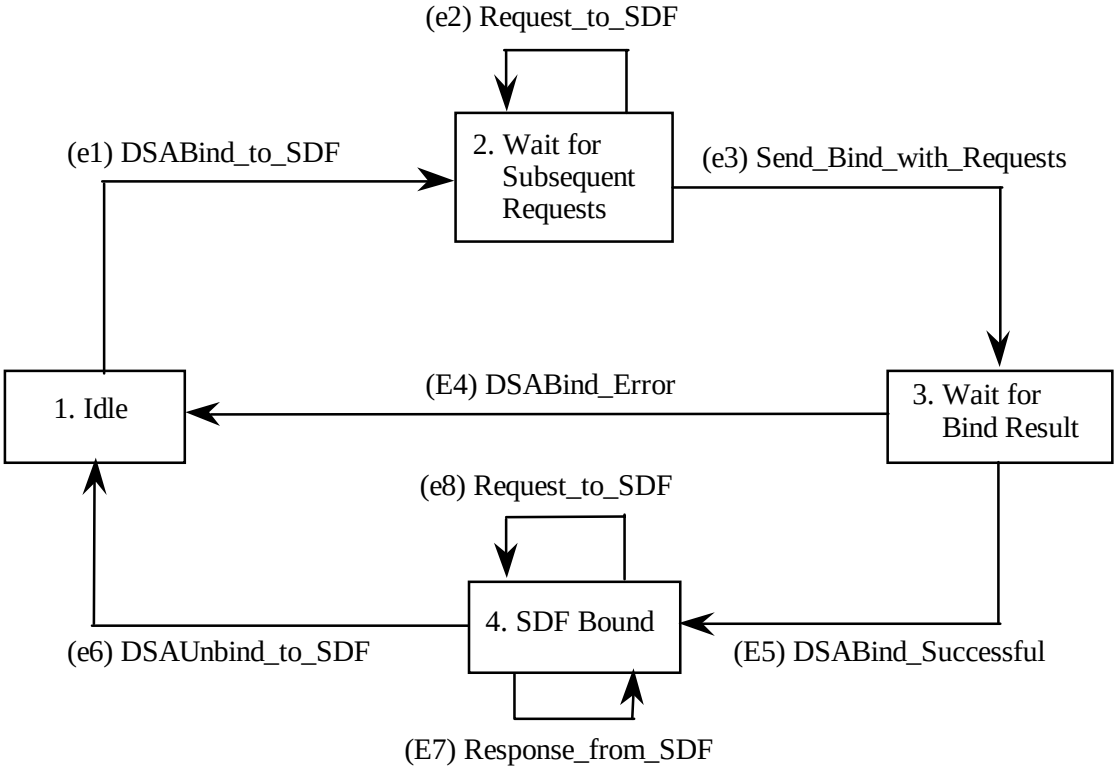


Figure 3-7/B-IF4.28 SDF/SDF chaining initiator finite state machine (SDSM-ChI)

State 1 : "Idle"
The only event accepted in this state is:

- (e1) DSABind_to_SDF: This is an internal event causing the sending of a DSABind operation to the SDSM-ChT. This event causes a transition out of this state to State 2 Wait for Subsequent Requests.

State 2 : “Wait for Subsequent Requests”
In this state, subsequent operations to be sent with the DSABind operation (in the same message) to the SDSM-ChT are expected. The following two events are considered in this state:

- (e2) Request_to_SDF: This is an internal event causing the sending of an operation. It involves one of the following operations:
- chainedSearch;
- chainedAddEntry;
- chainedRemoveEntry;
- chainedModifyEntry;
- chainedExecute

  The operation is buffered until the reception of a delimiter (or a timer expiration). This event causes a transition to the same state.;
- (e3) Send_Bind_with_Requests: This is an internal event caused by the reception of a delimiter, that indicates the reception of the last operation to be sent. Once the delimiter is received, a message containing those arguments of the DSABind operation and other operations, if any, is sent to the SDSM-ChT. This event causes a transition out of this state to State 3 Wait for Bind Results.

State 3 : "Wait for Bind Result"

In this state, a DSABind request has been sent to the SDSM-ChT. The SDSM-ChT is performing the SDF access control procedures associated with the DSABind operation (e.g., access authentication). Two events are considered in this state:

- (E4) DSABind_Error: This is an external event caused by the failure of the DSABind operation previously issued to the SDSM-ChT. This event causes a transition out of this state to State 1 Idle;
- (E5) DSABind_Successful: This is an external event caused by the reception of the DSABind confirmation for the DSABind operation previously issued to the SDSM-ChT. This event causes a transition out of this state to State 4 SDF Bound.

State 4 : "SDF Bound"

In this state, the access of the SDSM-ChI to the SDSM-ChT was authorized, chained operations can be sent to the SDSM-ChT, and results of chained operations coming from the SDSM-ChT are accepted. Three events are considered in this state:

- (e6) DSAUnbind_to_SDF: This is an internal event, causing the sending of the in-DSAUnbind operation to the SDSM-ChT. The SDF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to State 1 Idle;
- (E7) Response_from_SDF: This is an external event, caused either by the reception results of the operations previously issued by the SDSM-ChI or reception of a referral from the SDSM-ChT. The SDSM-ChI remains in the same state;
- (e8) Request_to_SDF: This is an internal event, causing the sending of a chained operation to the SDSM-ChT. It involves one of the following operations:
- chainedSearch;
- chainedAddEntry;
- chainedRemoveEntry;
- chainedModifyEntry;
- chainedExecute

The SDSM-ChI remains in the same state.

SDF state transition models for chaining termination (SDSM-ChT)

The Finite State Machine for an SDFs interaction with another SDF when acting as a chaining terminator is depicted in Figure 3-8/B-IF4.28.

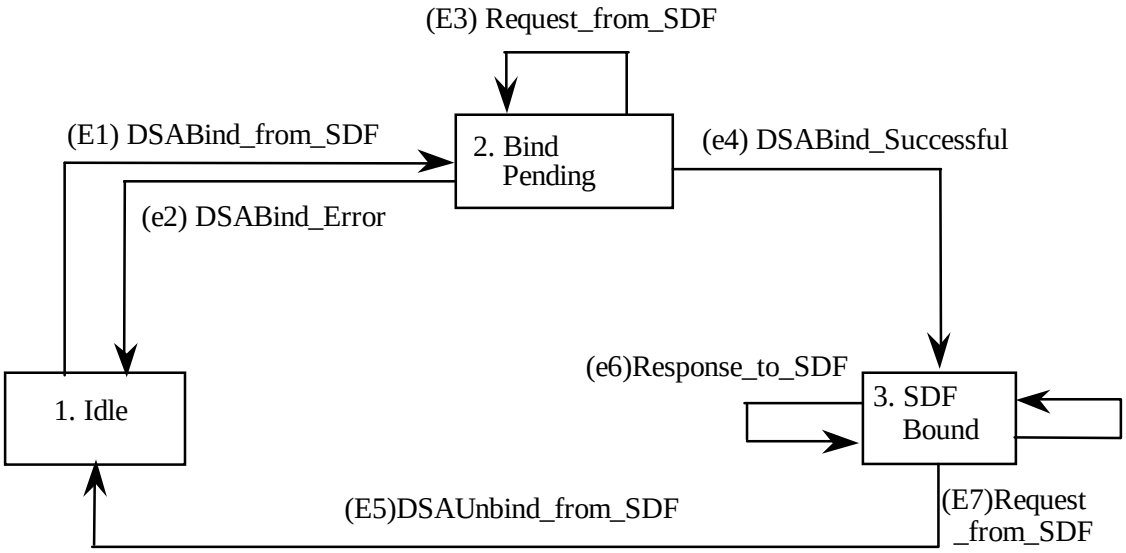


Figure 3-8/B-IF4.28 SDF/SDF chaining terminator finite state machine (SDSM-ChT)

State 1 : "Idle"
The only event accepted in this state is:

- (E1) DSABind_from_SDF: This is an external event caused by the reception of a DSABind operation from an SDSM-ChI. This event causes a transition out of this state to State 2 Bind Pending.

State 2 : "Bind Pending"
In this state, a DSABind request has been received from the SDSM-ChI. The SDSM-ChT is performing the SDF access control procedures associated with the DSABind operation (e.g., access authentication). Three events are considered in this state:

- (e2) DSABind_Error: This is an internal event caused by the failure of the DSABind operation previously issued from the SDSM-ChI. This event causes a transition out of this state to State 1 Idle, and a Bind error is returned to the SDSM-ChI;
- (E3) Request_from_SDF: This is an external event, caused by the reception of operations before the result from the DSABind operation is determined. It involves one of the following operations:
- chainedSearch;
- chainedAddEntry;
- chainedRemoveEntry;
- chainedModifyEntry;
- chainedExecute

The operations are stored and the SDSM-ChT remains in the same state. When a transition occurs to another state, the operations are re-examined as if they had occurred in that state: and

- (e4) DSABind_Successful: This is an internal event caused by the successful completion of the DSABind operation previously issued from the SDSM-ChI. This event causes a transition out of this state to State 3 SDF Bound.

State 3 : "SDF Bound"
In this state, the access of the SDSM-ChI to the SDSM-ChT was authorized and chained operations coming from the SDSM-ChI are accepted. Besides waiting for requests from the SDSM-ChI, the

SDSM-ChT can send in this state responses to previously issued operations. Three events are considered in this state:

- (E5) DSAUnbind_from_SDF: This is an external event, caused by the reception of the in-DSAUnbind operation from the SDSM-ChI. The SDF/SDF association is ended and all associated resources are released. This event causes a transition out of this state to State 1 Idle;
- (e6) Response_to_SDF: This is an internal event, caused either by the completion of operations previously issued by the SDSM-ChI or generation of a referral to the SDSM-ChI. Responses/referrals are sent to the SDSM-ChI. The SDSM-ChT remains in the same state;
- (E7) Request_from_SDF: This is an external event, caused by the reception of a request from the SDSM-ChI. It involves one of the following operations:
- chainedSearch;
- chainedAddEntry;
- chainedRemoveEntry;
- chainedModifyEntry;
- chainedExecute

The SDSM-ChT remains in the same state.

3.2 Error procedures

This subclause defines the generic error procedures for the INAP.

3.2.1 Operation related error procedures

The following subclauses define the generic error handling for the operation related errors. The errors are defined as operation errors in clause 2.
Errors which have a specific procedure for an operation are described in subclause 3.1 with the detailed procedure of the related operation.

3.2.1.1 Attribute Error

3.2.1.1.1 General description

3.2.1.1.1.1 Error description

This error is sent by the SDF to another SDF to report an attribute related problem. The conditions under which an attribute error is to be issued are defined in 12.4/ITU-T Recommendation X.511(1993).

3.2.1.1.1.2 Argument description

The attribute error parameter and problem codes are specified in 12.4/ITU-T Recommendation X.511(1993).

3.2.1.1.2 Operations SDF->SDF

ChainedSearch
ChainedAddEntry
ChainedExecute
ChainedModifyEntry
Procedures at invoking entity (SDF)
A) Sending Operation
Precondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
B) Receiving Error
Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound
This error is reported to the SCF.
Procedures at responding entity (SDF)
A) Receiving Operation
Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound
B) Returning Error
Precondition: SDSM-ChT state 3 SDF Bound

Postcondition: SDSM-ChT state 3 SDF Bound
The SDF could not perform the operation due to an attribute problem and therefore sends an Attribute error to the other SDF. After returning the error, no further error treatment is performed.

3.2.1.2 Name Error

3.2.1.2.1 General description

3.2.1.2.1.1 Error description

This error is sent by the SDF to another SDF to report an problem related to the name of the object. The conditions under which a name error is to be issued are defined in 12.5/ITU-T Recommendation X.511(1993).

3.2.1.2.1.2 Argument description

The name error parameter and problem codes are specified in 12.5/ITU-T Recommendation X.511(1993).

3.2.1.2.2 Operations SDF->SDF

ChainedSearch
ChainedAddEntry
ChainedExecute
ChainedRemoveEntry
ChainedModifyEntry
Procedures at invoking entity (SDF)
A) Sending Operation
Precondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
B) Receiving Error
Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound
This error is reported to the SCF.
Procedure at Responding Entity (SDF)
A) Receiving Operation
Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound
B) Returning Error
Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound
The SDF could not perform the operation due to a name problem and therefore sends an Name error to the other SDF. After returning the error, no further error treatment is performed.

3.2.1.3 Security

3.2.1.3.1 General description

3.2.1.3.1.1 Error description

This error is sent by the SDF to another SDF to report a problem in carrying out an operation for security reasons. The conditions under which a security error is to be issued are defined in 12.7/ITU-T Recommendation X.511(1993).

3.2.1.3.1.2 Argument description

The security error parameter and problem codes are specified in 12.7/ITU-T Recommendation X.511(1993).

3.2.1.3.2 Operations SDF->SDF

ChainedSearch
ChainedAddEntry
ChainedExecute
ChainedRemoveEntry
ChainedModifyEntry

Procedures at invoking entity (SDF)
A) Sending Operation
Precondition: SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests Postcondition:
SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests
B) Receiving Error
Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound
This error is reported to the SCF in case of chained operations.
Procedures at responding entity (SDF)
A) Receiving Operation
Precondition: SDSM-ChT state 3 SDF Bound Postcondition: SDSM-ChT
state 3 SDF Bound
B) Returning Error
Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound
The SDF could not perform the operation for security reasons and therefore sends a Security error to the other SDF. After returning the error, no further error treatment is performed.

3.2.1.4 Service

3.2.1.4.1 General description

3.2.1.4.1.1 Error description

This error is sent by the SDF to the SCF or another SDF to report a problem related to the provision

of the service. The conditions under which a service error is to be issued are defined in 12.8/ITU-T Recommendation X.511(1993).

3.2.1.4.1.2 Argument description

The Service error parameter and problem codes are specified in 12.8/ITU-T Recommendation X.511(1993).

3.2.1.4.2 Operations SDF->SDF

ChainedSearch
ChainedAddEntry
ChainedExecute
ChainedRemoveEntry
ChainedModifyEntry

Procedures at invoking entity (SDF)
A) Sending Operation
Precondition: SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests Postcondition:
SDSM-ChI state 4 SDF Bound or
state 2 Wait for subsequent requests

B) Receiving Error
Precondition: SDSM-ChI state 4 SDF Bound Postcondition: SDSM-ChI
state 4 SDF Bound
This error is reported to the SCF in case of chained operations.
Procedures at responding entity (SDF)
A) Receiving Operation
Precondition: SDSM-ChT state 3 SDF Bound Postcondition: SDSM-ChT
state 3 SDF Bound
B) Returning Error
Precondition: SDSM-ChT state 3 SDF Bound Postcondition: SDSM-ChT
state 3 SDF Bound
The SDF could not perform the operation due to a service related problem and sends a Service error to the other SDF. After returning the error, no further error treatment is performed.

3.2.1.5 Update

3.2.1.5.1 General description

3.2.1.5.1.1 Error description

This error is sent by the SDF to another SDF to report a problem related to attempts to add, delete, or modify information in the SDF. The conditions under which an update error is to be issued are defined in 12.9/ITU-T Recommendation X.511(1993).

3.2.1.5.1.2 Argument description

The update error parameter and problem codes are specified in 12.9/ITU-T Recommendation X.511(1993).

3.2.1.5.2 Operations SDF->SDF

ChainedAddEntry
ChainedExecute
ChainedRemoveEntry
ChainedModifyEntry
Procedures at invoking entity (SDF)
A) Sending Operation
Precondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
B) Receiving Error
Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound
This error is reported to the SCF.
Procedures at responding entity (SDF)
A) Receiving Operation
Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound
B) Returning Error
Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound
The SDF could not perform the operation due to a problem related to the addition, deletion or modification of information and sends an Update error to the other SDF. After returning the error, no further error treatment is performed.

3.2.1.6 DSAReferral

3.2.1.6.1 General description

3.2.1.6.1.1 Error description

This error is sent by the SDF to another SDF to report a problem related to chaining to a chained operation. The conditions under which a DSAReferral error is to be issued are defined in 8.3/ITU-T Recommendation X.518(1993).

3.2.1.6.1.2 Argument description

The DSAReferral error parameter and problem codes are specified in 13.2/ITU-T Recommendation X.518(1993).

3.2.1.6.2 Operations SDF->SDF

ChainedAddEntry
ChainedExecute
ChainedRemoveEntry
ChainedModifyEntry
ChainedSearch
Procedures at invoking entity (SDF)
A) Sending Operation
Precondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
Postcondition: SDSM-ChI state 4 SDF Bound or
SDSM-ChI state 2 Wait for subsequent requests
B) Receiving Error
Precondition: SDSM-ChI state 4 SDF Bound
Postcondition: SDSM-ChI state 4 SDF Bound
After receiving a DSAReferral error, the SDF can continue to execute the operation by accessing another SDF which is indicated by this error.
Procedures at responding entity (SDF)
A) Receiving Operation
Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound
B) Returning Error
Precondition: SDSM-ChT state 3 SDF Bound
Postcondition: SDSM-ChT state 3 SDF Bound

After returning the error, no further error treatment is performed.

3.2.1.7 Shadow

3.2.1.7.1 General description

3.2.1.7.1.1 Error description

This error is sent by the SDF to another SDF to report a problem related to shadowing. The conditions under which a shadow error is to be issued are defined in 12/ITU-T Recommendation X.525(1993).

3.2.1.7.1.2 Argument description

The shadow error parameter and problem codes are specified in 11.3.3/ITU-T Recommendation X.525(1993).

3.2.1.7.2 Operations SDF->SDF

InCoordinateShadowUpdate
InRequestShadowUpdate
InUpdateShadow
Procedures at Supplier Entity (SDF)

A-1) Sending Operation
Precondition: SDSM-ShSSi (in case of supplier initiated)
state 8 SDF Bound (in case of CoordinateShadowUpdate) or
state 10 Wait for update (in case of UpdateShadow)
SDSM-ShSCi (in case of consumer initiated)
state 5 Wait for update (in case of UpdateShadow)
Postcondition: SDSM-ShSSi (in case of supplier initiated)
state 9 Wait for coordination result (in case of CoordinateShadowUpdate) or
state 11 Wait for update confirmation (in case of UpdateShadow)
SDSM-ShSCi (in case of consumer initiated)
state 6 Wait for update confirmation (in case of UpdateShadow)
B-1) Receiving Error
Precondition: SDSM-ShSSi (in case of supplier initiated)
state 9 Wait for coordination result (in case of CoordinateShadowUpdate) or
state 11 Wait for update confirmation (in case of UpdateShadow)
SDSM-ShSCi (in case of consumer initiated)
state 6 Wait for update confirmation (in case of UpdateShadow)
Postcondition: SDSM-ShSSi (in case of supplier initiated)
state 8 SDF Bound (in case of CoordinateShadowUpdate) or
state 8 SDF Bound (in case of UpdateShadow)
SDSM-ShSCi (in case of consumer initiated)
state 3 SDF Bound (in case of UpdateShadow)
A-2) Receiving Operation
Precondition: SDSM-ShSCi (in case of consumer initiated)
state 3 SDF Bound (in case of RequestShadowUpdate)
Postcondition: SDSM-ShSCi (in case of consumer initiated)
state 4 Wait for RequestShadow result (in case of RequestShadowUpdate)
B-2) Returning Error
Precondition: SDSM-ShSCi (in case of consumer initiated)
state 4 Wait for RequestShadow result (in case of RequestShadowUpdate)
Postcondition: SDSM-ShSCi (in case of consumer initiated)
state 3 SDF Bound (in case of RequestShadowUpdate)
After receiving or returning the error, no further error treatment is performed.
Procedures at Consumer Entity (SDF)
A-1) Sending Operation
Precondition: SDSM-ShCCi (in case of consumer initiated)
state 5 SDF Bound (in case of RequestShadowUpdate)
Postcondition: SDSM-ShCCi (in case of consumer initiated)
state 6 Wait for RequestShadow result ((in case of RequestShadowUpdate)
B-1) Receiving Error
Precondition: SDSM-ShCCi (in case of consumer initiated)
state 6 Wait for RequestShadow result (in case of RequestShadowUpdate)

Postcondition: SDSM-ShCCi (in case of consumer initiated)
state 5 SDF Bound (in case of RequestShadowUpdate)
A-2) Receiving Operation
Precondition: SDSM-ShCSi (in case of supplier initiated)
state 3 SDF Bound (in case of CoordinateShadowUpdate) or
state 5 Wait for update (in case of UpdateShadow)
SDSM-ShCCi (in case of consumer initiated)
state 7 Wait for update (in case of UpdateShadow)
Postcondition: SDSM-ShCSi (in case of supplier initiated)
state 4 Wait for coordination result (in case of
CoordinateShadowUPdate) or
state 6 Wait for update confirmation (in case of UpdateShadow)
SDSM-ShCCi (in case of consumer initiated)
state 8 Wait for update confirmation (in case of UpdateShadow)
B-2) Returning Error
Precondition: SDSM-ShCSi (in case of supplier initiated)
state 4 Wait for coordination result (in case of
CoordinateShadowUPdate) or
state 6 Wait for update confirmation (in case of UpdateShadow)
SDSM-ShCCi (in case of consumer initiated)
state 8 Wait for update confirmation (in case of UpdateShadow)
Postcondition: SDSM-ShCSi (in case of supplier initiated)
state 3 SDF Bound (in case of CoordinateShadowUpdate) or
state 3 SDF Bound (in case of UpdateShadow)
SDSM-ShCCi (in case of consumer initiated)
state 5 SDF Bound (in case of UpdateShadow)
After receiving or returning the error, no further error treatment is performed.

3.2.1.8 ExecutionError

3.2.1.8.1 General description

3.2.1.8.1.1 Error description

ExecutionError is an error sent from the SDF to another SDF, indicating that the request to execute an entry method has failed. The failure can be due to an incorrect input value or the failure of an internal operation or data access logic associated with the execution of the entry method.

3.2.1.8.1.2 Argument description

```
PARAMETER   OPTIONALLY-PROTECTED
                        SET {
                                problem [0] ExecutionProblem },
                                COMPONENTS OF CommonResults },
```

DIRQOP.&dirErrors-QOP{@dirqop} }

```
ExecutionProblem  ::=  INTEGER {
      missingInputValues (1),
      executionFailure (2) }
```

3.2.1.8.2 Operations SDF->SDF

ChainedExecute
Procedure at Invoking Entity (SDF)
A) Sending Operation

Precondition:	SDSM-ChI	state 4	SDF Bound or
		state 2	Wait for subsequent requests
Postcondition:	SDSM-ChI	state 4	SDF Bound or
		state 2	Wait for subsequent requests

B) Receiving Error

Precondition:	SDSM-ChI	state 4	SDF Bound
Postcondition:	SDSM-ChI	state 4	SDF Bound

This error is reported to the SCF in case of chained operations.

Procedure at Responding Entity (SDF)

A) Receiving Operation

Precondition:	SDSM-ChT	state 3	SDF Bound
Postcondition:	SDSM-ChT	state 3	SDF Bound

B) Returning Error

Precondition:	SDSM-ChT	state 3	SDF Bound
Postcondition:	SDSM-ChT	state 3	SDF Bound

The SDF could not perform the operation and therefore sends an executionError to the other SDF. After returning the error, no further error treatment is performed.

3.3 Detailed Operation Procedure

3.3.1 Chained Operations procedure

3.3.1.1 dSABind

3.3.1.1. 1 General description

The X.500 'dSABind' operation is used by the invoking SDF to create an authenticated association between an invoking SDF and an responding SDF to enable distributed processing of operations on behalf of the end user. It carries the authentication information of the end user if any. For a full description of the dSABind operation, see 11.1/X.518. A full description of distributed operation procedures can be found in X.518.

3.3.1.1.1.1 Parameters

See 11.1/X.518.

3.3.1.1.1.2 Invoking entity (SDF)

3.3.1.1.2.1 Normal procedure

SDF Preconditions:

(1) A request for data access operation from an end user has arrived which needs to be chained to a remote SDF for processing and no chaining association exists between the SDFs for the originating end user.

(2) SDSM-ChI: "Idle"

SDF Postconditions:
(1) SDSM-ChI: "SDF Bound" in case of success
(2) SDSM-ChI: "Idle" in case of failure.
(3) A chaining association exists between the invoking SDF and the responding SDF for the end user.

When the SDSM-ChI is in the state "Idle" and a need of the service logic to interrogate an SDF exists, an internal event occurs. This event, called (e1) DSABind_to_SDF, causes a transition to the state "Wait for Subsequent Requests" and other operations are awaited. Until the application process has indicated by a delimiter that the DSABind should be sent, the SDSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The reception of the delimiter causes a transition to the state "Wait for Bind Result" through the internal event (e3) Send_Bind_with_Requests. The operation is sent to the responding SDF . The SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E5) DSABind_Successful) to the DSABind operation previously issued to the responding SDF causes a transition of the invoking SDF to the state "SDF Bound" if the result of the DSABind operation is positive. Otherwise the reception of an error ((E4) DSABind_Error) moves back the SDSM-ChI to the state "Idle".

3.3.1.1.2.2 Error handling

Generic error handling for the operation related errors is described in sub-clause 3.2 and ITU-T Rec. 13/ X.518 and the TCAP services that are used for reporting operating errors are described in sub-clause 2.8.

3.3.1.1.3 Responding Entity (SDF)

3.3.1.1.3.1 Normal procedure

SDF Preconditions:
1) SDSM-ChT: "Idle"

SDF Postconditions:
(1) SDSM-ChT: "SDF Bound" (success)
(2) SDSM-ChT: "Idle" (failure)

The SDSM-ChT is initially in the state "Idle". After accepting the external event (E1) DSABind_from_SDF caused by the reception of a 'DSABind' operation from the invoking SDF , a transition to state "Bind Pending" occurs. The responding SDF performs the DSABind operation according to the contents of the dSABind argument. Once the responding SDF has completed the 'DSABind' operation, the result or error indication is returned to the invoking SDF. The responding SDF returns to the state "Idle" if the DSABind fails or to the state "SDF Bound" if the DSABind is successful. Should the Bind request succeed, the result returned may consist of credentials of the dSABindResult. These credentials allow the user to establish the identity of the Directory. They allow information identifying the responding SDF (that is directly providing the Directory service) to be conveyed to the invoking SDF. The credentials are of the same form as those supplied by the user.

3.3.1.1.3.2 Error handling

Generic error handling for the operation related errors is described in sub-clause 3.2 and ITU-T Rec. 13/X.518 and the TCAP services that are used for reporting operating errors are described in sub-clause 2.8.

3.3.1.2 in- DSAUnbind procedure

3.3.1.2.1 General description

The in-DSAUnbind operation is used by the invoking SDF to end an authenticated chaining association between an invoking SDF and a responding SDF on behalf of the end user.

3.3.1.2.1.1 Parameters

None

3.3.1.2.2 Invoking entity (SDF)

3.3.1.2.2.1 Normal procedure

SDF Preconditions:

(1) An Unbind request has been received by the invoking SDF indicating that the end user association which required operation chaining has been released.
(2) SDSM-ChI: "SDF Bound"

SDF Postconditions:
(1) SDSM-ChI: "Idle"

The SDSM-ChI has previously initiated a successful DSABind operation to the responding SDF directory. It is in ? state "SDF Bound". The invoking SDF has received an indication that the authenticated chained access to the responding SDF is to be terminated. It issues an in-DSAUnbind operation ((e6) DSAUnbind_to_SDF) that causes the SDSM-ChI to transit back to the state "Idle".

3.3.1.2.2.2 Error handling

The 'Unbind' operation does not have operation related errors.

3.3.1.2.3 Responding entity (SDF)

3.3.1.2.3.1 Normal procedure

SDF Preconditions:
(1) SDSM-ChT: "SDF Bound"

SDF Postconditions:
(1) SDSM-ChT: "Idle"

AdSABind operation was previously issued and the SDSM-ChT is in the state "SDF Bound" waiting for a request from the invoking SDF and/or performing an operation. The reception of the in-DSAUnbind operation causes a transition to the state "Idle" with the transition (E5) DSAUnbind_from_SDF.

3.3.1.2.3.2 Error handling

The 'Unbind' operation does not have operation related errors

3.3.1.3 Chained Operations

3.3.1.3.1 chainedModifyEntry procedure

3.3.1.3.1.1 General description

The X.500 'chainedModifyEntry' operation is used to request remote processing of an ModifyEntry

operation on behalf of an end user. For a full description of the chainedModifyEntry operation, see ITU-T Rec. X.518 subclause 12.1.

3.3.1.3.1.1.1 Parameters

See ITU-T Rec. X.518 subclause 12.1.

3.3.1.3.1.2 Invoking entity (SDF)

3.3.1.3.1.2.1 Normal procedure

SDF Precondition:

(1) The invoking SDF has received a request to perform an ModifyEntry operation for an end user which requires the operation to be chained to a responding SDF for processing.
(2) SDSM-ChI: "SDF Bound" or "Wait for Subsequent Requests"

SDF Postcondition:

(1) SDSM-ChI: "SDF Bound"
(2) A response to the request to perform an ModifyEntry operation has been received by the invoking SDF.

When the SDSM-ChI is in the state "Wait for Subsequent Requests" and has received a request to modify an entry in the service data which requires processing in the responding SDF, an internal event ((e2) Request_to_SDF) occurs. Until the application process has indicated with a delimitor (or a timer expiry) that the operation should be sent, the SDSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the responding SDF in a message containing a Bind argument. The SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E5) DSABind_Successful or (E4) DSABind_Error) to the Bind operation previously issued to the SDF causes a transition of the SDF to the state "SDF Bound" or to the state "Idle". When the SDSM-ChI has moved to state "Idle", the ModifyEntry operation was discarded. In the State "SDF Bound", the response of the chainedModifyEntry operation ((E7) Response_from_SDF) causes a transition of the SDF to the same state ("SDF Bound"). It may be either the result of the chainedModifyEntry operation or an error. This response will be returned to the originating end user.

When the SDSM-ChI is in the state "SDF Bound" and has received a request to modify an entry in the service data which requires processing in the responding SDF, an internal event occurs. This event, called (e8) Request_to_SDF causes a transition to the same state "SDF Bound" and the SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E7) Response_from_SDF) to the chainedModifyEntry operation previously issued to the responding SDF causes a transition of the invoking SDF to the same state "SDF Bound". The response from the responding SDF may be either the result of the chainedModifyEntry operation or an error. This response will be returned to the originating end user.

3.3.1.3.1.2.2 Error handling

Generic error handling for the operation related errors is described in sub-clause 3.2 and ITU-T Rec. X.518 clause 13 and the TCAP services that are used for reporting operating errors are described in sub-clause 2.8.

3.3.1.3.1.3 Responding entity (SDF)

3.3.1.3.1.3.1 Normal procedure

SDF Precondition:

(1) SDSM-ChT: "SDF Bound"or "Bind Pending"

SDF Postcondition:

(1) SDSM-ChT "SDF Bound"

When the responding SDF is in the state "Bind Pending", the external event (E3) Request_from_SDF caused by the reception of a 'chainedModifyEntry' operation from the invoking SDF occurs. The responding SDF does not proceed with processing the operation until the DSABind operation has been successfully executed. It remains in the same state. If the DSABind fails then the operation is discarded. If the DSABind operation succeeds then the chainedModifyEntry operation is processed by the responding SDF.

When the responding SDF is in the state "SDF Bound", the external event (E7) Request_from_SDF caused by the reception of a 'chainedModifyEntry' operation from the invoking SDF occurs. This operation is processed by the responding SDF.

The responding SDF may process the ChainedModifyEntry operation in one of two ways:

(1) if the invoking SDF is located in another network then the responding SDF may chain the operation to another SDF within the same network as the responding SDF.

(2) the operation is processed according to the actions described in the ModifyEntry procedure.

After the responding SDF has finished processing the operation, any results or errors from the operation are returned to the invoking SDF. The sending of this response corresponds to the event (e6) Response_to_SDF.

3.3.1.3.1.3.2 Error handling

Generic error handling for the operation related errors is described in sub-clause 3.2 and ITU-T Rec. X.518 clause 13 and clause and the TCAP services that are used for reporting operating errors are described in sub-clause 2.8.

3.3.1.3.2 chainedExecute procedure

3.3.1.3.2.1 General description

The 'chainedExecute' operation is used to request remote processing of an Execute operation on behalf of an end user. For a full description of the operation chaining mechanism, see ITU-T Rec. X.518 subclause 12.1.

3.3.1.3.2.1.1 Parameters

See ITU-T Rec. X.518 subclause 12.1 and Execute operation parameters

3.3.1.3.2.2 Invoking entity (SDF)

3.3.1.3.2.2.1 Normal procedure

SDF Precondition:

(1) The invoking SDF has received a request to perform an Execute operation for an end user which requires the operation to be chained to a responding SDF for processing.
(2) SDSM-ChI: "SDF Bound" or "Wait for Subsequent Requests"

SDF Postcondition:

(1) SDSM-ChI: "SDF Bound"
(2) A response to the request to perform an Execute operation has been received by the invoking SDF.

When the SDSM-ChI is in the state "Wait for Subsequent Requests" and has received a request to add an entry in the service data which requires processing in the responding SDF, an internal event ((e2) Request_to_SDF) occurs. Until the application process has indicated with a delimitor (or a timer expiry) that the operation should be sent, the SDSM-ChI remains in the state "Wait for Subsequent Requests" and the operation is not sent. The operation is sent to the responding SDF in a message containing a Bind argument. The SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E5) DSABind_Successful or (E4) Bind_Error) to the Bind operation previously issued to the SDF causes a transition of the SDF to the state "SDF Bound" or to the state "Idle". When the SDSM-ChI has moved to state "Idle", the Execute operation was discarded. In the State "SDF Bound", the response of the chainedExecute operation ((E7) Response_from_SDF) causes a transition of the SDF to the same state ("SDF Bound"). It may be either the result of the chainedExecute operation or an error. This response will be returned to the originating end user.

When the SDSM-ChI is in the state "SDF Bound" and has received a request to add an entry in the service data which requires processing in the responding SDF, an internal event occurs. This event, called (e8) Request_to_SDF causes a transition to the same state "SDF Bound" and the SDSM-ChI waits for the response from the responding SDF. The reception of the response ((E7) Response_from_SDF) to the chainedExecute operation previously issued to the responding SDF causes a transition of the invoking SDF to the same state "SDF Bound". The response from the responding SDF may be either the result of the chainedExecute operation or an error. This response will be returned to the originating end user.

3.3.1.3.2.2.2 Error handling

Generic error handling for the operation related errors is described in sub-clause 3.2 and ITU-T Rec. X.518 clause 13 and the TCAP services that are used for reporting operating errors are described in sub-clause 2.8.

3.3.1.3.2.3 Responding entity (SDF)

3.3.1.3.2.3.1 Normal procedure

SDF Precondition:

(1) SDSM-ChT: "SDF Bound"or "Bind Pending"

SDF Postcondition:

(1) SDSM-ChT "SDF Bound"

When the responding SDF is in the state "Bind Pending", the external event (E3) Request_from_SDF caused by the reception of a 'chainedExecute' operation from the invoking SDF occurs. The responding SDF does not proceed with processing the operation until the DSABind operation has been successfully executed. It remains in the same state. If the DSABind fails then the operation is discarded. If the DSABind operation succeeds then the chainedExecute operation is processed by the responding SDF.

When the responding SDF is in the state "SDF Bound", the external event (E7) Request_from_SDF caused by the reception of a 'chainedExecute' operation from the invoking SDF occurs. This operation is processed by the responding SDF.

The responding SDF may process the chainedExecute operation in one of two ways:

(1) if the invoking SDF is located in another network then the responding SDF may chain the operation to another SDF within the same network as the responding SDF.
(2) the operation is processed according to the actions described in the Execute procedure.

After the responding SDF has finished processing the operation, any results or errors from the operation are returned to the invoking SDF. The sending of this response corresponds to the event (e6) Response_to_SDF.

3.3.1.3.2.3.2 Error handling

Generic error handling for the operation related errors is described in sub-clause 3.2 and ITU-T Rec. X.518 clause 13 and clause and the TCAP services that are used for reporting operating errors are described in sub-clause 2.8.

3.3.2 Shadow Operations procedure

3.3.2.1 DSAShadowBind procedure

3.3.2.1.1 General Description

The X.500 "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525.

3.3.2.1.1.1 Parameters

The parameters for the DSA ShadowBind operation are the same as those for the in-DirectoryBind operation specified in subclause 2.4.2.

3.3.2.1.2 Supplier entity (SDF)

3.3.2.1.2.1 Normal Procedure

3.3.2.1.2.1.1 Supplier-initiated DSAShadowBind

3.3.2.1.2.1.1.1 DSAShadowBind sent by itself

SDF Preconditions:
(1) SDSM-ShSSi: “Idle”

SDF Postconditions:

(1) SDSM-ShSSi: “SDF Bound” in case of success
(2) SDSM-ShSSi: “Idle” in case of failure

When the SDSM-ShSSi is in the state "Idle" and a need of providing copies exists, an internal event occurs. This event, called (e1) Bind_to_Consumer, causes a transition to the state "Wait for Subsequent Requests" and other operations are awaited. Until the application process has indicated by a delimiter that the DSAShadowBind should be sent, the SDSM-ShSSi remains in the state "Wait for Subsequent Requests" and the operation is not sent. The reception of the delimiter causes a transition to the state "Wait for Bind Result" through the internal event (e3) Send_Bind. The operation is sent to another SDF (consumer SDF). The SDSM-ShSSi waits for the response from the consumer SDF. The reception of the response ((E14)SDF_Bind_Success) to the “DSAShadowBind” operation previously issued to the consumer SDF causes a transition to the state "SDF Bound" if the result of the 'DSAShadowBind' operation is positive. Otherwise the reception of an error ((E13)SDF_Bind_Error) moves back the SDSM-ShSSi to the state "Idle".

3.3.2.1.2.1.1.2 DSA ShadowBind sent with CoordinateShadowUpdate

SDF Preconditions:

(1) SDSM-ShSSi: "Idle".

SDF Postconditions:

(1) SDSM-ShSSi: "Wait For Coordination Result" in case of success.
(2) SDSM-ShSSi: "Idle" in case of failure.

When the SDSM-ShSSi is in the state "Idle" and a need of providing copies exists, an internal event occurs. This event, called (e1) Bind_to_Consumer, causes a transition to the state " Wait for Subsequent Requests" and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow subsequent operations to be sent at the same time as the DSAShadowBind. The reception of a CoordinateShadowUpdate operation in the " Wait for Subsequent Requests" state through the internal event (e2) Shadow_Coordinate_to Consumer, causes a transition to the state "Bind with Coordinate Shadow".

The reception of the delimiter in the "Bind with Coordinate Shadow" state causes a transition to the state "Bind with CoordinateShadow Only" through the internal event (e5) Send_Bind_with_CoordShadow. This causes the two operations to be sent simultaneously to another SDF(consumer SDF). The SDSM-ShSSi waits for the response from the consumer SDF. The reception of the response ((E7)SDF_Bind_Success) to the "DSAShadowBind" operation previously issued to the consumer SDF causes a transition to the state "Wait for Coordination Result" if the result of the "DSAShadowBind" operation is positive. Otherwise the reception of an error((E6)SDF_Bind_Error) moves back the SDSM-ShSSi to the state "Idle".

3.3.2.1.2.1.1.3 DSAShadowBind sent with CoordinateShadowUpdate and UpdateShadow

SDF Preconditions:

(1) SDSM-ShSSi: "Idle".

SDF Postconditions:

(1) SDSM-ShSSi: "Bound with CoordinateShadow Sent" in case of success.
(2) SDSM-ShSSi: "Idle" in case of failure.

When the SDSM-ShSSi is in the state "Idle" and a need of providing copies exists, an internal event occurs. This event, called (e1) Bind_to_Consumer, causes a transition to the state " Wait for Subsequent Requests" and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow subsequent operations to be sent at the same time as the DSAShadowBind. The reception of a CoordinateShadowUpdate operation in the "Wait for Subsequent Requests" state through the internal event (e2) Shadow_Coordinate_to_Consumer, causes a transition to the state "Bind with Coordinate Shadow".

The subsequent reception of an "UpdateShadow" operation through the internal event (e4) Update to_Consumer causes a transition to the state "Bind with CoordinateShadow and Update". This causes the three operations to be sent simultaneously to another SDF(consumer SDF). The SDSM-ShSSi

waits for the response from the consumer SDF. The reception of the response

((E9)SDF_Bind_Success) to the "DSAShadowBind" operation previously issued to the consumer SDF causes a transition to the state "Bound with Coordinate Shadow Sent" if the result of the "DSAShadowBind" operation is positive. Otherwise the reception of an error((E8)SDF_Bind_Error) moves back the SDSM-ShSSi to the state "Idle".

3.3.2.1.2.1.2 Consumer-initiated DSAShadowBind

SDF Preconditions:

(1) SDSM-ShSCi: "Idle".

SDF Postconditions:

(1) SDSM-ShSCi:"SDF Bound " in case of success.
(2) SDSM-ShSCi:"Idle" in case of failure.

The SDF is initially in the state "Idle". After accepting the external event (E1) Bind_from_Consumer caused by the reception of a "DSAShadowBind" operation from the SDF(consumer SDF), a transition to the state "Wait for Bind Result" occurs. The DSAShadowBind operation may be received at the same time as the RequestShadowUpdate operation. In this case, a transition to the same state through the external event (E3) Request_from_Consumer occurs. The SDF performs the "DSAShadowBind" operation according to the contents of the "DSAShadowBind" argument. Once the SDF has completed the "DSAShadowBind" operation, the result or error indication is returned to the consumer SDF. The SDF returns to the state "Idle" if the "DSAShadowBind" fails or to the state "SDF Bound" if the "DSAShadowBind" is successful.

3.3.2.1.2.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 clause 12 and the TCAP services that are used for reporting operating errors are described in subclause 2.8.

3.3.2.1.3 Consumer entity (SDF)

3.3.2.1.3.1 Normal Procedure

3.3.2.1.3.1.1 Supplier-initiated DSAShadowBind

SDF Preconditions:
(1) SDSM-ShCSi: “Idle”

SDF Postconditions:
(1) SDSM-ShCSi: “SDF Bound” in case of success
(2) SDSM-ShCSi: “Idle” in case of failure

The SDF is initially in the state "Idle". After accepting the external event (E1) Bind_from_Supplier

caused by the reception of a 'DSAShadowBind' operation from the SDF (supplier SDF), a transition to the state "Wait for Bind Result" occurs.

The DSAShadowBind operation may be received at the same time as the RequestShadowUpdate operation. In these cases, a transition to the same state through the external event (E3) Request_from_Supplier occurs once or twice.

The SDF performs the 'DSAShadowBind' operation according to the contents of the 'DSAShadowBind' argument. Once the SDF has completed the 'DSAShadowBind' operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "Idle" if the 'DSAShadowBind' fails or to the state "SDF Bound" if the 'DSAShadowBind' is successful.

3.3.2.1.3.1.2 Consumer-initiated DSAShadowBind

3.3.2.1.3.1.2.1 DSAShadowBind sent by itself

SDF Preconditions:

(1) SDSM-ShCCi: "Idle".

SDF Postconditions:

(1) SDSM-ShCCi: "SDF Bound" in case of success.
(2) SDSM-ShCCi: "Idle" in case of failure.

When the SDSM-ShCCi is in the state "Idle" and a need of requesting updates exists, an internal event occurs. This event, called (e1) Bind_to_Supplier, causes a transition to the state " Wait for Subsequent Requests " and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow these subsequent operations to be sent at the same time as the DSAShadowBind. Until the application process has indicated by a delimiter that the DSAShadowBind should be sent, the SDSM-ShCCi remains in the state "Wait for Subsequent Requests" and the operation is not sent. The reception of the delimiter causes a transition to the state "Wait for Bind Result" through the internal event (e3) Send_Bind. The operation is then sent to another SDF(supplier SDF). The SDSM-ShCCi waits for the response from the supplier SDF. The reception of the response ((E7)SDF_Bind_Success) to the "DSAShadowBind" operation previously issued to the supplier SDF causes a transition to the state "SDF Bound" if the result of the "DSAShadowBind" operation is positive. Otherwise the reception of an error((E6)SDF_Bind_Error) moves back the SDSM-ShCCi to the state "Idle".

3.3.2.1.3.1.2.2 DSAShadowBind sent with RequestShadowUpdate

SDF Preconditions:
(1) SDSM-ShCCi: “Idle”

SDF Postconditions:

(1) SDSM-ShCCi:”Wait for RequestShadow Result” in case of success
(2) SDSM-ShCCi: “Idle” in case of failure

When the SDSM-ShCCi is in the state "Idle" and a need of requesting updates exists, an internal event occurs. This event, called (e1) Bind_to_Supplier, causes a transition to the state "Wait for Subsequent Requests" and other operations are awaited. The sending of the DSAShadowBind is delayed, to allow these subsequent operations to be sent at the same time as the DSAShadowBind. The reception of the RequestShadowUpdate in the " Wait for Subsequent Requests" state through the internal event (e2) Request_to_Supplier, causes a transition to the state "Bind with RequestShadow". The two operations are then sent to another SDF (supplier SDF). The SDSM-ShCCi waits for the response from the supplier SDF. The reception of the response ((E5)SDF_Bind_Success) to the 'DSAShadowBind' operation previously issued to the supplier SDF causes a transition to the state "Wait for RequestShadow Result " if the result of the 'DSAShadowBind' operation is positive. Otherwise the reception of an error ((E4)SDF_Bind_Error) moves back the SDSM-ShCCi to the state "Idle".

3.3.2.1.3.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 clause 12 and the TCAP services that are used for reporting operating errors are described in subclause 2.8.

3.3.2.2 in-DSAShadowUnbind procedure

3.3.2.2.1 General Description

The X.500 "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525.

3.3.2.2.1.1 Parameters

None.

3.3.2.2.2 Supplier entity (SDF)

3.3.2.2.2.1 Normal Procedure

3.3.2.2.2.1.1 Supplier-initiated DSAShadowUnbind

SDF Preconditions:

(1) SDSM-ShSSi: "SDF Bound"
(2) SDSM-ShSSi:”Bound with CoordinateShadow Sent”
(3) SDSM-ShSSi: "Wait for Coordination Result"
(4) SDSM-ShSSi: "Wait for Update"
(5) SDSM-ShSSi: "Wait for Update Confirmation"

SDF Postconditions:
(1) SDSM-ShSSi: "Idle"

The SDSM-ShSSi has previously initiated a successful 'DSAShadowBind' operation to the consumer SDF. It is in either of the states "SDF Bound", “Bound with CoordinateShadow Sent”, "Wait for Coordination Result", "Wait for Update", or "Wait for Update Confirmation". It determines that the "authenticated association" established between two SDFs is to be terminated (e.g., during a user’s release procedure) and issues a “DSAShadowUnbind” operation ((e12),(e15),(e17),(e20) or (e22) SDF_Unbind) that causes the SDSM-ShSSi to transit back to the state “Idle”.

3.3.2.2.1.2 Consumer-initiated DSAShadowUnbind

SDF Preconditions:
(1) SDSM-ShSCi: "SDF Bound"
(2) SDSM-ShSCi: "Wait for RequestShadow Result"
(3) SDSM-ShSCi: "Wait for Update"
(4) SDSM-ShSCi: "Wait for Update Confirmation"

SDF Postconditions:
(1) SDSM-ShSCi: "Idle"

A 'DSAShadowBind' operation has previously been issued and the SDSM-ShSCi is in either of the states "SDF Bound", "Wait for RequestShadow Result", "Wait for Update", or "Wait for Update Confirmation" waiting for a request/response from the consumer SDF or performing an operation. The reception of the 'DSAShadowUnbind' operation causes a transition to the state "Idle" with the transition SDF_Unbind ((E6), (E8), (E12), or (E15)).

3.3.2.2.2.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 clause 12 and the TCAP services that are used for reporting operating errors are described in subclause 2.8.

3.3.2.2.3 Consumer entity (SDF)

3.3.2.2.3.1 Normal Procedure

3.3.2.2.3.1.1 Supplier-initiated DSAShadowUnbind

SDF Preconditions:
(1) SDSM-ShCSi: "SDF Bound"
(2) SDSM-ShCSi: "Wait for Coordination Result"

(3) SDSM-ShCSi: "Wait for Update"
(4) SDSM-ShCSi: "Wait for Update Confirmation"

SDF Postconditions:
(1) SDSM-ShCSi: "Idle"

A 'DSAShadowBind' operation has previously been issued and the SDSM-ShCSi is in either of the states "SDF Bound", "Wait for Coordination Result", "Wait for Update", or "Wait for Update Confirmation" waiting for a request from the supplier SDF or performing an operation. The reception of the 'DSAShadowUnbind' operation causes a transition to the state "Idle" with the transition SDF_Unbind ((E6), (E9), (E12) or(E14)).

3.3.2.2.3.1.2 Consumer-initiated DSAShadowUnbind

SDF Preconditions:
(1) SDSM-ShCCi: "SDF Bound"
(2) SDSM-ShCCi: "Wait for RequestShadow Result"
(3) SDSM-ShCCi: "Wait for Update"
(4) SDSM-ShCCi: "Wait for Update Confirmation"

SDF Postconditions:
(1) SDSM-ShCCi: "Idle"

The SDSM-ShCCi has previously initiated a successful 'DSAShadowBind' operation to the supplier SDF. It is in either of the states "SDF Bound", "Wait for RequestShadow Result", "Wait for Update", or "Wait for Update Confirmation". It determines that the "authenticated association" established between two SDFs is to be terminated (e.g., during a user’s release procedure) and issues a 'DSAShadowUnbind' operation ((e8), (e10), (e13) or(e15) SDF_Unbind) that causes the SDSM-ShCCi to transit back to the state “Idle”.

3.3.2.3 CoordinateShadowUpdate procedure

3.3.2.3.1 General Description

The X.500 "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525.

3.3.2.3.1.1 Parameters

For the coordinateShadowUpdate operation, see ITU-T Recommendation X.525, subclause 11.1.

3.3.2.3.2 Supplier entity (SDF)

3.3.2.3.2.1 Normal Procedure

3.3.2.3.2.1.1 CoordinateShadowUpdate sent by itself.

SDF Preconditions:
(1) SDSM-ShSSi: "SDF Bound"

SDF Postconditions:

(1) SDSM-ShSSi: "Wait for Update" in case of success
(2) SDSM-ShSSi: "SDF Bound" in case of failure

When the SDSM-ShSSi is in the state "SDF Bound" and a need of coordinating the shadow exists, an internal event occurs. This event, called (e16) Shadow_Coordinate_to_Consumer, causes a transition to the state "Wait for Coordination Result" and the operation is sent to consumer SDF. The SDSM-ShSSi waits for the response from the consumer. The reception of the response ((E18) Shadow_Coordinate_Confirmed) to the “coordinateShadowUpdate” operation previously issued to the consumer SDF causes a transition to the state "Wait for Update" if the result of the “coordinateShadowUpdate” operation is positive. Otherwise the reception of an error ((E19) Coordinate_Failure) moves back the SDSM-ShSSi to the state "SDF Bound".

3.3.2.3.2.1.2 CordinateShadowUpdate sent with DSAShadowBind

SDF Preconditions:

(1) SDSM-ShSSi: "Wait for Subsequent Requests".

SDF Postconditions:

(1) SDSM-ShSSi: "Wait for Update" in case of success.
(2) SDSM-ShSSi: "SDF Bound" in case of failure.

When the SDSM-ShSSi is in the state " Wait for Subsequent Requests " and a need of coordinating the shadow exists, an internal event occurs. This event, called (e2) Shadow_Coordinate_to_Consumer, causes a transition to the state "Bind with Coordinate Shadow". The reception of the delimiter causes a transition to the state "Bind with CoordinateShadow Only" through the internal event (e5) Send_Bind_with_CoordShadow and the operations are sent to the consumer SDF. The SDSM-ShSSi then waits for the response from the consumer. The reception of the response (E7) SDF_Bind_Success to the previously issued "DSAShadowBind" causes a transition to the state "Wait for Coordination Result". The SDSM-ShSSi waits for the response from the consumer to the "coordinateShadowUpdate" operation previously issued to the consumer SDF. If the result of the "coordinateShadowUpdate" operation is positive, the event ((E18)

Shadow_Coordinate_Confirmed) causes a transition to the state "Wait for Update". Otherwise the reception of an error ((E19) Coordinate_Failure) moves back the SDSM-ShSSi to the state "SDF Bound".

3.3.2.3.2.1.3 CoordinateShadowUpdate sent with DSAShadowBind and UpdateShadow

SDF Preconditions:

(1) SDSM-ShSSi: " Wait for Subsequent Requests".

SDF Postconditions:

(1) SDSM-ShSSi: "Wait for Update Confirmation" in case of success.
(2) SDSM-ShSSi: "SDF Bound" in case of failure.

When the SDSM-ShSSi is in the state " Wait for Subsequent Requests " and a need of coordinating the shadow exists, an internal event occurs. This event, called (e2) Shadow_Coordinate_to_Consumer, causes a transition to the state "Bind with Coordinate Shadow". The subsequent reception of an "UpdateShadow" operation through the internal event (e4) Update_to_Consumer causes a transition to the state "Bind with CoordinateShadow and Update" and the operations are sent to the consumer SDF. The SDSM-ShSSi then waits for the response from the consumer. The reception of the response (E9) SDF_Bind_Success to the previously issued “DSAShadowBind” causes a transition to the state "Bound with Coordinate Shadow Sent". The SDSM-ShSSi waits for the response from the consumer to the "coordinateShadowUpdate" operation previously issued to the consumer SDF. If the result of the "coordinateShadowUpdate" operation is positive the event ((E10) Shadow_Coordinate_Confirmed) causes a transition to the state "Wait for Update Confirmation". Otherwise the reception of an error ((E11) Coordinate_Failure) moves back the SDSM-ShSSi to the state "SDF Bound".

3.3.2.3.2.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 clause 12 and the TCAP services that are used for reporting operating errors are described in subclause 2.8.

3.3.2. 3.3 Consumer entity (SDF)

3.3.2.3.3.1 Normal Procedure

3.3.2.3.3.1.1 CoordinateShadowUpdate received by itself

SDF Preconditions:
(1) SDSM-ShCSi: "SDF Bound"

SDF Postconditions:
(1) SDSM-ShCSi: "Wait for Update" in case of success
(2) SDSM-ShCSi: "SDF Bound" in case of failure

The SDF is initially in the state "SDF Bound". After accepting the external event (E7) Shadow_Coordinate_from_Supplier caused by the reception of a “coordinateShadowUpdate” operation from the supplier SDF, a transition to the state "Wait for Coordination Result" occurs. The SDF performs the “coordinateShadowUpdate” operation according to the contents of the “coordinateShadowUpdate” argument. Once the SDF has completed the “coordinateShadowUpdate” operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "SDF Bound" if the coordinateShadowUpdate fails or to the state "Wait for Update" if the coordinateShadowUpdate is successful.

3.3.2.4 RequestShadowUpdate procedure

3.3.2.4.1 General Description

The X.500 "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525.

3.3.2.4.1.1 Parameters

For the requestShadowUpdate operation, see ITU-T Recommendation X.525, subclause 11.2.

3.3.2.4.2 Supplier entity (SDF)

3.3.2.4.2.1 Normal Procedure

3.3.2.4.2.1.1 RequestShadowUpdate received by itself

SDF Preconditions:
(1) SDSM-ShSCi: "SDF Bound"

SDF Postconditions:
(1) SDSM-ShSCi: "Wait for Update" in case of success
(2) SDSM-ShSCi: "SDF Bound" in case of failure

The SDF is initially in the state "SDF Bound". After accepting the external event (E7) Request_for_Shadow_from_Consumer caused by the reception of a 'requestShadowUpdate' operation from the consumer SDF, a transition to the state "Wait for RequestShadow Result" occurs. The SDF performs the 'requestShadowUpdate' operation according to the contents of the

'requestShadowUpdate' argument. Once the SDF has completed the 'requestShadowUpdate' operation, the result or error indication is returned to the consumer SDF. The SDF returns to the state "SDF Bound" if the 'requestShadowUpdate' fails or to the state "Wait for Update" if the 'requestShadowUpdate' is successful.

3.3.2.4.2.1.2 DSA ShadowBind sent with CoordinateShadowUpdate

3.3.2.4.2.1.3 RequestShadowUpdate received with DSAShadowBind

SDF Preconditions:

(1) SDSM-ShSCi: "Wait for Bind Result".

SDF Postconditions:

(1) SDSM-ShSCi: "Wait for Update" in case of success.
(2) SDSM-ShSCi: "SDF Bound" in case of failure.

The SDF is initially in the state "Wait for Bind Result" waiting for other operations to be received than the "DSAShadowBind" operation. When receiving the "RequestShadowUpdate" operation, a transition to the same state occurs through the external event (E3) Request_from_Consumer. The SDF performs the "DSAShadowBind" operation and a transition to the state "SDF Bound" occurs through the internal event (e5) SDF_Bind_Success. Since the "RequestShadowUpdate" operation has already been received, a transition to the state "Wait for RequestShadow Result" occurs through the external event (E7) Request_for_Shadow_from_Consumer. Then, the SDF performs the "RequestShadowUpdate" operation according to the contents of the "RequestShadowUpdate" argument. Once the SDF has completed the "RequestShadowUpdate" operation, the result or error indication is returned to the consumer SDF. The SDF returns to the state "SDF Bound" if the "RequestShadowUpdate" fails or to the state "Wait for Update" if the "RequestShadowUpdate" is successful.

3.3.2.4.3 Consumer entity (SDF)

3.3.2.4.3.1 Normal Procedure

3.3.2.4.3.1.1 RequestShadowUpdate sent by itself.

SDF Preconditions:

(1) SDSM-ShCCi: "SDF Bound".

SDF Postconditions:

(1) SDSM-ShCCi: "Wait for Update" in case of success.
(2) SDSM-ShCCi: "SDF Bound" in case of failure.

When the SDSM-ShCCi is in the state "SDF Bound" and a need of requesting the shadow to be updated exists, an internal event occurs. This event, called (e9) Shadow_Request_to_Supplier, causes

a transition to the state "Wait for RequestShadow Result" and the operation is sent to the supplier SDF. The SDSM-ShCCi waits for the response from the supplier. The reception of the response ((E12) Request_Shadow_Result) to the "requestShadowUpdate" operation previously issued to the supplier SDF causes a transition to the state "Wait for Update" if the result of the "requestShadowUpdate" operation is positive. Otherwise the reception of an error ((E11) Failed_Shadow_Request) moves back the SDSM-ShCCi to the state "SDF Bound".

3.3.2.4.3.1.2 RequestShadowUpdate sent with DSAShadowBind

SDF Preconditions:

(1) SDSM-ShCCi: "Wait for Subsequent Requests".

SDF Postconditions:

(1) SDSM-ShCCi: "Wait for Update" in case of success.
(2) SDSM-ShCCi: "SDF Bound" in case of failure.

When the SDSM-ShCCi is in the state " Wait for Subsequent Requests " and a need of requesting the shadow to be updated exists, an internal event occurs. This event, called (e2) Request_to_Supplier, causes a transition to the state "Bind with RequestShadow" and the operations are sent to the supplier SDF. The SDSM-ShCCi waits for the response from the supplier. When the DSAShadowBind is successful, the event (E5) SDF_Bind_Success causes a transition to the state "Wait for RequestShadow Result". The SDSM-ShCCi then waits for the response from the supplier. The reception of the response ((E12) Request_Shadow_Result) to the "requestShadowUpdate" operation previously issued to the supplier SDF causes a transition to the state "Wait for Update" if the result of the "requestShadowUpdate" operation is positive. Otherwise the reception of an error ((E11) Failed_Shadow_Request) moves the SDSM-ShCCi to the state "SDF Bound".

3.3.2.4.3.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 clause 12 and the TCAP services that are used for reporting operating errors are described in subclause 2.8.

3.3.2.5 UpdateShadow procedure

3.3.2.5.1 General Description

The X.500 "shadowing" operations allow information to be copied between two SDFs. The shadowing operations are also used to maintain this copied information. For each shadowing agreement between a pair of SDFs, one SDF is designated as the supplier of copied information and the other SDF is the consumer.

The DSAShadowBind and DSAShadowUnbind operations are used by cooperating SDFs at the beginning and end of a particular period of providing copies. The coordinateShadowUpdate is used by a shadow supplier to indicate the shadowing agreement for which it intends to send updates. The requestShadowUpdate operation is used by the shadow consumer to request updates from the shadow

supplier. The updateShadow operation is invoked by the shadow supplier to send copied data to the shadow consumer. This operation must be preceded first by either a coordinateShadowUpdate or a requestShadowUpdate operation. For a full description of the "shadowing" operations, see ITU-T Recommendation X.525.

3.3.2.5.1.1 Parameters

UpdateShadow operation, see ITU-T Recommendation X.525, subclause 11.3.

3.3.2.5.2 Supplier entity (SDF)

3.3.2.5.2.1 Normal Procedure

3.3.2.5.2.1.1 Supplier-initiated updateShadow

3.3.2.5.2.1.1.1 updateShadow sent by itself

SDF Preconditions:
(1) SDSM-ShSSi: "Wait for Update”

SDF Postconditions:
(1) SDSM-ShSSi: "SDF Bound"

When the SDSM-ShSSi is in the state "Wait for Update" and a need of updating the shadow exists, an internal event occurs. This event, called (e21) Shadow_Update_to_Consumer, causes a transition to the state "Wait for Update Confirmation" and the operation is sent to the consumer SDF. The SDSM-ShSSi waits for the response from the consumer. The reception of the response ((E23) Shadow_Update_Confirmed) to the 'updateShadow' operation previously issued to the consumer SDF causes a transition to the state "SDF Bound". The response from the consumer SDF may be either the result of the 'updateShadow' operation or an error.

3.3.2.5.2.1.1.2 updateShadow sent with DSAShadowBind and CoordinateShadowUpdate

SDF Preconditions:

(1) SDSM-ShSSi: "Bind with CoordinateShadow".

SDF Postconditions:

(1) SDSM-ShSSi: "SDF Bound".

When the SDSM-ShSSi is in the state " Bind with CoordinateShadow " and a need of updating the shadow exists, an internal event occurs. This event, called (e4) Update_to_Consumer, causes a transition to the state "Bind with CoordinateShadow and Update" and the operation is sent to the consumer SDF with a DSAShadowBind and CoordinateShadowUpdate operation. The SDSM-ShSSi then waits for the response from the consumer. The reception of the response (E9) SDF_Bind_Success to the previously issued DSAShadowBind causes a transition to the state “Bound with Coordinate Shadow Sent". The SDSM-ShSSi waits for the response from the consumer to the

"coordinateShadowUpdate" operation previously issued to the consumer SDF. The reception of the event ((E10) Shadow_Coordinate_Confirmed) causes a transition to the state "Wait for Update Confirmation". The reception of the response ((E23) Shadow_Update_Confirmed) to the "updateShadow" operation previously issued to the consumer SDF causes a transition to the state "SDF Bound". The response from the consumer SDF may be either the result of the "updateShadow" operation or an error.

3.3.2.5.2.1.2 Consumer-initiated updateShadow

3.3.2.5.2.1.2.1 updateShadow sent by itself

SDF Preconditions:

(1) SDSM-ShSCi: "Wait for Update".

SDF Postconditions:

(1) SDSM-ShSCi: "SDF Bound".

When the SDSM-ShSCi is in the state "Wait for Update" and a need of updating the shadow exists, an internal event occurs. This event, called (e13) Shadow_Update_to_Consumer, causes a transition to the state "Wait for Update Confirmation" and the operation is sent to the consumer SDF. The SDSM-ShSCi waits for the response from the consumer. The reception of the response ((E14) Shadow_Update_Confirmed) to the "updateShadow" operation previously issued to the consumer SDF causes a transition to the state "SDF Bound". The response from the consumer SDF may be either the result of the "updateShadow" operation or an error.

3.3.2.5.2.2 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 subclause 12 and the TCAP services that are used for reporting operating errors are described in subclause 2.8.

3.3.2.5.3 Consumer entity (SDF)

3.3.2.5.3.1 Normal Procedure

3.3.2.5.3.1.1 Supplier-initiated UpdateShadow

3.3.2.5.3.1.1.1 UpdateShadow received by itself

SDF Preconditions:
(1) SDSM-ShCSi: "Wait for Update"

SDF Postconditions:
(1) SDSM-ShCSi: "SDF Bound"

The SDF is initially in the state "Wait for Update". After accepting the external event (E13)

Shadow_Update_from_Supplier caused by the reception of a 'updateShadow' operation from the supplier SDF, a transition to the state "Wait for Update Confirmation" occurs. The SDF performs the 'updateShadow' operation according to the contents of the 'updateShadow' argument. Once the SDF has completed the 'updateShadow' operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "SDF Bound.

3.3.2.5.3.1.1.2 UpdateShadow received with DSAShadowBind and CoordinateShadowUpdate

SDF Preconditions:

(1) SDSM-ShCSi: "Wait for Bind Result".

SDF Postconditions:

(1) SDSM-ShCSi: "SDF Bound".

The SDF is initially in the state "Wait for Bind Result" waiting for other operations to be received than the "DSAShadowBind" operation. When receiving the "UpdateShadow" operation after receiving the "CoordinateShadowUpdate" operation, a transition to the same state occurs through the external event (E3) Request_from_Supplier. The SDF performs the "DSAShadowBind" operation and a transition to the state "SDF Bound" occurs through the internal event (e5) SDF_Bind_Success. Since the "CoordinateShadowUpdate" operation has already been received, a transition to the state "Wait for Coordination Result" occurs through the external event (E7) Shadow_Coordinate_from_Supplier. Then, the SDF performs the "CoordinateShadowUpdate" operation and a transition to the state "Wait for Update" occurs through the internal event (e10) Shadow_Coordinate_Confirmed. Since the "UpdateShadow" operation has also already been received, a transition to the state "Wait for Update Confirmation" occurs through the external event (E13) Shadow_Update_from_Supplier. Finally, the SDF performs the "UpdateShadow" operation according to the contents of the "updateShadow" argument. Once the SDF has completed the "updateShadow" operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "SDF Bound".

3.3.2.5.3.1.2 Consumer-initiated updateShadow

SDF Preconditions:

(1) SDSM-ShCCi: "Wait for Update".

SDF Postconditions:

(1) SDSM-ShCCi: "SDF Bound".

The SDF is initially in the state "Wait for Update". After accepting the external event (E14) Shadow_Update_from_Supplier caused by the reception of a "updateShadow" operation from the supplier SDF, a transition to the state "Wait for Update Confirmation" occurs. The SDF performs the "updateShadow" operation according to the contents of the "updateShadow" argument. Once the SDF has completed the "updateShadow" operation, the result or error indication is returned to the supplier SDF. The SDF returns to the state "SDF Bound".

3.3.2.5.4 Error Handling

Generic error handling for the shadowing operations related errors are described in ITU-T Recommendation X.525 clause 12 and the TCAP services that are used for reporting operating errors are described in subclause 2.8.

Annex A

Execute operation

A.1. Execute Operation

The execute operation performs a sequence of execution steps, according to a pre-defined method, using input information and returns result information. Each step is either a DAP operation (that could be an execute operation), the execution of an algorithm or a decision test.
The parameters of the individual DAP operations are taken from the input parameters and the results of previous operations and/or the output of the algorithms associated with the method. The output parameters are taken from the results of the individual operations. The execute operation is considered to be an atomic operation.

```
execute OPERATION ::= {
        ARGUMENT        ExecuteArgument
        RESULT          ExecuteResult
        ERRORS          { attributeError | nameError |
                          serviceError | referral |
                          securityError |
                          updateError | executionError }
        CODE            id-opcode-execute }
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.&id({SupportedMethods}{@method-id}) OPTIONAL,
                                valuesWithContext [0] SET OF SEQUENCE {
                                                value [0]
METHOD.&InputAttributes.&id({SupportedMethods}{@method-id})
OPTIONAL,
                                                contextList [1] SET OF Context
                                                } OPTIONAL
                                } OPTIONAL,
                specific-input  [3]
METHOD.&SpecificInput({SupportedMethods}{@method-id}) OPTIONAL,
                COMPONENTS OF   CommonArguments },
        DIRQOP.&dapModifyEntryArg-QOP{@qop} }
```

The **object** field identifies the entry in the DIT from/on which the method is to be executed.
The **execute-id** field identifies the method which is to be executed within the SDF.
The **input-assertions** field provides a set of attribute values which are used as an input to the method execution.
The **specific-input** field identifies the additional information which is required by the SDF in order to perform the method.

```
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.

```
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.2 Execution Error

The executionError is returned by an Execute operation in the case ofthe operation not completing.

```
executionError ERROR ::= {
    PARAMETER      OPTIONALLY-PROTECTED {
                SET {
                    problem       [0]   ExecutionProblem,
                    COMPONENTS OF CommonResults },
                DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE              id-errcode-executionError }

ExecutionProblem  ::=  INTEGER {
    missingInputValues (1),
    executionFailure(2) }
```

The executeProblem identifies the cause of the execute operation failure:

- missingInputValues is returned in the input-values field contains the wrong input information for the method being executed.

- executionFailure is returned when the method fails to complete correctly. This is caused by the failure of one of the DAP operations contained within the method.

A.3 Code Value

-- object identifier assignment
-- error codes
id-errcode-executionError Code ::= local:10
-- operation codes
id-opcode-execute Code ::=local:10

Annex B

Glossary and Abbreviation

This annex as is specified in ITU-T Recommendation Q.1290 is applied.