



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.1248.5

(07/2001)

SERIES Q: SWITCHING AND SIGNALLING
Intelligent Network

Interface Recommendation for Intelligent Network Capability Set 4: SDF-SDF interface

ITU-T Recommendation Q.1248.5

ITU-T Q-SERIES RECOMMENDATIONS
SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.799
Q3 INTERFACE	Q.800–Q.849
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1699
SIGNALLING REQUIREMENTS AND PROTOCOLS FOR IMT-2000	Q.1700–Q.1799
SPECIFICATIONS OF SIGNALLING RELATED TO BEARER INDEPENDENT CALL CONTROL (BICC)	Q.1900–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

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

ITU-T Recommendation Q.1248.5

Interface Recommendation for Intelligent Network Capability Set 4: SDF-SDF interface

Summary

The Q.1248.x family of ITU-T Recommendations defines the Intelligent Network (IN) Application Protocol (INAP) for IN Capability Set 4 (IN CS-4). This definition is based upon IN CS-3 ITU-T Q.1238 and ITU-T Q.1234 (2000) and upon the general rules for INAP provided in ITU-T Q.1208, and is consistent with the scope of IN CS-4 as defined in ITU-T Q.1241.

Within the Q.124x series of ITU-T Recommendations, the Q.1248.x family of ITU-T Recommendation describes the protocol realizing the Q.1241 Distributed Functional Plane in a service and vendor implementation independent manner, as constrained by the capabilities of the embedded base of network technology. This provides the flexibility to allocate distributed functionality into multiple physical network configurations and to evolve IN from IN CS-4 to some future IN CS-N.

This Recommendation belongs to the Q.1248.x family of ITU-T Recommendations for IN CS-4. It covers the SDF-SDF interface including the description of the aspects of the SDF and SDF Functional Entities which are relevant to this interface.

This Recommendation includes an electronic attachment containing Annex A, ASN.1 definitions.

Source

ITU-T Recommendation Q.1248.5 was prepared by ITU-T Study Group 11 (2001-2004) and approved under the WTSC Resolution 1 procedure on 13 July 2001.

Keywords

Intelligent Network (IN), Intelligent Network Application Protocol (INAP), IN Capability Set 4 (IN CS-4), Service Data Function (SDF), SDF-SDF Interface.

FOREWORD

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

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

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

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

NOTE

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

INTELLECTUAL PROPERTY RIGHTS

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

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

© ITU 2002

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from ITU.

CONTENTS

Page

1	Scope.....	1
2	References.....	1
3	Abbreviations and acronyms	1
4	Introduction.....	1
4.1	SDF-SDF relationship.....	1
4.2	Introduction to the IN X.500 DSP and DISP subset.....	2
4.3	Versions and the rules for extensibility	3
4.3.1	Version negotiation.....	3
4.3.2	Initiating SDF side.....	3
4.3.3	Request processing at the responding SDF side	4
5	FSM for SDF	4
5.1	Overview.....	4
5.2	SDME-FSM	6
5.2.1	State transition model for TFC Initiator-related states	6
5.2.2	State transition model for TFC Responder-related states.....	7
5.3	SDF state transition model for SDF-related states.....	9
5.3.1	SDF state transition models for shadowing.....	9
5.3.2	SDF state transition models for chaining	21
6	Detailed operation procedure.....	25
6.1	ChainedExecute procedure	25
6.1.1	General description.....	25
6.1.2	Parameters	25
6.1.3	Invoking entity (SDF).....	26
6.1.4	Responding entity (SDF)	26
6.2	IN-ChainedAddEntry procedure.....	27
6.2.1	General description.....	27
6.2.2	Parameters	27
6.2.3	Invoking entity (SDF).....	27
6.2.4	Responding entity (SDF)	28
6.3	IN-ChainedModifyEntry procedure.....	28
6.3.1	General description.....	28
6.3.2	Parameters	28
6.3.3	Invoking entity (SDF).....	28
6.3.4	Responding entity (SDF)	28
6.4	IN-ChainedRemoveEntry procedure	29
6.4.1	General description.....	29



Documents

Document Sharing Hub

ITU-T Rec. Q.1248.5 (07/2001)

iii

	Page
6.4.2 Parameters	29
6.4.3 Invoking entity (SDF).....	29
6.4.4 Responding entity (SDF)	29
6.5 IN-ChainedSearch procedure.....	29
6.5.1 General description.....	29
6.5.2 Parameters	30
6.5.3 Invoking entity (SDF).....	30
6.5.4 Responding entity (SDF)	30
6.6 IN-CoordinateShadowUpdate procedure.....	30
6.6.1 General description.....	30
6.6.2 Parameters	30
6.6.3 Supplier entity (SDF).....	31
6.6.4 Consumer entity (SDF).....	32
6.7 IN-DSABind procedure	33
6.7.1 General description.....	33
6.7.2 Parameters	33
6.7.3 Invoking Entity (SDF)	33
6.7.4 Responding entity (SDF)	34
6.8 IN-DSAShadowBind procedure	34
6.8.1 General Description	34
6.8.2 Parameters	35
6.8.3 Supplier entity (SDF).....	35
6.8.4 Consumer entity (SDF).....	37
6.9 IN-RequestShadowUpdate procedure.....	38
6.9.1 General description.....	38
6.9.2 Parameters	38
6.9.3 Supplier entity (SDF).....	39
6.9.4 Consumer entity (SDF).....	39
6.10 IN-Unbind procedure	40
6.10.1 General description.....	40
6.10.2 Parameters	40
6.10.3 Inviting entity (SDF in shadowing).....	40
6.10.4 Responding entity (SDF in shadowing).....	41
6.10.5 Invoking entity (SDF in directory access).....	42
6.10.6 Responding entity (SDF in directory access)	42
6.10.7 Invoking entity (Initiating SDF in traffic flow control)	43
6.10.8 Responding entity (Responding SDF in traffic flow control)	43
6.11 IN-UpdateShadow procedure	43
6.11.1 General description.....	43

	Page
6.11.2 Parameters	44
6.11.3 Supplier entity (SDF).....	44
6.11.4 Consumer entity (SDF).....	45
6.12 TFCBind procedure	46
6.12.1 General description.....	46
6.12.2 Invoking entity (initiating SDF)	46
6.12.3 Responding entity (responding SDF)	47
6.13 TrafficFlowControl procedure	47
6.13.1 General description.....	47
6.13.2 Parameters	48
6.13.3 Invoking entity (Initiating SDF)	48
6.13.4 Responding entity (Responding SDF).....	48
6.14 Operation timer values.....	49
7 Parameters.....	49
7.1 AgreementID	49
7.2 Argument	49
7.3 ControlType	49
7.4 Duration	50
7.5 IN-ChainedArgument	50
7.6 IN-ChainedResult	52
7.7 Information	52
7.8 LastUpdate	52
7.9 RequestedStrategy	52
7.10 Result	53
7.11 Security	53
7.12 SecurityParameters	53
7.13 TfcCriteria.....	53
7.14 UpdatedInfo	53
7.15 UpdateStrategy.....	54
7.16 UpdateTime	54
7.17 UpdateWindow	54
7.18 WasChained	54
8 Errors	54
8.1 Operation related error procedures	54
8.1.1 AttributeError	55
8.1.2 ExecutionError.....	56
8.1.3 IN-DirectoryBindError	56



	Page
8.1.4 IN-DSAReferral.....	57
8.1.5 NameError	57
8.1.6 SecurityError	58
8.1.7 ServiceError.....	59
8.1.8 ShadowError.....	59
8.1.9 TFCBindError.....	60
8.1.10 UpdateError	60
9 Service assumed from TCAP.....	61
9.1 Normal procedures.....	61
9.2 Abnormal procedures.....	61
9.3 Dialogue handling.....	61
9.3.1 Dialogue establishment.....	61
9.3.2 Dialogue continuation.....	61
9.3.3 Dialogue termination	61
9.3.4 User Abort	61
9.3.5 Provider Abort	61
9.3.6 Mapping to TC Dialogue Primitives	62
9.4 Component Handling.....	63
9.4.1 Procedures for INAP Operations.....	63
9.4.2 Mapping to TC Component parameters	63
Annex A – ASN.1 Definitions (electronic attachment)	63

ITU-T Recommendation Q.1248.5

Interface Recommendation for Intelligent Network Capability Set 4: SDF-SDF interface¹

1 Scope

This Recommendation belongs to the Q.1248.x family of ITU-T Recommendations for IN Capability Set 4. It specifies the protocol on the SDF-SDF interface and provides a description of the aspects of the SDF and SDF Functional Entities which are involved in the realization of this interface.

2 References

All ITU-T Recommendations and other references are identified in ITU-T Q.1248.1.

3 Abbreviations and acronyms

All abbreviations and acronyms used in this text are defined in ITU-T Q.1248.1.

4 Introduction

4.1 SDF-SDF relationship

This SDF-SDF relationship is used for messages between two SDFs in the public network. The relationship supports both access to service data, which can be for intra-network or inter-network purposes.

The SDF-SDF relationship supports messages for services when no call is actually taking place (call unrelated) as well as during call processing. For the most part, the call unrelated actions are to support the registration, authentication, encryption and handover procedures for terminal and personal mobility.

The SDF-SDF relationship is also one of a limited set of relationships to support inter-networking. As such it provides a point of interconnection to the network, effectively hiding the specific structure of the network and providing access security to the network from other public networks.

The SDF-SDF relationship is used for various purposes. Each of the purposes can apply to the intra-network, or inter-network cases. Specifically, the relationship is used for the following:

- a) **Secured Data Acquisition** between SDFs. In this case a SDF requires data which is not stored within the SDF to complete a request for service data by another SDF. A crucial aspect of this relationship is security necessary to support network isolation and inter networking.
- b) **Secured Copying of Service Data** between SDFs. In this case a SDF copies a portion of service data which is controlled by another SDF. A crucial aspect of this relationship is the keeping of the original security and access information relation to the service data once the data has been copied and ensuring that copies of data are deleted when no longer required.

¹ This Recommendation includes an electronic attachment containing Annex A, ASN.1 definitions.

4.2 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 fulfil the IN CS-4 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 1997 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 behaviour 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.

Several assumptions were used to design the DSP and DISP for IN CS-4. 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 online 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 deactivated. 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.

4.3 Versions and the rules for extensibility

The Directory may be distributed and more than two Directory Application Entities may interoperate to service a request. The Directory AEs may be implemented conforming to different editions of the Directory specification of the Directory service which may or may not be represented by different protocol version numbers. The version number is negotiated to the highest common version number between two directly binding Directory AEs.

4.3.1 Version negotiation

When accepting an association, i.e. binding, utilizing the DSP or DISP, the version negotiated shall only affect the point-to-point aspects of the protocol exchanged between the initiating SDF and the responding SDF to which it is connected. Subsequent requests or responses on the dialogue shall be constrained by the version negotiated.

4.3.2 Initiating SDF side

4.3.2.1 Request and response processing at the initiating SDF side

The initiating SDF may initiate requests using the highest edition of the specification of that request it supports. If one or more elements of the request are critical, it shall indicate the extension number(s) in the critical Extensions parameter.

NOTE 1 – If the information the extension replaced in a CHOICE, ENUMERATED or INTEGER (used as ENUMERATED) type would be essential for proper operation in a responding SDF implemented according to an earlier edition of the specification, it is recommended that the extension be marked critical.

When processing a response, an initiating SDF shall:

- a) ignore all unknown bit name assignments within a bit string; and
- b) ignore all unknown named numbers in an ENUMERATED type or INTEGER type that is being used in the enumerated style, provided the number occurs as an optional element of a SET or SEQUENCE; and
- c) ignore all unknown elements in SETs, at the end of SEQUENCEs, or in CHOICEs where the CHOICE is itself an optional element of a SET or SEQUENCE.

NOTE 2 – Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown CHOICEs in mandatory elements of SETs and SEQUENCEs can be ignored without invalidating the operation. The identification of such elements is for further study.

- d) not consider the receipt of unknown attribute types and attribute values as a protocol violation; and
- e) optionally report the unknown attribute types and attribute values to the user.

4.3.2.2 Extensibility rules for error handling at the initiating SDF side

When processing a known error type with unknown indicated problems and parameters, an initiating SDF shall:

- a) not consider the receipt of unknown indicated problems and parameters as a protocol violation (i.e. it shall not issue a TC-U-REJECT or abort the dialogue); and
- b) optionally report the additional error information to the user.

When processing an unknown error type, an initiating SDF shall:

- a) not consider the receipt of unknown error type as a protocol violation (i.e. it shall not issue a TC-U-REJECT or abort the application association); and
- b) optionally report the error to the user.

4.3.3 Request processing at the responding SDF side

If any responding SDF performing an operation detects an element **criticalExtensions** whose semantic is unknown, it shall return an **unavailableCriticalExtension** indication as a **serviceError**.

NOTE 1 – If a **criticalExtensions** string with one or more zero values is received, this indicates either that the extensions corresponding to the values are not present or are not critical. The presence of a zero value in a **criticalExtensions** string shall not be inferred as either the presence or absence of the corresponding extension in the APDU.

Otherwise, when processing a request from an initiating SDF, a responding SDF shall:

- a) ignore all unknown bit name assignments within a bit string; and
- b) ignore all unknown named numbers in an ENUMERATED type or INTEGER type that is being used in the enumerated style, provided the number occurs as an optional element of a SET or SEQUENCE; and
- c) ignore all unknown elements in SETs, at the end of SEQUENCEs, or in CHOICES where the CHOICE is itself an optional element of a SET or SEQUENCE.

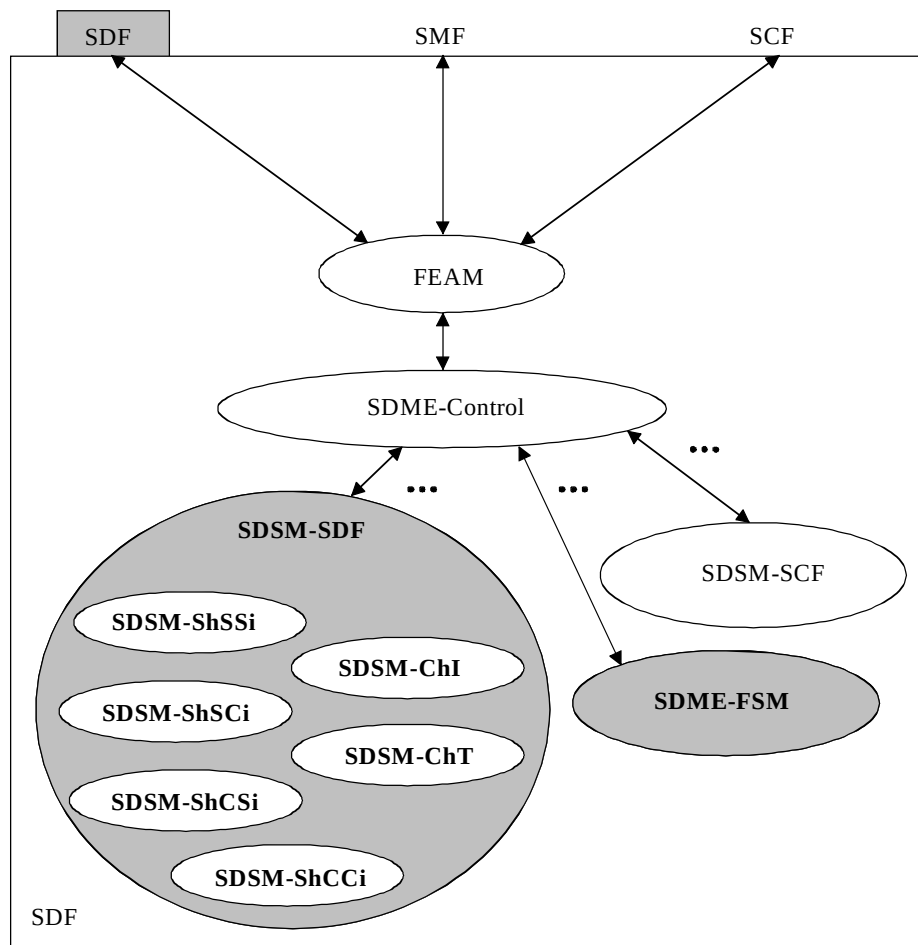
NOTE 2 – Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown CHOICES in mandatory elements of SETs and SEQUENCEs can be ignored without invalidating the operation. The identification of such elements is for further study.

5 FSM for SDF

5.1 Overview

The SDSM-ChI and SDSM-ChT handle the interactions between SDFs for chaining initiation and termination. The SDSM-ShSSi and SDSM-ShCSi handle the interactions between SDFs for shadowing supplier and consumer initiated by supplier respectively. The SDSM-ShSCi and SDSM-ShCCi handle the same as above initiated by consumer. The SDSM-FSM handles the interaction between the SDF and the SDF management functions.

The structure of the SDF FSMs is illustrated in Figure 1. The Service Data Function (SDF) contains and manages data associated with the service logic processing programs (SLPs), and accessed during the execution of the SLP instances (SLPs). The SDF also includes functional entity access management and security management.



T11107370-00

Figure 1/Q.1248.5 – SDF interfaces

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 SCF or cooperating SDF are executed by the SDF Management Entity (SDME). The SDME is comprised of an SDME control, traffic control (TFC), and several instances of SDME FSMs. The SDME control interfaces the different SDF FSMs (e.g. SDSM-SCF) and SDME-FSMs respectively as well as the Functional Entity Access Manager (FEAM).

The FEAM provides the low level interface maintenance functions.

The SDME Control maintains the associations with the SCF and cooperating SDFs on behalf of all instances of the SDF FSMs (e.g. SDSM-SCF, 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.

5.2 SDME-FSM

SDME-FSM has TFC Initiator finite state model and TFC Responder finite state model.

5.2.1 State transition model for TFC Initiator-related states

The TFC Initiator finite state model (FSM) which is located in SDME-FSM is depicted in Figure 2.

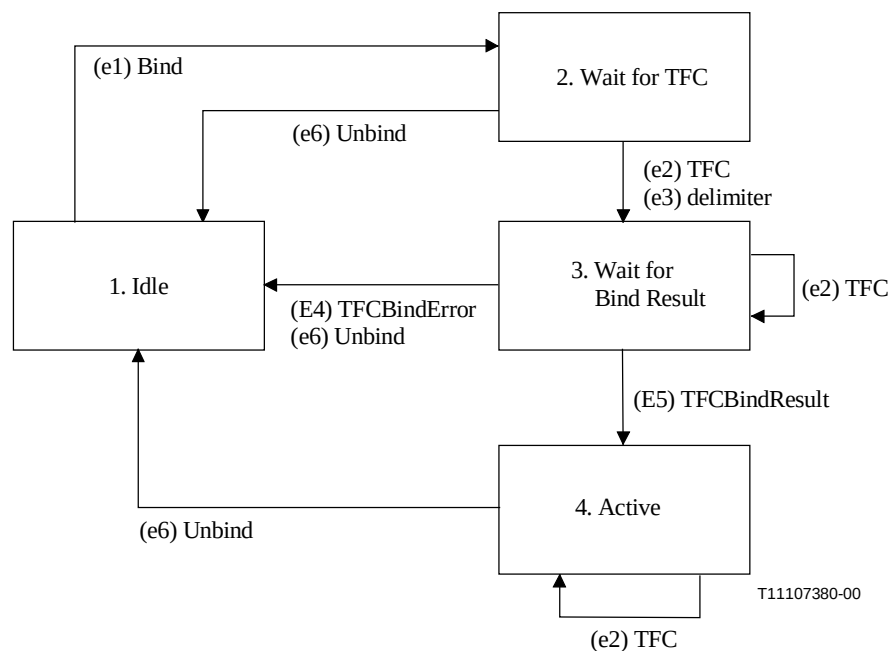


Figure 2/Q.1248.5 – State model for TFC initiator

Each state is discussed in one of the following subclauses.

General rules applicable to more than one state are as follows:

In any state, if the dialogue with the Responder is terminated, then the TFC Initiator FSM returns to **Idle** state after ensuring that any resources allocated to the dialogue have been deallocated.

5.2.1.1 State 1: "Idle"

This state is the initial state for the FSM after it has been created by the FE message management. The only event accepted in this state is:

- (e1) Bind: On receipt of this event the FSM constructs a **TFCBind** operation. This event causes a transition out of this state to State 2, **Wait for TFC**.

5.2.1.2 State 2: "Wait for TFC"

In this state, a Bind request has been received from the Application and a **TFCBind** operation has been constructed, but not sent. Three events are considered in this state:

- (e2) TFC: This event causes a **TrafficFlowControl** operation to be constructed. The **TFCBind** and **TrafficFlowControl** operations are sent to the remote TFC Responder FSM and the event causes a transition out of this state to State 3, **Wait for Bind Result**;
- (e3) delimiter: This event causes the constructed **TFCBind** operation to be sent to the remote TFC Responder FSM and the event causes a transition out of this state to State 3, **Wait for Bind Result**; and
- (e6) Unbind: This event causes the constructed **TFCBind** operation to be discarded and causes a transition out of this state to State 1, **Idle**.

5.2.1.3 State 3: "Wait for Bind Result"

The following events are considered in this state:

- (e2) TFC: This event causes the FSM to save the event to be processed after a transition to another state occurs. The FSM remains in the same state;
- (E4) TFCBindError: This event causes the FSM to notify the Application that the association establishment failed and causes a transition out of this state to State 1, **Idle**;
- (E5) TFCBindResult: This event causes the FSM to notify the Application that the association establishment succeeded and transitions out of this state to State 4, **Active**; and
- (e6) Unbind: This event causes the FSM to construct and send an **IN-Unbind** to the remote TFC Responder FSM and causes a transition out of this state to State 1, **Idle**.

5.2.1.4 State 4: "Active"

The following events are considered in this state:

- (e2) TFC: This event causes a **TrafficFlowControl** operation to be constructed and sent to the remote TFC Responder FSM. The FSM remains in the same state; and
- (e6) Unbind: This event causes the FSM to construct and send an **IN-Unbind** to the remote TFC Responder FSM and causes a transition out of this state to State 1, **Idle**.

5.2.2 State transition model for TFC Responder-related states

The TFC Responder finite state model (FSM) is depicted in Figure 3.

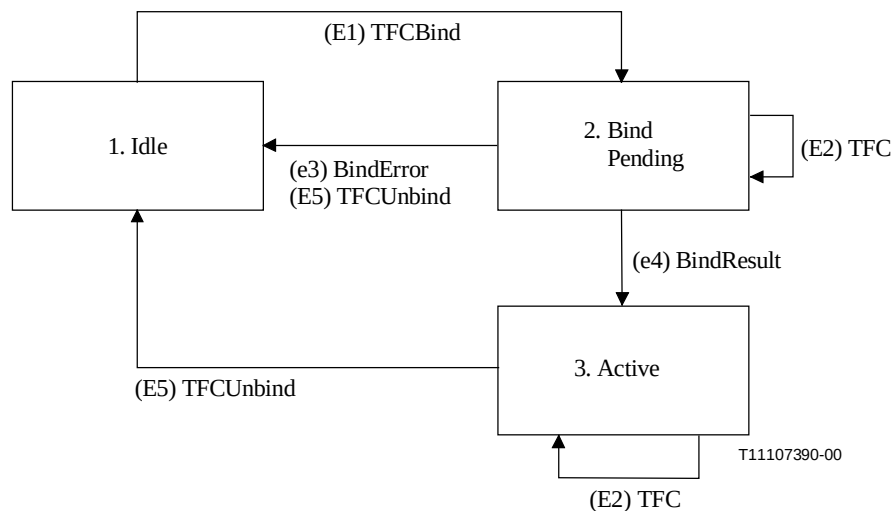


Figure 3/Q.1248.5 – State model for TFC responder

Each state is discussed in one of the following subclauses.

General rules applicable to more than one state are as follows:

In any state, if the dialogue with the Responder is terminated, then the TFC Responder FSM returns to **Idle** state after ensuring that any resources allocated to the dialogue have been deallocated.

5.2.2.1 State 1: "Idle"

This state is the initial state for the FSM after it has been created by the FE message management. The only event accepted in this state is:

- (E1) TFCBind: This event causes the FSM to notify the Application that a TFC association has been requested by a remote TFC Initiator FSM and causes a transition out of this state to State 2, **Bind Pending**;

5.2.2.2 State 2: "Bind Pending"

The following events are considered in this state:

- (E2) TFC: This event causes the FSM to save the event to be processed after a transition to another state occurs. The FSM remains in the same state;
- (e3) BindError: This event causes the FSM to construct a **tfcbindError** error message and sends it to the remote TFC Initiator FSM and causes a transition out of this state to State 1, **Idle**;
- (e4) BindResult: This event causes the FSM to construct a **tfcbindResult** message and sends it to the remote TFC Initiator FSM and causes a transition out of this state to State 3, **Active**;
- (E5) TFCUnbind: This event causes the FSM to notify the Application that the association has been terminated by an **IN-Unbind** operation from the remote TFC Initiator FSM and causes a transition out of this state to State 1, **Idle**.

5.2.2.3 State 3: "Active"

The following events are considered in this state:

- (E2) TFC: This event causes the FSM to notify the Application of the need to process a new traffic flow control request from the remote TFC Initiator FSM. The FSM remains in the same state; and
- (E5) TFCUnbind: This event causes the FSM to notify the Application that the association has been terminated by an **IN-Unbind** operation from the remote TFC Initiator FSM and causes a transition out of this state to State 1, **Idle**.

5.3 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 5.3.1. States related to SDF/SDF interactions for chaining are given in 5.3.2.

5.3.1 SDF state transition models for shadowing

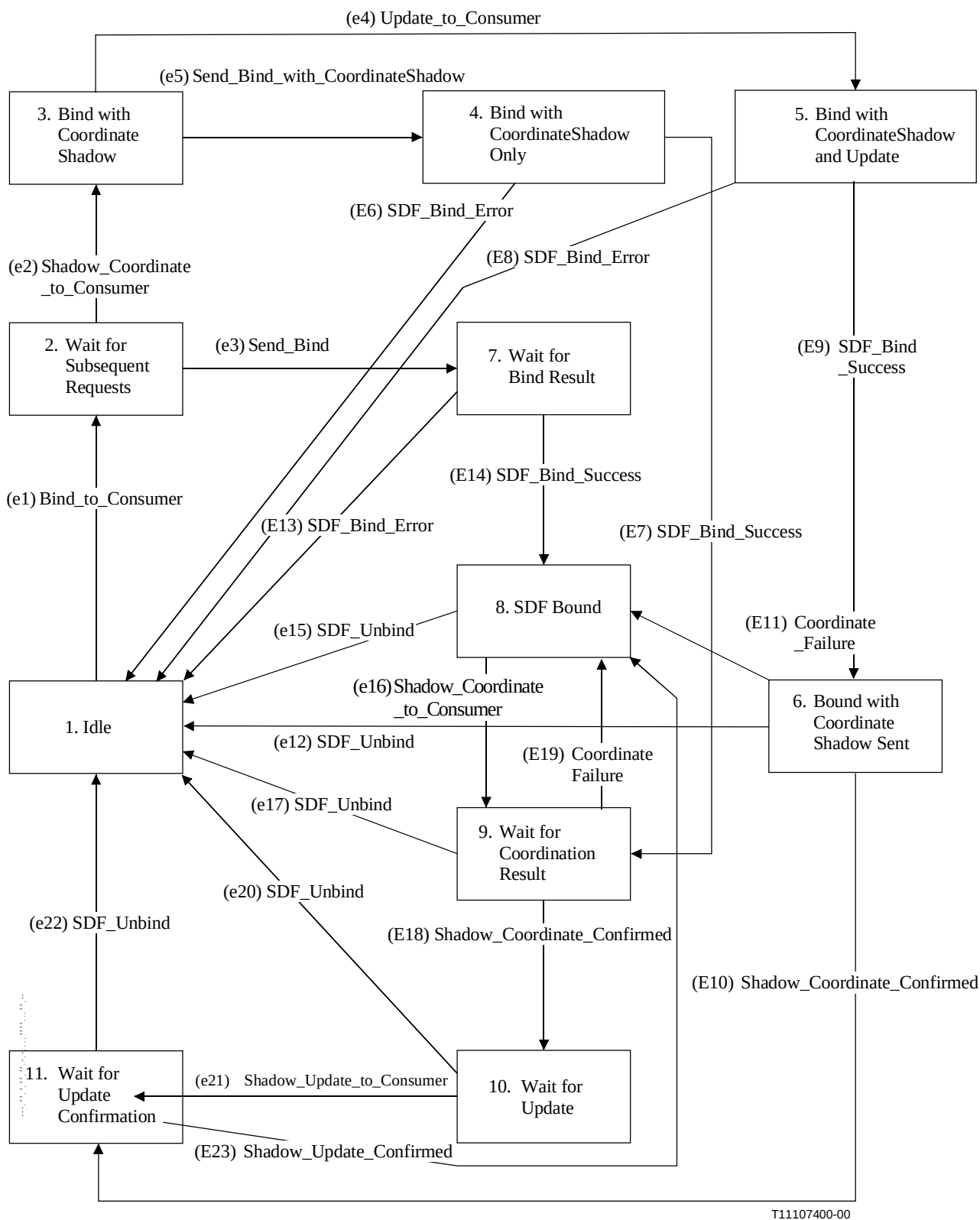
As for the shadowing procedure, an SDF can play the role of a copy supplier and a copy consumer. Moreover, the shadowing procedure can be initiated by a copy consumer as well as a copy supplier. Therefore, there can be four FSMs in total as described below.

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 **IN-DSAShadowBind** operation together with other DISP operations in one TC message is taken into consideration.

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

The SDF Finite State Machine for a shadow supplier in case of supplier-initiated (SDSM-ShSSi) is depicted in Figure 4.



T11107400-00

Figure 4/Q.1248.5 – SDF FSM for a shadow 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 request to execute an **IN-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, an **IN-CoordinateShadowUpdate** operation to be sent with the **IN-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 an **IN-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 or the expiration of a timer. Once the internal event is received, a TCAP message containing the **IN-DSAShadowBind** operation is sent to the consumer SDF. This event causes a transition out of this state to State 7, **Wait for Bind Result**.

State 3: "Bind with Coordinate Shadow"

In this state, an **IN-UpdateShadow** operation to be sent with the **IN-DSAShadowBind** and **IN-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 **IN-UpdateShadow**. This event causes a TCAP message containing the **IN-DSAShadowBind**, **IN-CoordinateShadowUpdate** and **IN-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_CoordinateShadow: 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 **IN-DSAShadowBind** and **IN-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, an **IN-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 **IN-DSAShadowBind** operation previously issued to the consumer SDF. An **IN-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 an **IN-DSAShadowBind** result. This indicates a successful completion of the **IN-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, an **IN-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 **IN-DSAShadowBind** operation previously issued to the consumer SDF. An **IN-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 an **IN-DSAShadowBind** result. This indicates a successful completion of the **IN-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, an **IN-CoordinateShadowUpdate** result is expected from the consumer SDF. Three events are considered in this state:

- (E10) Shadow_Coordinate_Confirmed: This is an external event, caused by the reception of an **IN-CoordinateShadowUpdate** result. This indicates the successful completion of the **IN-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 **IN-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, an **IN-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 **IN-DSAShadowBind** operation previously issued to the consumer SDF. An **IN-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 **IN-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 an **IN-CoordinateShadowUpdate** operation to it. 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) or causing the issuing of the **IN-Unbind** 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 9, **Wait for Coordination Result**.

State 9: "Wait for Coordination Result"

In this state, the supplier SDF has sent an **IN-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 **IN-CoordinateShadowUpdate** operation. This event causes a transition out of this state to State 10, **Wait for Update**;
- (E19) Coordinate_Failure: This is an external event caused by the reception of an error to the previously issued **IN-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 **IN-CoordinateShadowUpdate** request and is ready to send an **IN-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 an **IN-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 **IN-UpdateShadow** operation. This event causes a transition out of this state to State 8, **SDF Bound**.

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

The SDF Finite State Machine for a shadow supplier in case of consumer-initiated (SDSM-ShSCi) is depicted in Figure 5.

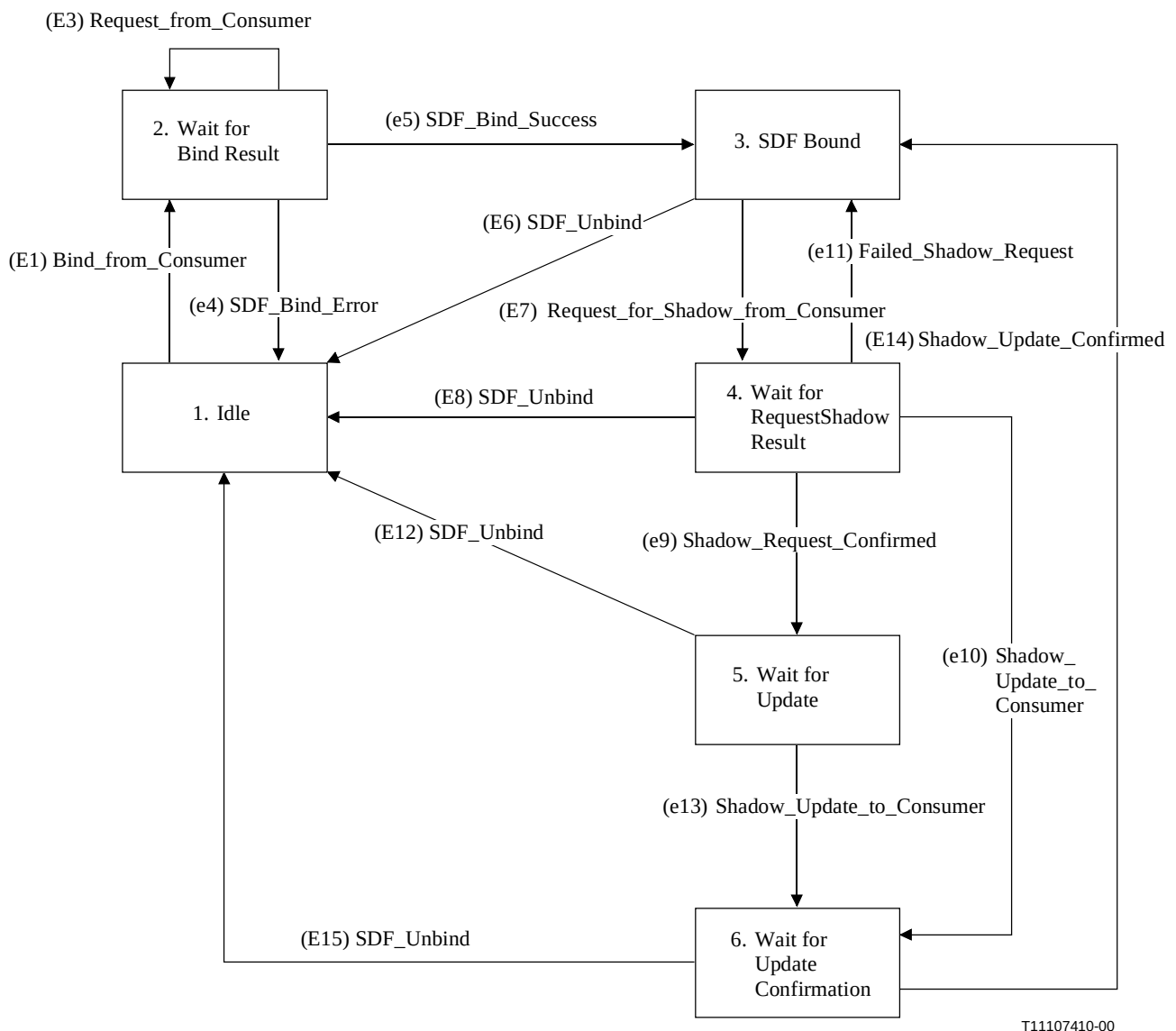


Figure 5/Q.1248.5 – SDF FSM for a shadow supplier in case of consumer-initiated (SDSM-ShSCi)

State 1: "Idle"

The only event accepted in this state is:

- (E1) Bind_from_Consumer: This is an external event, caused by the reception of an **IN-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, an **IN-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 an **IN-RequestShadowUpdate** operation before the result from the **IN-DSAShadowBind** operation is determined. This occurs when the **IN-RequestShadowUpdate** is sent in the same TCAP message as the bind request. The **IN-RequestShadowUpdate** message is stored and a transition occurs to the same state. When a transition occurs to the next state, the **IN-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 **IN-DSAShadowBind** operation previously issued from the consumer. An **IN-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 **IN-DSAShadowBind** operation previously issued from the consumer. This event causes an **IN-DSAShadowBind** result to be returned, and a transition out of this state to State 3, **SDF Bound**.

State 3: "SDF Bound"

In this state, the supplier SDF is expecting an **IN-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-Unbind** 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 an **IN-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 an **IN-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 a response to the previously received **IN-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 **IN-UpdateShadow** message is ready to be sent. This causes the sending of both an **IN-RequestShadowUpdate** result and an **IN-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 **IN-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 **IN-RequestShadowUpdate** operation and is ready to send an **IN-UpdateShadow** operation to the consumer SDF. 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_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**.