**DRAFT**

pr **ETS 300 670**

**August 1996**

**EUROPEAN TELECOMMUNICATION STANDARD**

Source: ETSI TC-SPS

Reference: DE/SPS-02004

ICS: 33.080

**Key words:** IN, UPT

# Universal Personal Telecommunication (UPT) phase 1; Intelligent Network (IN) Capability Set 1 (CS1); Application of core Intelligent Network Application Protocol (INAP)

**ETSI**

European Telecommunications Standards Institute

**ETSI Secretariat**

**Postal address:** F-06921 Sophia Antipolis CEDEX - FRANCE
**Office address:** 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
**X.400:** c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

# Contents

## Foreword

This draft European Telecommunication Standard (ETS) has been produced by the Signalling Protocols and Switching (SPS) Technical Committee of the European Telecommunications Standards Institute (ETSI), and is now submitted for the Public Enquiry phase of the ETSI standards approval procedure.

This ETS is based on the information contained in ETR 066 [9], ETS 300 374-1 [2] and ETS 300 374-5 [3].

## Introduction

This ETS describes the application of core INAP as specified in ETS 300 374-1 [2] and ETS 300 374-5 [3] for UPT phase 1. Clauses 1 to 3 contain general information. Clause 4 describes the network requirements for UPT. In clause 5 the application contexts used for UPT are listed. Clause 6 describes both the UPT information model and the UPT security model as used in the SDF. In clause 7 the UPT specific behaviour of the SCF is described for every UPT procedures, using both a textual and an SDL description. Clause 8 contains the description of the operations exchanged between networks on the SCF-SDF interface. This clause together with clause 6 constitutes the core part of the UPT specification. The other clauses are included in the ETS because clauses 6 and 8 are not self-explanatory.

| **Proposed transposition dates** | |
|---|---|
| Date of latest announcement of this ETS (doa): | 3 months after ETSI publication |
| Date of latest publication of new National Standard or endorsement of this ETS (dop/e): | 6 months after doa |
| Date of withdrawal of any conflicting National Standard (dow): | 6 months after doa |

Blank page

# 1 Scope

This European Telecommunication Standard (ETS) specifies the application of core INAP for the UPT service and describes the internetwork interface. It is applicable to UPT phase 1 as defined in ETR 055-2 [6]. It is mainly based on the information in ETR 066 [9] and ETS 300 374-1 [2] and ETS 300 374-5 [3]. The UPT service relies on the IN architecture as described in ETS 300 356-15 [1].

# 2 Normative references

This ETS incorporates by dated and undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this ETS only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

[1] ETS 300 356-15 (1995): "Integrated Services Digital Network (ISDN); Signalling System No.7; ISDN User Part (ISUP) version 2 for the international interface; Part 15: Diversion supplementary services [ITU-T Recommendation Q.732, clauses 2 to 5 (1993), modified]".

[2] ETS 300 374-1 (1994): "Intelligent Network (IN); Intelligent Network Capability Set 1 (CS1), Core Intelligent Network Application Protocol (INAP); Part 1: Protocol specification".

[3] ETS 300 374-5 (1996): "Intelligent Network (IN); Intelligent Network Capability Set 1 (CS1), Core Intelligent Network Application Protocol (INAP), Part 5: Protocol specification for the Service Control Function (SCF) - Service Data Function (SDF) interface".

[4] ETS 300 287 -1 (1996): "Integrated Services Digital Network (ISDN); Signalling System No.7; Transaction Capabilities (TC) version 2; Part 1: Protocol specification".

[5] ETS 300 391-1: "Universal Personal Telecommunication (UPT); Specification of the security architecture for UPT phase 1; Part 1: Specification".

[6] ETR 055-2: "Universal Personal Telecommunication (UPT); The service concept; Part 2: General service description".

[7] ETR 055-3: "Universal Personal Telecommunication (UPT); The service concept; Part 3: Service aspects of charging, billing and accounting".

[8] ETR 060 (1995): "Signalling Protocols and Switching (SPS); Guidelines for using Abstract Syntax Notation One (ASN.1) in telecommunication application protocols".

[9] ETR 066 (1993): "Universal Personal Telecommunication (UPT); Requirements on information flows and protocols".

[10] ETR 164 (1995): "Integrated Services Digital Network (ISDN); Intelligent Network (IN); Interaction between IN Application Protocol (INAP) and ISDN User Part (ISUP) version 2".

[11] ETR 315 (1996): "Universal Personal Telecommunication (UPT); UPT access parameters and identities for UPT users".

[12] ITU-T E.164: "Telephone Network and ISDN operation; Numbering, Routing and Mobile Service; Numbering Plan for the ISDN era".

[13] ITU-T Recommendation Q.1213: "General Recommendations on Telephone Switching and Signalling; Intelligent Network, Global Functional Plane for Intelligent Network CS-1".

[14] ITU-T Recommendation X.501 (1993): "Information Technology; Open Systems Interconnection; The Directory: Models".

# 3 Definitions and abbreviations

## 3.1 Definitions

No definitions have been identified.

## 3.2 Abbreviations

For the purposes of this ETS, the following abbreviations apply:

| | |
|---|---|
| AC | Application Context |
| BCSM | Basic Call State Model |
| CD | Call Deviation |
| CFB | Call Forwarding on Busy |
| CFNR | Call Forwarding on No Reply |
| CFU | Call Forwarding Unconditional |
| CS1 | Capability Set 1 |
| DEREG_IN | Deregistration for incoming calls |
| DTMF | Dual Tone Multi Frequency |
| FRI | Feature Request Identification |
| FSM | Finite State Model |
| IA | Identification and Authentication |
| IN | Intelligent Network |
| INAP | Intelligent Network Application Protocol |
| INCALL | Incoming Call |
| IP | Intelligent Peripheral |
| ISDN | Integrated Services Digital Network |
| $n_S$ | Sent part of the sequence number, i.e. the 16 least significant bits |
| OUTCALL | Outgoing Call |
| PIN | Personal Identification Number |
| PSTN | Public Switched Telephone Network |
| PUI | Personal User Identity |
| REG_IN | Registration for incoming calls |
| SCF | Service Control Function |
| SCSM | SCF State Model |
| SDF | Service Data Function |
| SDFh | Home Service Data Function |
| SDFo | Originating Service Data Function |
| SIB | Service Independent Building Blocks |
| SLP | Service Logic Program |
| SLPI | Service Logic Program Invocation |
| SPIN | Special Personal Identification Number |
| SPM | Service Profile Modification |
| SRF | Specialised Resource Function |
| SSF | Service Switching Function |
| SSP | Service Switching Point |
| TCAP | Transaction Capabilities Application Part |
| UPT | Universal Personal Telecommunication |
| UPTAC | UPT Access Code |
| UPTAN | UPT Access Number |

# 4 UPT phase 1 requirements

The UPT phase 1 service is a set of UPT features that can be implemented without major changes to current technology, and is basically restricted to provision in PSTN and ISDN, with voice and telephony type services (see ETR 055-2 [6]). This clause includes a number of operational requirements.

### 4.1 Architecture requirements

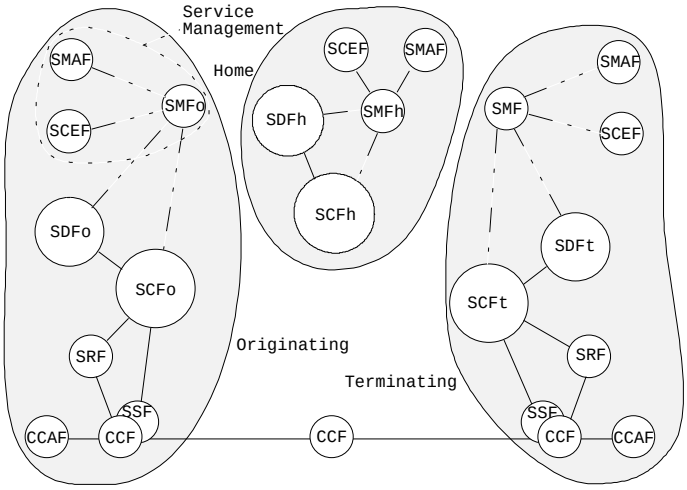


**Figure 1: General UPT functional architecture.**

Figure 1 gives an overview of the general UPT functional architecture. Apart from standard Intelligent Network (IN) terminology, the following notations are used in figure 1:

**SCFh** Home SCF;
**SDFh** Home SDF;
**SMFh** Home SMF;
**SCFo** Local ("visited") SCF, originating side;
**SDFo** Local ("visited") SDF, originating side;
**SMFo** Local ("visited") SMF, originating side;
**SCFt** Local ("visited") SCF, terminating side;
**SDFt** Local ("visited") SDF, terminating side.

The functional architecture for UPT Phase 1 is described in figure 2.

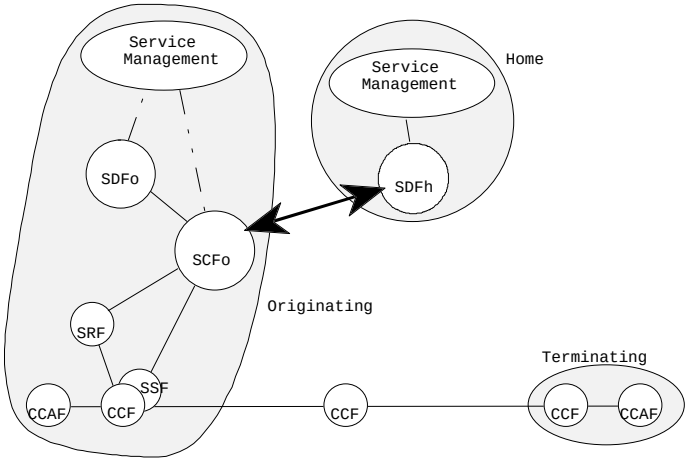


**Figure 2: Phase 1 UPT functional architecture.**

The differences with the general UPT functional architecture are :

a) The interconnection of networks takes place between the SCFo and SDFh functional entities, as indicated by the arrow on the figure. The interface between SCF and SDF is specified in IN CS1;

b) SDFh stores all data related to the UPT user (i.e. the database in UPT Phase 1 is centralized);

c) SDFh must as a consequence provide access control functions to check whether or not requests received from remote entities are authorized requests or not;

d) SDFh performs the authentication of the UPT user;

e) SDFo stores a list of agreements, which indicates the identity of all the service providers whose subscribers are allowed to access UPT service in SDFo's network;

f) SDFo stores a list of service limitations resulting from agreements with service providers or network limitations;

g) SDFo also stores information related to the management of the UPT service in its network, e.g. charging records which will be used later on for accounting.

### 4.2 Requirements on the network

The UPTAC, UPTAN and UPT number shall be recognized by the SSP.

#### 4.2.1 Requirements on the originating network side

The requirements on the originating network side are:

a) Signalling systems used on the UNI will be those for the PSTN and ISDN. For interaction with the user, user-information is to be collected in-band by means of Dual Tone Multi Frequency (DTMF). This means that for ISDN the D-channel is not used for collecting user-information.

b) In the PSTN, DTMF devices or terminals have to be used, because e.g. the non-fixed length numbers are ended with an #.

c) ISDN terminals have to be provided with DTMF functionality, otherwise it would not be possible to carry in-band information for user interactions.

#### 4.2.2 Requirements on the terminating network side

Signalling systems used at the UNI will be those for the PSTN and ISDN.

#### 4.2.3 Requirements on the fixed network

Feature interactions with line services are outside the scope of this ETS. It is covered in ETR 164 [10].

## 5 UPT Application Contexts

The UPT phase 1 service shall use the application contexts as defined in ETS 300 374-1 [2] and ETS 300 374-5 [3]. The following application contexts are used at the interfaces with the SCF:

SSP-SCP

Core-INAP-CS1-SSP-to-SCP-AC
Core-INAP-CS1-assist-handoff-SSP-to-SCP-AC
Core-INAP-CS1-SCP-to-SSP-AC
Core-INAP-CS1-SCP-to-SSP-traffic-management-AC
Core-INAP-CS1-SCP-to-SSP-service-management-AC

IP-SCP

Core-INAP-CS1-IP-to-SCP-AC

SCP-SDP

Core-INAP-CS1-SCP-to-SDP-AC

# 6 UPT Information Model

## 6.1 Introduction

The UPT phase 1 needs a significant amount of data stored in the SDFs. Data for UPT are contained in the following data model. The aim of that model is twofold: first to provide a list of all the data needed to support UPT phase 1 from the service and secondly to present the data as formally as possible so that they are ready to be used as parameters of the database operations.

Due to the amount of information contained in the data model, the model needs to be formally organized. The information and its associated structure make up the UPT information base (UPT-IB). The contents of the data model is described in subclause 6.2.1 and its structure in subclause 6.2.2. The generic information base on which the present UPT-IB is based is described in ETS 300 374-5 [3]. In that document the different classes are specified.

The organization of information provided with the UPT-IB does not imply any physical mapping of information even though some specific mappings will facilitate the use of the model.

Subclause 6.2 defines the objects, their attributes and the relations between them are specified according to ETR 060 [8]. This part of the data model is UPT-specific (but could probably be extended to other services). The complete ASN.1 module gathering the ASN.1 definitions of the next clauses is provided in annex A.

## 6.2 UPT Information Base

### 6.2.1 Information Base

The information model has been organized in object classes. Each object class is a general representation of an object of telecommunications (service, user, subscriber...). An object is an instance of the object class. Each object class is characterized by attributes. The attributes contain the data needed to fulfil the service.

Several object classes have been identified as well as their attributes. Figure 3 gives the inheritance relationships between the different object classes. All the object classes are subclasses of **top** which is an abstract class from which all the other classes are subclasses.

Apart from **top**, 8 types of object classes have been identified:

1) alias;
2) UPT provider;
3) agreed service;
4) user profile;
5) basic service
6) user profile alias;
7) routeing service;
8) registered routeing service (one for each type of routeing service).

They are described in more detail in the following subclauses. The classes **top** and **alias** are part of the ITU-T Recommendation X.501 [14] and are therefore not described within this ETS.

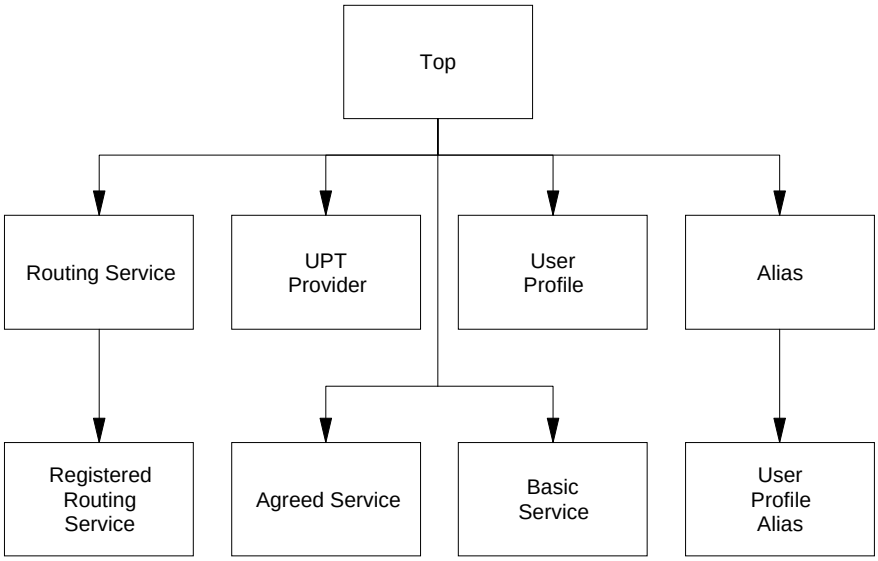


**Figure 3: Inheritance for the object classes**

#### 6.2.1.1 UPT provider

This object-class defines a UPT provider. It gives all the information concerning the provider that is necessary to support the UPT service. The definition involves:

- identifying the provider;

The following ASN.1 description shall be used to define the UPT provider object class:

```
uptProvider  OBJECT-CLASS ::= {
    MUST CONTAIN {providerId}
    ID  id-oc-uptProvider}

providerId  ATTRIBUTE ::= {
    WITH SYNTAX AddressString {ub-providerId}
    EQUALITY MATCHING RULE  numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    SINGLE VALUE    TRUE
    ID  id-at-providerId}

AddressString {INTEGER: ub-max-value} ::= NumericString (SIZE (1..ub-max-value))
```

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is as follows:

- The **providerId** attribute identifies the UPT provider. This identifier is a numeric string. It may be part of a numbering plan. In case of the home service provider, it shall be possible to get the **providerId** value from a translation of the UPT number or the PUI.

#### 6.2.1.2 Agreed Service

This object class gives the service, that is provided to home users by a visited provider, and its restrictions of use. The definition involves:

- identifying the service;

- giving the restrictions on the use of the service. The restrictions on a service under agreement might be different from those on a service proposed by a provider to its subscribers.

The following ASN.1 description shall be used to define the agreed service object classes:

```
agreedService  OBJECT-CLASS ::= {
    MUST CONTAIN {
        providedServiceId|
        providedLocations}
    ID  id-oc-agreement}

providedServiceId  ATTRIBUTE ::= {
    WITH SYNTAX Service
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE    TRUE
    ID  id-at-providedServiceId}

Service ::= INTEGER {
--basic services 0-9
    isdnTelephony (0),
-- registration service 10-19
    icRegistration (10),
--profile service 20-29
    serviceProfileModification (20),
-- charging service 30-39
    standard (30),
-- routing service 40-49
    callForwardingUnconditional (40),
    callForwardingOnNoReply (41),
    callForwardingOnBusy (42),
    variableRoutingOnTime (43),
    variableRoutingOnCallingLine (44)}

providedLocations  ATTRIBUTE ::= {
    WITH SYNTAX AddressString{ub-locations}
    EQUALITY MATCHING RULE  numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringMatch
    ID  id-at-providedLocations}
```

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is as follows:

- the **providedServiceId** attribute gives the service identifier;

- the **providedLocations** attribute gives the service restrictions on the use of the service, in particular the destination of a service (geographic coverage zone for a unique provider). The restrictions are specified at the provider level. They give the limits on the provision of a service agreed between the provider and other providers for the roaming users. If any other restriction appears in the future, it should accompany the attribute presently described.

  The values contained in attributes describing ISDN addresses or part of them shall be built upon international addresses without the international prefix.

#### 6.2.1.3 User profile

This object class defines a user profile. The user profile gives the service information attached to one of the users in a subscription. This information may differ from one user to another in the same subscription. This definition involves:

- identifying the user;

- giving the list of the allowed services and locations for those services;

- giving service parameters for the allowed services.

The following ASN.1 description shall be used to define the user profile object class:

```
userProfile  OBJECT-CLASS ::= {
    MUST CONTAIN {
        pui|
        chargingAttributeSet|
        allowedServices|
        allowedCFParameters|
        nbOfFailedAuthentication}
    MAY CONTAIN {
        userPassword|   -- defined in X.509
        specialPassword|
        variablePassword|
        callInfoRecords}
    ID  id-oc-userProfile}

pui  ATTRIBUTE ::= {
    WITH SYNTAX         AddressString{ub-pui}
    EQUALITY MATCHING RULE  numericStringMatch
    SINGLE VALUE        TRUE
    ID              id-at-pui}

specialPassword  ATTRIBUTE ::= {
    WITH SYNTAX         OCTET STRING (SIZE (0..ub-special-password)
    EQUALITY MATCHING RULE  octetStringMatch
    ID              id-at-specialPassword}

variablePassword  ATTRIBUTE ::= {
    WITH SYNTAX         OCTET STRING (SIZE (0..ub-variable-password)
    EQUALITY MATCHING RULE  octetStringMatch
    ID              id-at-variablePassword}

nbOfFailedAuthentications  ATTRIBUTE ::= {
    WITH SYNTAX         INTEGER (1..ub-max-nbOfFailedAuthentications)
    ORDERING MATCHING RULE  integerOrderingMatch
    SINGLE VALUE        TRUE
    ID              id-at-nbOfFailedAuthentications}

chargingAttributeSet  ATTRIBUTE ::= {
    defaultChargingReference|
    userCredit|
    activeChargingService}

defaultChargingReference  ATTRIBUTE ::= {
    WITH SYNTAX             IsdnAddress,
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    SINGLE VALUE            TRUE
    ID                  id-at-defaultChargingReference}

IsdnAddress ::=AddressString{ub-international-isdn-number}

userCredit  ATTRIBUTE ::= {
    WITH SYNTAX             INTEGER (1..ub-maxUserCredit)
    ORDERING MATCHING RULE      integerOrderingMatch
    SINGLE VALUE            TRUE
    ID                  id-at-userCredit}

callInfoRecords  ATTRIBUTE ::= {
    WITH SYNTAX CallInfoRecord
    ID      id-at-callInfoRecords}

CallInfoRecord ::= SEQUENCE {
    authenticationTime      [0] UTCTime,
    callStopTimeValue       [1] UTCTime,
    callStartTimeValue      [2] UTCTime,
    callingAddressValue     [3] IsdnAddress,    calledNumber        [4] IsdnAddress,
    duration        [5] INTEGER (0..2147483647) OPTIONAL,
    routingAddress      [6] IsdnAddress OPTIONAL,
    forwardedToAddress      [7] IsdnAddress OPTIONAL,
    invokedSupplementaryServices    [8] CFServices OPTIONAL,
    visitedNetwork      [9] NetworkCode OPTIONAL,
    callCost        [10] Cost OPTIONAL,
    surcharges      [11] Cost OPTIONAL,
    releaseCause        [12] Cause OPTIONAL}
Cost ::= CHOICE {
    pulse   [0] INTEGER (1..ub-pulse),
    cost    [1] CurrencyValue}
CurrencyValue::=CHOICE {
    usDollar [0] Currency,
    frenchFranc [1] Currency,
    germanMark [2] Currency,
    dutchGuilder [3] Currency,
    italianLira [4] Currency,
```

```
    englishPound [5] Currency,
    spanishPeseta [6] Currency,
    swedishKrone [7] Currency,
    norwegianKrone [8] Currency}
Currency::= REAL
CFServices ::= SET OF Service (40..49)
Cause ::= OCTET STRING (SIZE(lb-causeLength..ub-causeLength))

activeChargingService  ATTRIBUTE ::= {
    WITH SYNTAX     Service (30..39)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE    TRUE
    ID          id-at-activeChargingService}

allowedServices  ATTRIBUTE ::= {
    WITH SYNTAX         Service
    EQUALITY MATCHING RULE  integerMatch
    ID              id-at-allowedServices}

allowedCFParameters  ATTRIBUTE ::= {
    WITH SYNTAX         CFParameter
    EQUALITY MATCHING RULE  integerMatch
    ID              id-at-allowedCFParameters}

CFParameter ::= INTEGER {
    notifyActivation (0),
    notifyForwarding (1),
    notifyCallingPartyWithNumber (2),
    notifyCallingPartyWithoutNumber (3),
    notifyForwardedTo (4)}
```

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is as follows:

- The **pui** (personal user identifier) attribute is a number used to identify the UPT user. The PUI may be made-up of a country code, a network code and a personal identifier (see ETR 164 [11]). It should be possible to retrieve the provider identifier of the user from the PUI.

- Several attributes are related to security. The **Password** attribute which could be of three types depending on the type of authentication available to the user gives the password used by the user to authenticate. The **userProfile** object class should at least contain one type of **Password** attribute. The **nbOfFailedAuthentications** gives the remaining number of authentications that can be failed before the identifier is blocked. This number is limited by a value controlled by the subscriber or the provider.

- The **chargingAttributeSet** attribute-set contains several attributes related to charging. The **defaultChargingReference** attribute gives the default reference point for charging. The reference point may be specified with an **IsdnAddress** type. The **userCredit** gives the credit still available to the user. It also permits the update of the value of the **userCredit** attribute (however this calculation cannot be performed on-line by the SDF that is not able to do it). The **activeChargingService** attribute indicates which charging service has been selected by the user and should be used to charge his calls.

- The **callInfoRecords** attribute contains all the call records related to a given user. It is used to keep track of the use of the service by a given user.

- The **allowedServices** attribute gives the list of the services subscribed by the user and also the places where the services are available (locations are defined as contexts of the different service values).

- The **allowedCFParameters** attribute provides a list of the call forwarding parameters that have been subscribed by the user for the different call forwarding services.

#### 6.2.1.4 User profile alias

This object class also describes the user profile. It is used to have another naming path for the user using in that case the UPT number. This definition involves:

- identifying the user;

- referring to the object (userProfile) that really contains the user profile information.

The following ASN.1 description shall be used to define the user profile alias object class:

```
userProfileAlias  OBJECT-CLASS ::= {
SUBCLASS OF {alias}
MUST CONTAIN {uptNumber}
ID  id-oc-userProfileAlias}

uptNumber  ATTRIBUTE ::= {
    WITH SYNTAX             IsdnAddress
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    SINGLE VALUE            TRUE
    ID                  id-at-uptNumber}
```

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is as follows:

- the **uptNumber** attribute is the dialable number through which the user may be reached. The format of this attribute is an ISDN address (see ITU-T Recommendation E.164 [12]);

#### 6.2.1.5 Basic service

This object class defines the registration addresses attached to a given basic service. The definition involves:

- identifying the service;
- giving the registration addresses for incoming and outgoing calls.

The following ASN.1 description shall be used to define the basic service object class:

```
basicService  OBJECT-CLASS ::= {
    MUST CONTAIN {
        basicServiceId|
        icRegistrationAddress|
        allowedDestinations|
        allowedRegistrationAddress}
    ID  id-oc-basicService}

basicServiceId  ATTRIBUTE ::= {
    WITH SYNTAX         Service (0..9)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE        TRUE
    ID              id-at-basicServiceId}

icRegistrationAddress  ATTRIBUTE ::= {
    WITH SYNTAX             IsdnAddress
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID                  id-at-icRegistrationAddress}

allowedRegistrationAddress  ATTRIBUTE ::= {
    WITH SYNTAX             AddressString{ub-locations}
    EQUALITY MATCHING RULE      numericString
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID              id-at-allowedRegistrationAddress}

allowedDestinations  ATTRIBUTE ::= {
    WITH SYNTAX             AddressString{ub-locations}
    EQUALITY MATCHING RULE      numericString
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID              id-at-allowedDestinations}
```

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is as follows:

- The **basicServiceId** attribute gives the service name or its identifier to which the registration addresses are attached.

- The **icRegistrationAddress** attribute gives the registration addresses for incoming calls and outgoing calls. It also contains the default registration addresses which is a value of the attribute with the default context. The other values have a time context that indicates the time validity of the values.

- The **allowedRegistrationAddress** attribute contains complete or initial part of international ISDN addresses corresponding to areas where the user can be registered. The **icRegistrationAddress** attribute should take its values within the values of that attribute.

- The **allowedDestinations** attribute contains complete or initial part of international ISDN addresses corresponding to areas to which the user can set-up a call.

#### 6.2.1.6 Routeing service

The **routingService** object class contains the information related to the routeing services in general. This object class is an abstract object class that is not used for the provision of the UPT service. It allows the building of the **registeredRoutingService** object classes (see next clause).

The definition involves:

- identifying the routeing service;
- giving the parameters of the routeing service.

The following ASN.1 description shall be used to define the routeing service object class:

```
routingService  OBJECT-CLASS ::={
    KIND    abstract
    MUST CONTAIN {
        routingAddress|
        allowedRoutingAddress|
        activationStatus}
    MAY CONTAIN {
        activatedCFParameters}
    ID  id-oc-routingService}

routingAddress  ATTRIBUTE ::= {
    WITH SYNTAX             IsdnAddress
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID                  id-at-routingAddress}

allowedRoutingAddress  ATTRIBUTE ::= {
    WITH SYNTAX             IsdnAddress
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID                  id-at-routingAddress}

activationStatus  ATTRIBUTE ::= {
    WITH SYNTAX         ActivationStatus
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE        TRUE
    ID              id-at-activationStatus}

ActivationStatus ::= INTEGER {
    notActivated (0),
    activated (1)}

activatedCFParameters  ATTRIBUTE ::= {
    WITH SYNTAX         CFParameter
    EQUALITY MATCHING RULE  integerMatch
    ID              id-at-activatedCFParameters}
```

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is as follows:

- the **routingAddress** is expressed as an international ISDN number according to the ITU-T Recommendation E.164 [12]. It indicates the address to which the call should be routed depending on the time of the day or on the calling party or on the status of the called user. This address should be within the limits authorized to the user by his subscriber. The permitted values are contained in the **allowedRoutingAddress** attribute. The **allowedRegistrationAddress** and the **allowedRoutingAddress** attributes could be one and the same and in that case it should be a collective attribute;

- the **activationStatus** attribute indicates whether a service is activated or not. This information may be modified by the user, using service profile modification procedures;

- the **activatedCFParameters** attribute gives the provision parameters for the call forwarding services selected by the user. The parameters can only be taken out of the subscribed parameters contained in the **allowedCFParameters** attribute.

#### 6.2.1.7 Registered routeing services

The object classes defined in this subclause are the specialisation of the abstract object class defined in the previous subclause. Each class is associated to a unique routeing service. The routeing services considered in UPT phase 1 are as follows:

- call forwarding unconditional;
- call forwarding on no reply;
- call forwarding on busy;
- variable routeing on time;
- variable routeing on calling line.

According to the routeing services 5 object classes are defined. Compared with the parent class only the attributes specific from a routeing service are added.

The following ASN.1 description shall be used to define the routeing service object classes:

```
cfuService  OBJECT-CLASS ::={
    SUBCLASS OF     {routingService}
    MUST CONTAIN {
        cfuServiceId}
    ID  id-oc-cfuService}

cfnrService  OBJECT-CLASS ::={
    SUBCLASS OF     {routingService}
    MUST CONTAIN {
        cfnrServiceId|
        noReplyConditionTimer}
    ID  id-oc-cfnrService}

cfbService  OBJECT-CLASS ::={
    SUBCLASS OF     {routingService}
    MUST CONTAIN {
        cfbServiceId}
    ID  id-oc-cfbService}

vrtService  OBJECT-CLASS ::={
    SUBCLASS OF     {routingService}
    MUST CONTAIN {
        vrtServiceId}
    ID  id-oc-vrtService}

vrclService  OBJECT-CLASS ::={
    SUBCLASS OF     {routingService}
    MUST CONTAIN {
        vrclServiceId}
    ID  id-oc-vrclService}

cfuServiceId  ATTRIBUTE ::= {
    WITH SYNTAX         Service (40)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE        TRUE
    ID              id-at-cfuServiceId}

cfnrServiceId  ATTRIBUTE ::= {
    WITH SYNTAX         Service (41)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE        TRUE
    ID              id-at-cfnrServiceId}

cfbServiceId  ATTRIBUTE ::= {
    WITH SYNTAX         Service (42)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE        TRUE
    ID              id-at-cfbServiceId}

vrtServiceId  ATTRIBUTE ::= {
    WITH SYNTAX         Service (43)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE        TRUE
    ID              id-at-vrtServiceId}
```

```
vrclServiceId  ATTRIBUTE ::= {
    WITH SYNTAX         Service (44)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE        TRUE
    ID              id-at-vrclServiceId}

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

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is as follows:

- the **routingServiceId** attribute indicates the routeing service described in the object class;

- the **noReplyConditionTimer** attribute gives the time after which a call is considered as not answered.

### 6.2.2 Structure of the UPT information model

#### 6.2.2.1 Existence relations between classes

Figure 4 gives the object classes used to define the UPT data model. The lines between the object classes indicate existence relations between two object classes. A existence relation expresses the fact that an object class does not exist by itself, it needs the object classes put above in the figure to have a meaning. For example, a **userProfile** object class is not a stand alone object class, it is subordinated to the existence of a **uptProvider** object class. Since the **Top** object class always exists, the **uptProvider** object class can exist by itself.

A existence relation is not a one to one relation. The instance of a "superior" object class can be associated to several instances of the "inferior" object class. For example, a subscriber profile can be linked with several user profile, since one subscriber can have several users within its subscription. In the present case, all the relations shown in figure 4 are one to n relations.

The use of the terms "superior" and "inferior" does not imply any class relation. It only means a instance of the "inferior" class has no meaning if the instance of the "superior" class does not exist.

This type of relations between classes has nothing to do with the naming relations that will be described in the following subclause. It has no direct effect on the ASN.1 notation, but should be considered when creating new class instances during O&M operations.

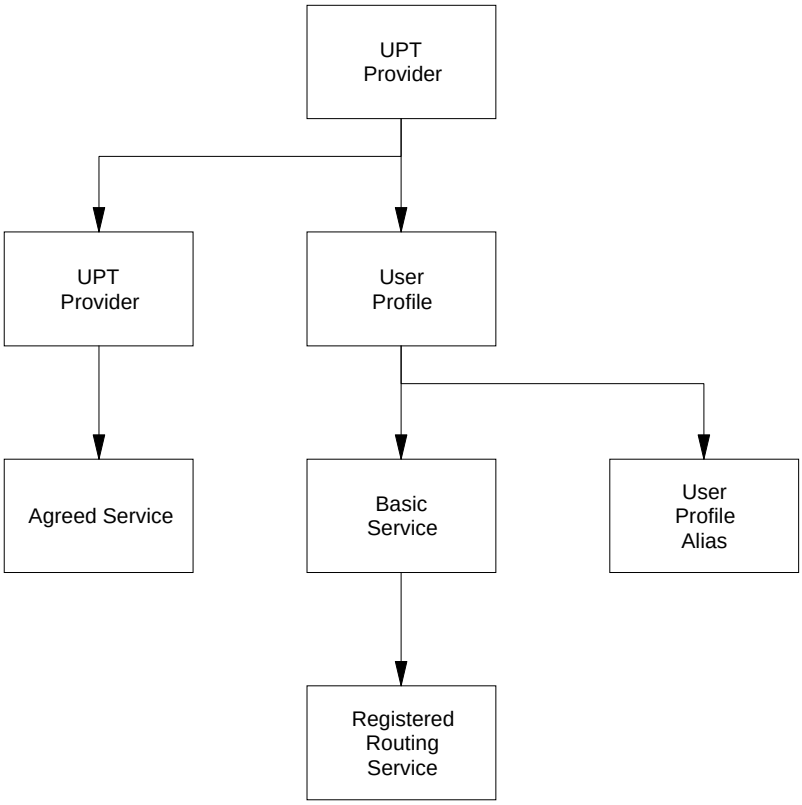


**Figure 4: Existence relationships between object classes**

#### 6.2.2.2 Name forms

For each object class, the name forms define the attributes which will be involved in the naming of the object class. This attribute will identify instances of the object class. The naming attribute is chosen so that instances of the object class can be uniquely addressed. The naming attribute should be a mandatory attribute of the object class.

The following ASN.1 description may be used to define the naming attributes of all the object classes defined in the previous chapter. The name form gives the object class to be named and its naming attribute. For the object classes defined in the previous subclause, the following name forms shall be used:

```
uptProviderNameForm  NAME-FORM ::= {
    NAMES       uptProvider
    WITH ATTRIBUTES {providerId}
    ID          id-nf-uptProviderNameForm}

agreedServiceNameForm  NAME-FORM ::= {
    NAMES       agreedService
    WITH ATTRIBUTES {providedServiceId}
    ID          id-nf-agreedServiceNameForm}

userProfileNameForm  NAME-FORM ::= {
    NAMES       userProfile
    WITH ATTRIBUTES {pui}
    ID          id-nf-userProfileNameForm}

userProfileAliasNameForm  NAME-FORM ::= {
    NAMES       userProfileAlias
    WITH ATTRIBUTES {uptNumber}
    ID          id-nf-userProfileAliasNameForm}

basicServiceNameForm  NAME-FORM ::= {
    NAMES       basicService
    WITH ATTRIBUTES {basicServiceId}
    ID          id-nf-basicServiceNameForm}
```

```
cfuServiceNameForm  NAME-FORM ::= {
    NAMES       cfuService
    WITH ATTRIBUTES {cfuServiceId}
    ID          id-nf-cfuServiceNameForm}

cfnrServiceNameForm  NAME-FORM ::= {
    NAMES       cfnrService
    WITH ATTRIBUTES {cfnrServiceId}
    ID          id-nf-cfnrServiceNameForm}

cfbServiceNameForm  NAME-FORM ::= {
    NAMES       cfbService
    WITH ATTRIBUTES {cfbServiceId}
    ID          id-nf-cfbServiceNameForm}

vrtServiceNameForm  NAME-FORM ::= {
    NAMES       vrtService
    WITH ATTRIBUTES {vrtServiceId}
    ID          id-nf-vrtServiceNameForm}

vrclServiceNameForm  NAME-FORM ::= {
    NAMES       vrclService
    WITH ATTRIBUTES {vrclServiceId}
    ID          id-nf-vrclServiceNameForm}
```

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is given below:

- The **providerId** attribute is the naming attribute for the **uptProvider** object class.

- The **providedServiceId** attribute is the naming attribute for the **agreedService** object class.

- The **uptNumber** attribute and the **pui** attribute are respectively used to name the **userProfile** and the **userProfileAlias** object classes.

- The **routingServiceId** attribute is the naming attribute for the **registeredRoutingService** object class.

#### 6.2.2.3 Structure rules

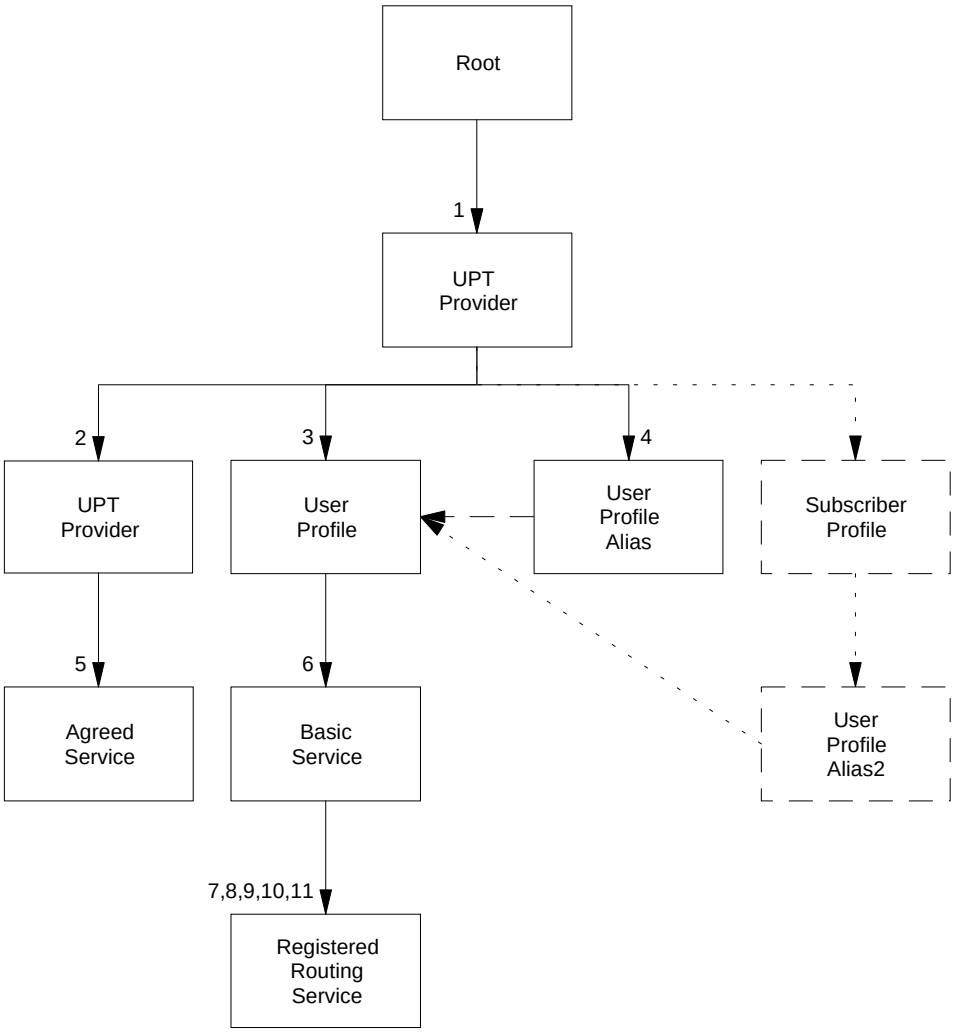


**Figure 5: Naming structure for UPT**

In the database, a data item is located with the name of the object to which it belongs. The name of the object is the concatenation of the names of the objects superior to it in the naming structure. This implies that a hierarchical structure exists between the objects to create the object names. The structure rules provide the relationships between the objects in the naming context. This structure is independent of the structure defined for the object-classes in the inheritance context and in the existence context.

The relationships between the object classes are represented by lines in figure 5.

To build a name to access a given object, it is necessary to follow a path defined on the figure. For example to access **userProfile** the path is (1,3). Each object is uniquely named. However an object can have another name through the use of an alias like for **userProfile**. The **userProfile** can be named with the **userProfileAlias** that directly points to **userProfile**. The dashed arrow shows the relationship between the alias and the object class it represents whereas the plain arrows show the structure rules.

The object classes **subscriberProfile** and **UserProfileAlias2** (see figure 5) have been introduced to show how the concept of subscriber could be integrated in the model without modifying it. The two classes are not described in this ETS, because they are not accessed through the SCF-SDF interface. They are not used for the UPT on-line service, even though they may be used in the UPT service. The dotted lines are used to represent the object classes linked to the subscriber since they are not integral part of this ETS.

The following ASN.1 description shall be used to define the structure rules for the global naming of all object classes:

```
sr1 STRUCTURE-RULE ::= {
    NAME FORM   uptProviderNameForm
    ID      1}

sr2 STRUCTURE-RULE::= {
    NAME FORM       uptProviderNameForm
    SUPERIOR RULES  {sr1}
    ID          2}

sr3 STRUCTURE-RULE::= {
    NAME-FORM       userProfileNameForm
    SUPERIOR RULES  {sr1}
    ID          3}

sr4 STRUCTURE-RULE::= {
    NAME FORM       userProfileAliasNameForm
    SUPERIOR RULES  {sr1}
    ID          4}

sr5 STRUCTURE-RULE::= {
    NAME FORM       agreedServiceNameForm
    SUPERIOR RULES  {sr2}
    ID          5}

sr6 STRUCTURE-RULE::= {
    NAME FORM       basicServiceNameForm
    SUPERIOR RULES  {sr3}
    ID          6}

sr7 STRUCTURE-RULE::= {
    NAME FORM       cfuServiceNameForm
    SUPERIOR RULES  {sr6}
    ID          7}

sr8 STRUCTURE-RULE::= {
    NAME FORM       cfnrServiceNameForm
    SUPERIOR RULES  {sr6}
    ID          8}

sr9 STRUCTURE-RULE::= {
    NAME FORM       cfbServiceNameForm
    SUPERIOR RULES  {sr6}
    ID          9}

sr10 STRUCTURE-RULE::= {
    NAME FORM       vrtServiceNameForm
    SUPERIOR RULES  {sr6}
    ID          10}

sr11 STRUCTURE-RULE::= {
    NAME FORM       vrclServiceNameForm
    SUPERIOR RULES  {sr6}
    ID          11}
```

The correspondence between the parts of the definition given above and the various pieces of the notation introduced by the object class is given below:

- The **uptProvider** object class can be accessed using the **providerId** attribute (relation 1).

- The **uptProvider** object class can also be accessed using the **providerId** attribute from the **uptProvider** object class (relation 2). In that case it represents the providers with whom the home provider has agreements.

- The **agreedService** object class can be accessed using the **providedServiceId** attribute from the **uptProvider** object class (relation 5).

- The **userProfile** object class can be accessed using the **pui** attribute from the **uptProvider** object class (relation 3) and its alias can be accesses using the **uptNumber** attribute from the same object class (relation 4).

- The **basicService** object class can be accessed using the **basicServiceId** attribute from the **usererProfile** object class (relation 6).

- The **registeredRoutingService** object classes can be accessed using the **routingServiceId** attributes from the **basicService** object class (relations 7, 8, 9, 10, 11).

## 6.3 UPT Security model

### 6.3.1 Basic access control

Different classes of access rights exist. Those classes are determined by the type of agent accessing them. These three agents are considered here:

a) the home UPT service provider;
b) the visited UPT service provider;
c) the UPT user.

NOTE: The subscriber could be another agent but was not considered in the description of the model. The management system could also have an access to the database and could also be considered as an agent. Both the subscriber and the manager could have specific access rights.

The access rights are described according to the rules given in ETS 300 374-5 [3]. The access rights do not only depend on the type of agent, they also depend on the type of authentication that has been performed. The user can identify and authenticate himself with three different authentication procedures that give him different rights. The user can modify his profile according to the rights attached to the subscription that rules his use of the service. The visited service providers acting on their own do not use authentication when accessing the home database, they have the lowest access rights and can only retrieve information concerning the routeing of the call. Whether the home service provider needs to authenticate is a local matter, because he can always be recognized with the access points he uses. He has all the rights on the data he owns.

The different access rights needed to access a data of the information model are defined in ETR 060 [8]. However this information is only indicative and the access rights given in this subclause try to follow the description given in ETR 060 [8]. They are also indicative, but they are given in this ETS to show the use of the access control information. They should be specified according to the security model contained in ETS 300 391-1 [5].

Three subentries can be defined to specify the access rights, one for each type of agents. The first one is for the home service provider that can perform all kinds of operations on data in the SDF. The second one is for the visited service providers that can only read routeing information for all the users. The third one is for the user that can only access his own data which he can modify according to his subscription.

The two subentries are instanciations of the **subentry** object class and as such contained the commonName and subentrySpecification attributes. The first subentry which is attached to the root administrative point has the following values for its two attributes:

```
wholeTree   DirectoryString ::= {
    universalString providerSubtree}

wholeSubtreeSpecification SubtreeSpecification ::={}

-- this indicates that this subtree corresponds to the whole tree
```

The second subentry is attached to the service provider administrative point and has the following values for its attributes:

```
usersTree   DirectoryString ::= {
    universalString usersSubtree}

usersSubtreeSpecification SubtreeSpecification ::={
    base    {{{ type    providerId,
              value   ???}}}, --provided when instanciated
    minimum 2}
```

NOTE: The ??? notation is used to represent values that have to be provided at instanciation time.

The subtree specification indicates that only the objects basic service and registered routeing service for all the users should be considered in this subtree.

The third subtree is attached to the user administrative point and has the following values for its attributes:

```
userTree    DirectoryString ::= {
    universalString userSubtree}

userSubtreeSpecification SubtreeSpecification ::={
    base    {{{ type    providerId,
            value   ???}},
        {{  type    pui,
            value   ???}}},
    minimum 1}
```

The subtree specification indicates that this subtree is the subtree under the user profile object.

The access rights for the first subtree that is used to define the access rights of the home service provider can be applied to the whole subtree. The service provider can have all the rights on the data he owns. Those access rights are expressed and contained in a **prescriptiveACI** attribute with the following value:

```
homeProviderSubentryACI ACIItem ::= {
    identificationTag   {
        universalString id-homeProviderACI},
    precedence      ???,   -- the definition of the precedence is a local matter
    authenticationLevel {
        basicLevels {
            level   none}}, -- a local qualifier could be added
    itemOrUserFirst {
        userFirst   {
            userClasses {
                name    {{dn    {{{type providerId,
                        value   ???}}}}}},
            userPermissions {{
                protectedItems  {
                    subtree wholeSubtreeSpecification},
                grantsAndDenials    {
                    grantAdd,
                    grantRead,
                    grantDiscloseOnError,
                    grantRemove,
                    grantBrowse,
                    grantModify,
                    grantRename,
                    grantReturnDN,
                    grantCompare,
                    grantFilterMatch}}}}}
```

NOTE: The attributes that can be modified by the user through service profile modification, could be protected against changes made by the service provider, but since the service provider defines and enforces the access control, he has all the rights.

The access rights for the second subtree that is used to define the access rights of the service providers are different for the home service provider. Those rights apply to the users subtree. The service providers do not perform authentication. They have only read access to the user information. Those access rights are expressed and contained in a **prescriptiveACI** attribute with the following value:

```
otherProviderSubentryACI    ACIItem ::= {
    identificationTag   {
        universalString id-otherProviderACI},
    precedence      ???, -- see above
    authenticationLevel {
        basicLevels {
            level       none}},
    itemOrUserFirst {
        userFirst   {
            userClasses {
                userGroup   {{dn    {{{type providerId,
                        value   ???}},
                        {{type  providerId,
                        value   ???}}}}}},
-- this example contains the name of one service provider instead of the name of all the
-- service providers
            userPermissions {{
                protectedItems  {
                    subtree usersSubtreeSpecification},
                grantsAndDenials    {
                    grantRead,
```

```
                    grantDiscloseOnError,
                    grantBrowse,
                    grantReturnDN,
                    grantCompare,
                    grantFilterMatch}}}}}}
```

NOTE: The access rights could be refined by restricting the access to only the attributes needed for the routeing. But the purpose of this ACIItem is more to show the general contents of the access control than a real ACIItem that will be implementation-dependent anyway.

To specify the user's access rights, it is possible to use a subtree **userTree** defined above. The rights are determined by the type of authentication procedure performed by the user. The access rights can be expressed with the following values of an **entryACI** object:

```
userSubentryACI ACIItem ::= {
    identificationTag   {
        universalString id-userACI},
    precedence      ???,
    authenticationLevel {
        basicLevels {
            level       simple,
            localQualifier  ???}},
-- a local qualifier needs to be added for simple authentication to differentiate PIN, SPIN
-- and strong authentication.
    itemOrUserFirst {
        userFirst   {
            userClasses {
                name    {{dn    {{{type providerId,
                        value   ???}},
                        {{type  pui,
                        value   ???}}}}}},
            userPermissions {{
                protectedItems  {
                    subtree userSubtreeSpecification},
                grantsAndDenials    {
                    grantRead,
                    grantDiscloseOnError,
                    grantBrowse,
                    grantModify,
                    grantReturnDN,
                    grantCompare,
                    grantFilterMatch}}}}}}
```

This **ACIItem** only gives the general rights for a user on all the user subtree, but for the different objects of the subtree stronger access control procedures may apply. For example, for the **userProfile** object class, restrictions on access should be defined. The restrictions are defined thanks to the addition of an attribute in the object class. They prevent a user from modifying his authentication information, his charging records and the options of his subscription. The attribute is of class **entryACI** and may take the following value:

```
userProfileACI  ACIItem ::= {
    identificationTag   {
        universalString id-userProfileACI},
    precedence      ???,
    authenticationLevel {
        basicLevels {
            level       simple,
            localQualifier  ???}},
    itemOrUserFirst {
        userFirst   {
            userClasses {
                thisEntry   NULL,
            userPermissions {{
                protectedItems  {
                    entry   NULL},
                grantsAndDenials    {
                    denyModify,
                    grantRead,
                    grantBrowse}}}}}}
```

For the SPIN authentication, the right for the user to modify his password should be added. The different **ACIItem** should be included in the different subentries or entries of the DIT to enforce the access rights control.

### 6.3.2 Authentication

For the weak authentication, the information sent to identify and authenticate the user are his PUI and his PIN. To fit the BIND operation description, the name contains the PUI. It is the name used to access the user data in the database, the concatenation of the providerId and of the PUI (where the provider identity has been removed). The PIN is transported in the password parameter and is not protected. This gives the following credentials for a user trying to identify himself with the weak authentication:

```
pinCredentials Credentials ::= {
    simple  {
        name    {{{
            type    providerId,
            value   ???},
            {type   pui,
            value   ???}}}, -- user's name
        password    {
            unprotected ???}}} -- user's password (pin)
```

The algorithm used for this type of authentication is just a comparison between the stored PIN and the provided PIN. When the user has used the weak authentication, he has the simple access rights with the local qualifier as defined by his service provider to differentiate this type of simple authentication from the other types.

The SPIN authentication leads to the same results as with the PIN. The only differences are that the unprotected password should contain the SPIN instead of the PIN. And once the SPIN authentication has succeeded, a local qualifier for the access rights is allocated to the user. This local qualifier shall be different from the one used for the PIN authentication.

For the strong authentication, the information sent to identify and authenticate the user are the PUI, the algorithm identifier, a sequential number and the authentication code. The PUI is used as it is done for the PIN authentication. The sequential number is carried in the validity parameter as a random number and the algorithm identifier with the authentication code is carried as a protected password. This gives the following credentials for a user trying to identify himself with the strong authentication:

```
strongCredentials Credentials ::= {
    simple  {
        name    {{{
            type    providerId,
            value   ???},
            {type   pui,
            value   ???}}},
        validity    {
            random1 ns},
        password    {
            protected   {
                algorithmIdentifier ???,
                encrypted       authenticationCode}}}}
```

The algorithmIdentifier parameter takes the values that describe the algorithm. The algorithm identifiers should be defined internationally. When the algorithm is specific to the service provider, the **externalProcedure** credentials should be used.

When the user has authenticated with the strong authentication procedure, he is granted the simple access rights with the local qualifier different from the ones given to the other simple authentications.

### 6.3.3 Permitted Values

The **icRegistrationAddress** attribute of the **userProfile** object class is an attribute with permitted values and with a restricted number of values. The permitted values are contained in the **allowedRegistrationAddress** attribute. It is exactly the same thing with the **routingAddress** attribute of the **registeredRoutingService** object class whose permitted values are contained in the **allowedRoutingAddress**. The relationships between the **icRegistrationAddress** and the **allowedRegistrationAddress** attributes can be expressed in a ACItem as follows:

```
registrationAddressACI  ACIItem ::={
    identificationTag   {
        universalString id-registrationAddressACI},
    precedence      ???,
    authenticationLevel {
        basicLevels {
            level       simple,
            localQualifier  ???}},
    itemOrUserFirst {
        itemFirst   {
            protectedItems  {
                restrictedBy    {{type      icRegistrationAddress,
                        valuesIn    allowedRegistrationAddress}},
                maxValueCount   {{type      icRegistrationAddress,
                            maxCount    ???}}},
                                -- decided at subscription time
            itemPermissions {{
                userClasses {
                    name    {{dn    {{{type providerId,
                            value   ???}},
                            {{type  pui,
                            value   ???}}}}}},
                grantsAndDenials    {
                    grantAdd,
                    grantRead,
                    grantDiscloseOnError,
                    grantRemove,
                    grantBrowse,
                    grantModify,
                    grantRename,
                    grantReturnDN,
                    grantCompare,
                    grantFilterMatch}}}}}}
```

The **ACIItem** for the **routingAddress** attribute would be similar to the one just described above.

# 7 SCF procedures

## 7.1 General

### 7.1.1 Overview

This subclause is an introduction to the UPT-specific procedures in the SCF.

The UPT-specific procedures described in this clause form the UPT-specific service logic program (SLP) needed in the SCF to handle the UPT procedures as described in ETR 066 [9]. An SLP is normally implementation-dependent and should not be standardized in general. However, this ETS gives a global view of the UPT service and puts in context the database requests that have to be standardized.

These procedures are:

a) The common sequences: These elementary procedures are independent of the actual UPT procedures but are executed before and after them. They include the Identification and Authentication procedure (IA), the Feature Request Identification procedure (FRI), the Release (RELEASE) procedure, the SRF Connection procedure (SRF_Connect) and the SRF Disconnection macro (SRF_Disconnect). They are further described in subclause 7.2.

b) The personal mobility procedures: This includes the Registration for Incoming Calls procedure (REG_IN) and the Deregistration for Incoming Calls (DEREG_IN). They are further described in subclause 7.3.

c) The UPT call handling procedures: This includes the Outgoing UPT Call procedure (OUTCALL) and the Incoming UPT Call procedure (INCALL). They are further described in subclause 7.4.

d) The service profile management procedures: This is only the Service Profile Modification procedure (SPM) which is further described in subclause 7.5.

e) The PIN modification procedure: This procedure is called the PIN Change procedure (PIN_CHANGE). It is further described in subclause 7.6.

The UPT-specific SLP is invoked when an instance of the SCF State Model (SCSM) is created in the SCF FSM on receipt of an IN request related to the UPT service (i.e. indicated by the value of the "serviceKey" parameter of the InitialDP operation). This occurs when the SSF detects the presence of an incoming UPT call or of a UPT user request. The recognition of a UPT request is based on the format of the UPT number, UPT access code or UPT access number. This format is specified in ETR 315 [11]. This part of the SLP is also called the Access procedure and is executed in all the SLP invocations (SLPI).

The FSM behaviour is described in ETS 300 374-1 [2] and is partly driven by the SLP.

The description technique for the SLP in this clause uses both text and SDLs. The SLP is modelled by a single SDL process type UPT_SLP. To each SLPI corresponds a process instance. The UPT_SLP process is further described in subclause 7.1.3.

### 7.1.2 Charging procedures in the SDLs

Several types of charging procedures have been defined in ETS 300 374-1 [2]. Depending on the operations used, the charging is performed in the SCF or in the SSF:

a) "FurnishChargingInformation" is used if call records are generated at the SSF side;

b) "ApplyCharging" is used if call records are generated at the SSF side and collected by the SCF. It is used when real-time transfer of charging information is necessary.

In the present ETS all the types of charging are included. However the choice of the type of charging procedure is an implementation choice and some charging operations may not be relevant for a given implementation.

### 7.1.3 Conventions and notation

Although the TC interface is not subject to standardization the use of some conventions for representing it in terms of events which virtually occur at this interface. Such events are used as input and output signals of the SDL description.

For that purpose, the TC interface is modelled using the following pseudo-events:

a) Events to TC:

   1) Dialogue_Released(x): The dialogue with the functional entity x has been released by the FSM or by the peer;

   2) <Operation_Name>.inv: A valid TC_Invoke_Ind primitive for <Operation_Name> operation has been received by the FSM;

   3) <Operation_Name>.res: A valid TC_Result_L_Ind primitive for <Operation_Name> operation has been received by the FSM;

   4) <Operation_Name>.err: A valid TC_U_Error_Ind primitive for <Operation_Name> operation has been received by the FSM. This signal has a formal parameter which represents the type or error being received;

   5) <Operation_Name>.rej: A valid TC_U_Reject_Ind primitive for <Operation_Name> operation has been received by the FSM.

b) Events from TC:

1) Release_Dialogue(x): The FSM is requested to release the dialogue with functional entity x;

2) <Operation_Name>.inv: The FSM is requested to invoke the <Operation_Name> operation;

3) <Operation_Name>.res: The FSM is requested to send a positive result for the <Operation_Name> operation;

4) <Operation_Name>.err: The FSM is requested to send an error for the <Operation_Name> operation. This signal has a formal parameter which represents the type of error being reported.

These conventions assume that the SCF FSM performs some logical transformations on the primitives received and sent on the TC/INAP interface. These transformations are such that:

a) from a receiving point of view, only the events which have an impact on the FSM are represented as input signals in the model of the TC interface;

b) if several events in the FSM have the same impact , they are combined into a single input signal in the TC interface model;

c) from a sending point of view, only the events which cannot be autonomously triggered by the FSM (according to ETS 300 374-1 [2]), are defined as output signal in the TC interface model.

These transformations and the associated assumptions are summarized in table 1 and 2.

Table 1 indicates for each TC service primitive which can be received by the FSM, the corresponding event on the TC interface and (if any) the subsequent request passed to TC by the FSM according to ETS 300 374-1 [2].

Table 2 indicates for each primitive which can be passed to TC by the FSM, whether it is generated autonomously by the FSM (according to the rules defined in ETS 300 374-1 [2], clause 10) or at the request of the SLP. In the former case, the associated signal passed by the FSM is indicated. In the latter case the name of the corresponding event on the TC interface is also provided.

**Table 1: Events on the TC interface on reception of TC-primitives**

| Input on the TC/FSM interface | Corresponding output on the TC interface | Associated subsequent output on FSM/TC interface |
|---|---|---|
| TC-Begin-Ind | (SLPI creation) | - |
| TC-Continue-Ind Initial | Bind.res[1)] | - |
| TC-Continue-Ind Subsequent | - | - |
| TC-End-Ind | Dialogue_Released(x) | - |
| TC-Notice-Ind | Dialogue_Released(x) | TC-End-Req (Local) |
| TC-U-Abort-Ind | Bind.err [2)] or Dialogue_Released(x) | |
| TC-P-Abort-Ind | Dialogue_Released(x) | - |
| TC-Invoke-Ind (Valid) | <Operation_Name>.inv | - |
| TC-Invoke-Ind (Invalid) | Dialogue_Released(x) | TC-U-Reject-Req, TC-End-Req |
| TC-Result-Ind (Valid) | <Operation_Name>.res | - |
| TC-Result-Ind (Invalid) | Dialogue_Released(x) | TC-U-Reject-Req, TC-End-Req |
| TC-U-Error-Ind (Valid) | <Operation_Name>.err | - |
| TC-U-Error-Ind (Invalid) | Dialogue_Released(x) | TC-U-Reject-Req, TC-End-Req |
| TC-U/R-Reject-Ind | <Operation_Name>.rej[3)] | - |
| TC-L-Reject-Ind | Dialogue_Released(x) | TC-End-Req |
| TC-L-Cancel-Ind (Class 1,3) | Dialogue_Released(x) | TC-U-Abort-Req |
| TC-L-Cancel-Ind (Class 2,4) | - | - |

**Table 2: Origins of TC-primitives**

| Output to TC | Originated by | Subsequent output from FSM to SLP |
|---|---|---|
| TC-Begin-Req | FSM Initiated | - |
| TC-Continue-Req | FSM Initiated | - |
| TC-End-Req (local) | FSM Initiated | Dialogue_Released(x) |
| TC-End-Req (basic) | Release_Dialogue(x) from SLP | - |
| TC-End-Req (basic) | FSM Initiated | Dialogue_Released(x) |
| TC-U-Abort-Req | FSM Initiated | Dialogue_Released(x) |
| TC-Invoke-Req | <Operation_Name>.inv received from SLP | - |
| TC-Result-Req | <Operation_Name>.res received from SLP | - |
| TC-U-Error-Req | - FSM Initiated (parameter missing, unexpected parameter,...), or<br>- <Operation_Name>.err (error.name) received from SLP | - |
| TC-U-Reject-Req | FSM Initiated, followed by TC-End-Req (basic) to TC (if the dialogue exists) | Dialogue_Released(x) |

---

1) If a Bind-Result PDU is received
2) If a Bind-Error PDU is received
3) If it appears that this signal is of no use in the SLP, it can be replaced by Dialogue_Released

The following BNF description summarizes the convention used for naming the signals exchanged across the TC interface:

<Internal_Signal> := <Dialogue_Control_Event> | <Operation_Event>

<Dialogue_Control_Event> := <Dialogue_Event_Name> <Functional_Entity>

<Dialogue_Event_Name> := "Dialogue_Released" | "Release_Dialogue"

<Functional_Entity> := "SSF" | "SRF" | "SDFo" | "SDFh" | "SDFhA" | "SDFhB"

<Operation_Event> := <Operation_Name> | <Result> | <Error> | <Reject>

<Result> := <Operation_Name> ".res"

<Error> := <Operation_Name> ".err"

<Reject> := <Operation_Name> ".rej"

<Operation_Name> := "P&C" | "PLAYANN" | "FURNCHGINFO" | "CONNTORES" | "APPLYCHG" | "INITIALDP" | "CONNECT" | "REQREPBCSM" | "ETC" | "ARI" | "EVREPBCSM" | "BIND" | "SEARCH" | "MODIFY" | "ADD" | "REMOVE" | "RELEASECALL" | "SRFRPT" | "DISCFWDCONN"

Tables 3 and 4 give the mapping between the names used in the above convention and the actual operation names. Table 4 gives the name of the operations on the SCF-SDF interface. These operations have to be used to fulfil the UPT service. Table 3 describes the other operations of the core INAP. These operations may be used for the UPT service, however, other operations, even proprietary operations, could be used because they are used within one network whereas SCF-SDF operations may cross network boundaries.

**Table 3: Intra-network operations used by the UPT SLP**

| Operation Name | SDL Signal |
|---|---|
| ApplyCharging | APPLYCHG |
| ApplyChargingReport | APPLYCHGRPT |
| AssistRequestInstructions | ARI |
| Connect | CONNECT |
| ConnectToResource | CONNTORES |
| DisconnectForwardConnection | DISCFWDCONN |
| EstablishTemporaryConnection | ETC |
| EventReportBCSM | EVREPBCSM |
| FurnishChargingInformation | FURNCHGINFO |
| InitialDP | INITIALDP |
| PlayAnnoucement | PLAYANN |
| PromptAndCollectUserInformation | P&C |
| ReleaseCall | RELEASECALL |
| RequestReportBCSMEvent | REQREPBCSM |
| SpecializedResourceReport | SRFRPT |

**Table 4: Inter-network operations used by the UPT SLP**

| Operation Name | SDL Signal |
|---|---|
| Bind | BIND |
| Search | SEARCH |
| RemoveEntry | REMOVE |
| ModifyEntry | MODIFY |
| AddEntry | ADD |

The following additional conventions are used as far as the parameter representation is concerned:

a) The parameters of an INAP operation are given in a comment box next to the signal box associated with the INAP operation.

b) The ASN.1 value notation is used to describe these parameters. This notation is extended to support variable parameters (noted with the suffix Var). The type of these parameters is given in the core INAP as part of the operation definitions.

c) The optional parameters depending on implantation choices are not represented.

d) A counter in the SCF is used to count the number of unsuccessful attempts (failed requests) to prevent the service or the network from being misused. This counter is named Counter1.

e) A counter in the SCF is used to count the number of times a database operation is sent to an busy SDF to discard the operation after too many attempts. This counter is named Counter2. Together with this counter, a timer is used to wait before sending again a database operation to the busy SDF.

### 7.1.4 SLP description

The UPT SLP is represented by the behaviour of the SCF in the "UPT_SLP" process (see figure 6). This process calls several SDL procedures, each of them corresponds to one of the UPT procedures defined in ETR 066 [9]. These SDL procedures are further described from subclause 7.2 to subclause 7.6. This SLP reflects the options adopted by the UPT Stage 2 as described in ETR 066 [9]. The ordering of the operations might be different, especially if the SLP is derived from a stage 2 based on SIBs as described in ITU-T Recommendation Q.1213 [13].

An instance of the "UPT_SLP" process is created by the SCSM on receipt of an InitialDP invocation with the "serviceKey" parameter identifying the UPT service. The SCSM moves at the same time to the state "Preparing SSF Instructions". The detection point "Analysed Information" indicates the presence of an incoming UPT call or a UPT user request.

The SLPI starts with the Access procedure. If the "calledPartyNumber" includes the UPT access code (UPTAC) or the UPT access number (UPTAN), the "UPT_SLP" process enters the Identification and Authentication procedure and the user request is further processed. Otherwise it calls the INCALL procedure and stops once it has been executed.

The "UPT_SLP" process starts by calling the IA procedure to identify and authenticate the user. If the user has used his special PIN code (SPIN) to authenticate, a success in the authentication procedure will allow the user to directly change his PIN code using for this purpose the PIN procedure. After a successful authentication with the normal PIN code, the service logic executes the FRI procedure to get the feature requested by the user.

Then, depending on the feature code provided by the UPT user, the REG_IN, the DEREG_IN, the OUTCALL, the SPM or the PIN_CHANGE SDL procedure is called.

As soon as a procedure is terminated, the user can either identify a new request (new call to the FRI SDL procedure) or abandon the follow-on procedure. The follow-on procedure allows a user to perform a sequence of service features with one unique authentication. It is represented in the SDL diagram by a loop that goes from a successful service feature procedure to the FRI procedure.

At any stage of the follow-on procedure, the user can abandon. If he does so, the RELEASE procedure is called.

The user can also be released by the network with the RELEASE procedure, if authentication has failed, if an operation error has occurred or if the user has misused the UPT service. The follow-on procedure ends when the user has performed all his requests and abandons or when the user is released by the network.

In any state of the different procedures (except in the states of the user/network initiated release procedure), the user can abandon, he can be disconnected or errors for pending operations can occur. In the latter case, the dialogue with the SSF and the other functional entities are released before ending the process.

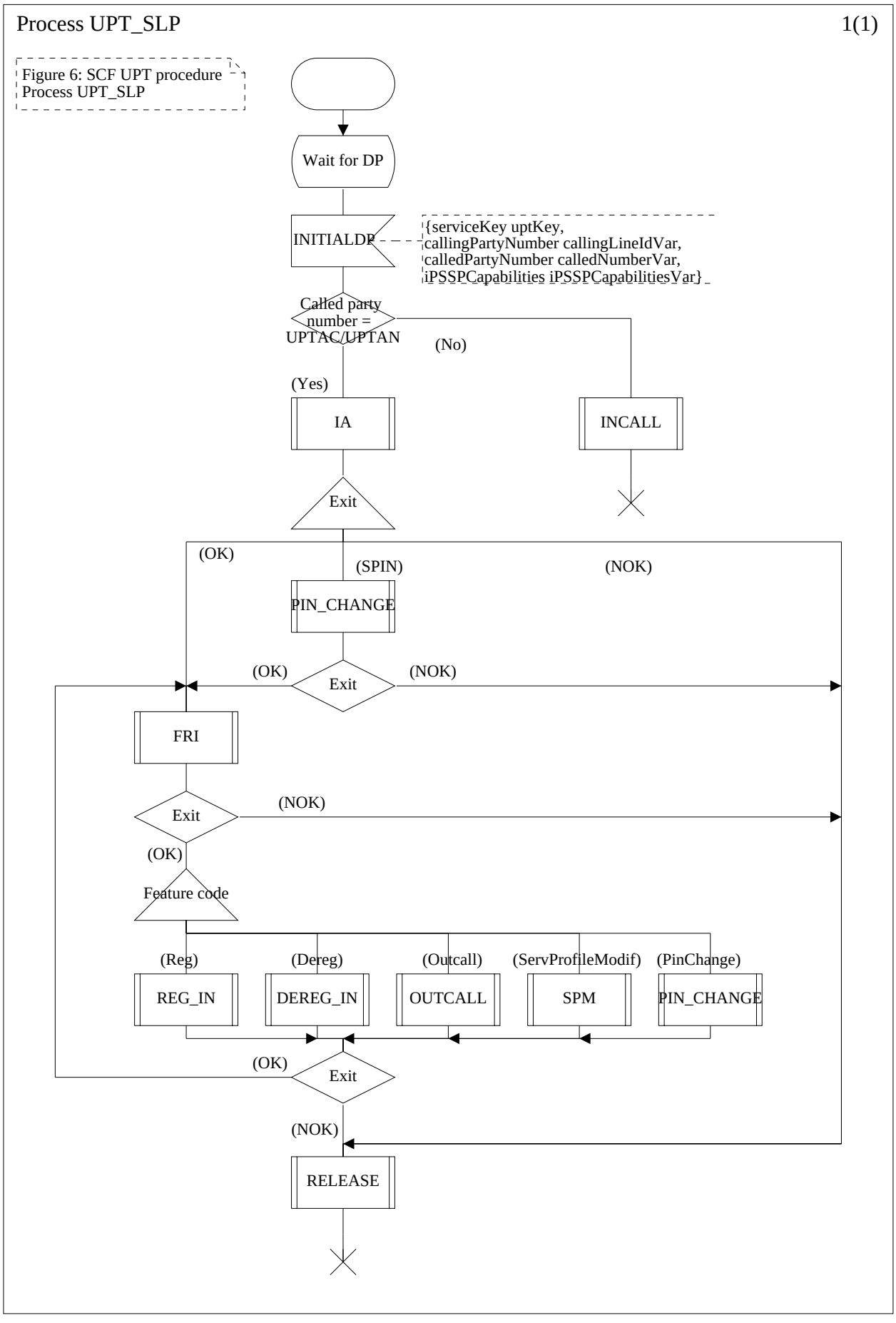


**Figure 6: Process UPT_SLP**

## 7.2 Generic sequences

This subclause regroups sequences of messages common to several procedures of UPT protocols. The information flows describing those sequences are originated from ETR 066 [9]. They are the basis to the SDL procedures presented in the following subclauses. In the SDL specification, the timers for the various states and operations are not described, only service related timers are shown.

### 7.2.1 Identification and authentication

#### 7.2.1.1 General

The identification and authentication (IA) procedure takes place each time a UPT user is requested to identify himself. This is always the case when the user asks for the access to the UPT service.

The user identifies himself with his personal identifier and secret codes depending on the security option chosen and the type of terminal. If the code matches the code stored in the database for the given identifier and for a predefined algorithm, the user gets access to the UPT service and other procedures can follow. The different authentication procedures are described in ETS 300 391-1 [5]

Within the SDL procedure describing the IA procedure, the user should have the possibility to make several identification attempts. The procedure has three logical outputs[4)] :

- OK: The procedure has succeeded and the user may proceed to the next procedure.

- NOK: The procedure has failed and the user is released by the network; the reason of this release can be either that the last permitted identification attempt has failed, that the maximum number of rejected requests is reached or that an error has occurred. The SCF and the SDF maintain a counter of the rejected attempts to have the user released by the network after a given number of consecutive retries to prevent misuse of the service and network. The other possibility to have this kind of output corresponds to the abandon of the request by the user.

- SPIN: The user has identified himself with his SPIN. This means that his authentication is immediately followed by a procedure to change his PIN code.

At any stage of the IA procedure, the user can abandon, he can be disconnected or errors for pending operations can occur. Therefore the SDL notation "State *" is used to show that those events can occur at any state of the procedure.

#### 7.2.1.2 Detailed procedure

Figure 7 shows the IA procedure.

The UPT request is notified to the SCF by a INITIALDP. The SSF has recognized the UPTAC (or UPTAN) in a user request. The purpose of the UPTAC (or UPTAN) is to identify the SCF that can handle UPT requests. This procedure and the structure of this number is presented in ETR 315 [11]. The need for an identification and authentication of the user has been recognized. The procedure can start. The SCF sends a REQREPBCSM to request the SSF to monitor a call-related event and to send a notification back to the SCF when the event is detected. For this procedure where only one party is involved, the only types of events to be notified are the user's abandon or his disconnection of the SCF. For charging purposes, the SCF sends also a FURNCHGINFO to instruct the SSF to create a call record. At the beginning of the procedure, the counter in the SCF (Counter1) for unsuccessful attempts (failed requests) to prevent the service or the network from being misused is reset.

---

**4)** A parameter named Exit is used to distinguish the two outputs. This parameter has three possible values, one for each output, OK, NOK and SPIN.

**Identification of the user**

Then the SCF asks for the connection of an SRF this is done through the "SRF Connect" procedure (SRF, see subclause 7.2.4.). Once the SRF is connected, the SCF is able to ask the user to provide his identity. This is done with a P&C. The SCF moves to the state "Waiting for user info". Three situations move it out of this state:

a) The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user release. The IA procedure is terminated and followed by a release procedure. This is included in the "State *" of the SDL diagram.

b) An error has occurred for the P&C operation or for the CONNTORES operation in the case of relayed operations: The SCF receives an error indication. This error indication includes a timer expiry, a rejection of an operation or an error response:

   1) if the error is of type UnavailableResource in response to the P&C operation, the implementer can (if it is possible) select another SRF and restart the connection procedure. This option which is implementer-dependent is not shown on the SDL diagram;

   2) if the error is an ImproperCallResponse error, the user is allowed to retry the authentication procedure from the beginning, but the number of retries is limited. A prompt informs him about his mistake and a counter of failed attempts is incremented;

   3) for the remaining errors, the IA procedure is terminated with Exit = NOK. The user is released.

c) The user has provided the data in a correct format: The SCF receives a P&C response and the procedure can continue as described below.

**Authentication**

In UPT phase 1, a user has three possible ways of authentication (simple PIN, SPIN or strong one-way ETS 300 391-1 [5]) depending on the authentication procedures that the user has subscribed and on the type of terminal available. To each authentication procedure corresponds a sequence of messages. The three sequences are:

a) PIN code authentication (manual): This is a two-step procedure. The user is requested to provide his identifier (PUI) and then to provide his PIN code via two consecutive P&Cs[5)]. As it was said previously the only error for P&C that receives special treatment is ImproperCallerResponse. It is followed by a procedure that informs the SSF of the failed attempt with a FURNCHGINFO and that allows the user to redial his authentication information or notifies him about a denied access to the UPT service depending on the number of failed attempts. This last procedure is used each time a user is suspected to have mistyped his identification sequence.

b) SPIN authentication (manual): This procedure is similar to the previous one (same message and same treatment). However, to be differentiated from the PIN procedure, the SPIN is preceded by the special digit * (the user has to dial two consecutive* digits in a row if he combines his identifier and his SPIN). This procedure is followed by a PIN procedure and only takes place when the PUI is blocked.

c) Strong one-way authentication (automatic): The user gives his identity and his variable authentication code at the same time. This is done automatically using a user device. The different pieces of data are combined with a separator "*".

---

**5)** The user has always the possibility to combine the 2 pieces of data or more (if wanted) in one dialling sequence using a separator * to separate the different pieces of data. By this means, the user speeds up the UPT procedures.

The use of a special leading digit ("*") differentiates the automatic authentication from the manual ones. After all the user information is received through a P&C response, the identity of the home service provider and the address of the database containing the user's information is extracted from the user identity[6)] . The procedure to obtain the database address is described in ETR 315 [11]. If this operation cannot be performed (i.e. the format of the user identity is wrong), the user is allowed to re-enter his authentication data following a procedure already described above. Once the identity of the home service provider is known, a database operation checks in the local database if an agreement exists between the local service provider and the service provider of the UPT user (home service provider). There are three outcomes to this query:

a) The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user's release. The IA procedure is terminated and followed by the release procedure.

b) An error has occurred: The SCF receives an error indication:

   1) If a service error of type Busy occurs, the implementer has the possibility (if desired) to resend the database operation to this same SDF after a given time set with a timer. This remark applies to all error configurations where this type of error can occur (i.e. when waiting for a database answer). The Counter2 counts the number of retries.

   2) Otherwise the IA procedure is terminated and followed by the release procedure.

c) A response to the database operation has been received: The content of the response is checked. If there is no agreement between the service providers, the SSF is informed of the end of the call and is requested to modify the call record by a FURNCHGINFO. The user is informed by a prompt that he cannot have access to the service and is later released. If there is an agreement between the service providers, a check with the home service provider takes place to authenticate the user and to know if the user is allowed to use the UPT service in that area.

If an agreement exists between the two service providers, the SCF starts an authenticated dialogue with the home database of the user. The dialogue is opened with a BIND operation whose argument contains all the authentication information provided by the user. The number of failed authentication attempts is limited for a given user's identifier. The SDFh monitors and keeps the count of the number of attempts. The counter used is attached to the PIN attribute. The three possible outcomes to this query are:

a) The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user release. The IA procedure is terminated and followed by a release procedure.

b) The BIND operation fails: The SCF receives an error indication:

   1) If the error is a service error (unavailable), the implementer can (if desired) resend the database operation to this same SDFh after a given time.

   2) If the error is a security error (blocked credentials), indicating that the maximum number of authentication attempts has been reached, the call record is updated with a FURNCHGINFO and before being released by the network, the user is informed by a PLAYANN that his line/number is blocked.

   3) If the error is a security error (invalidCredential), the SSF is requested to modify the call record (FURNCHGINFO) and the user is requested to restart dialling his identification information.

   4) If the error is a security error (inappropriateAuthentication), the SSF is requested to modify the call record (FURNCHGINFO) and the user is informed by a PLAYANN.

   5) Otherwise the IA procedure is terminated and followed by a release procedure.

c) The BIND operation succeeds: The user can go on to the next procedure (feature identification request, see subclause 7.2.2.).

---

**6)** It is assumed that the identifier contains enough information to know the home service provider and the corresponding SDF.

Once the authentication has been successfully performed, the SRF instructed by the SCF sends a prompt to the user requesting him to enter the code of the feature service he wants to access. This is only valid for all authentications except the SPIN one (see paragraph above) and when the user has not yet entered the feature code.

NOTE: To decrease the signalling load on an inter-network interface, a first database operation could be sent with the authentication information in the BEGIN message. The operation to be send would be the first database operation encountered in the FRI procedure (check on agreement between service providers). For the sake of simplicity, the different operations are treated separately.

Before a release after a prompt, the SRF is disconnected. The disconnection with a macro is described in subclause 7.2.5.

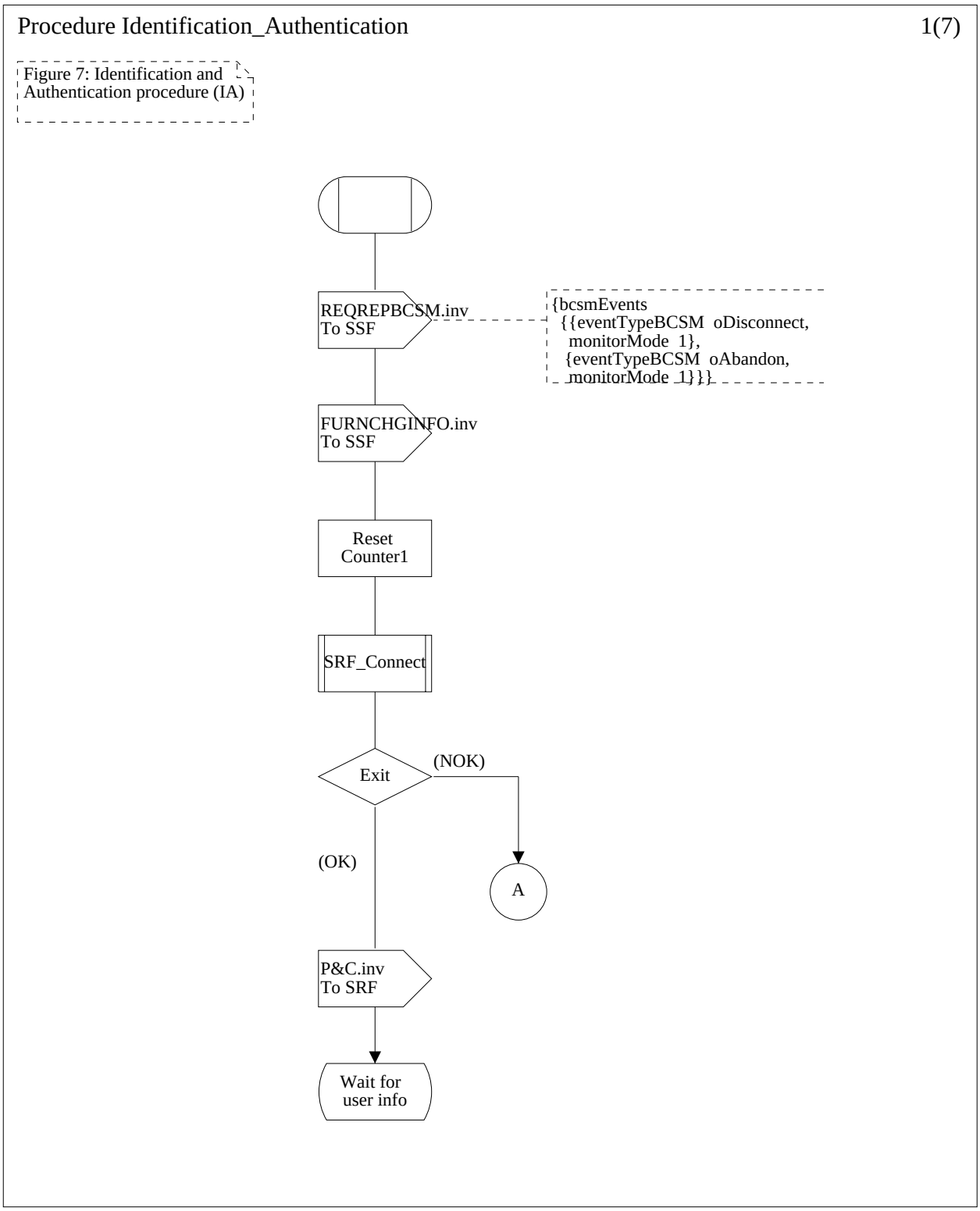


**Figure 7 (sheet 1 of 7): Identification and Authentication procedure (IA)**

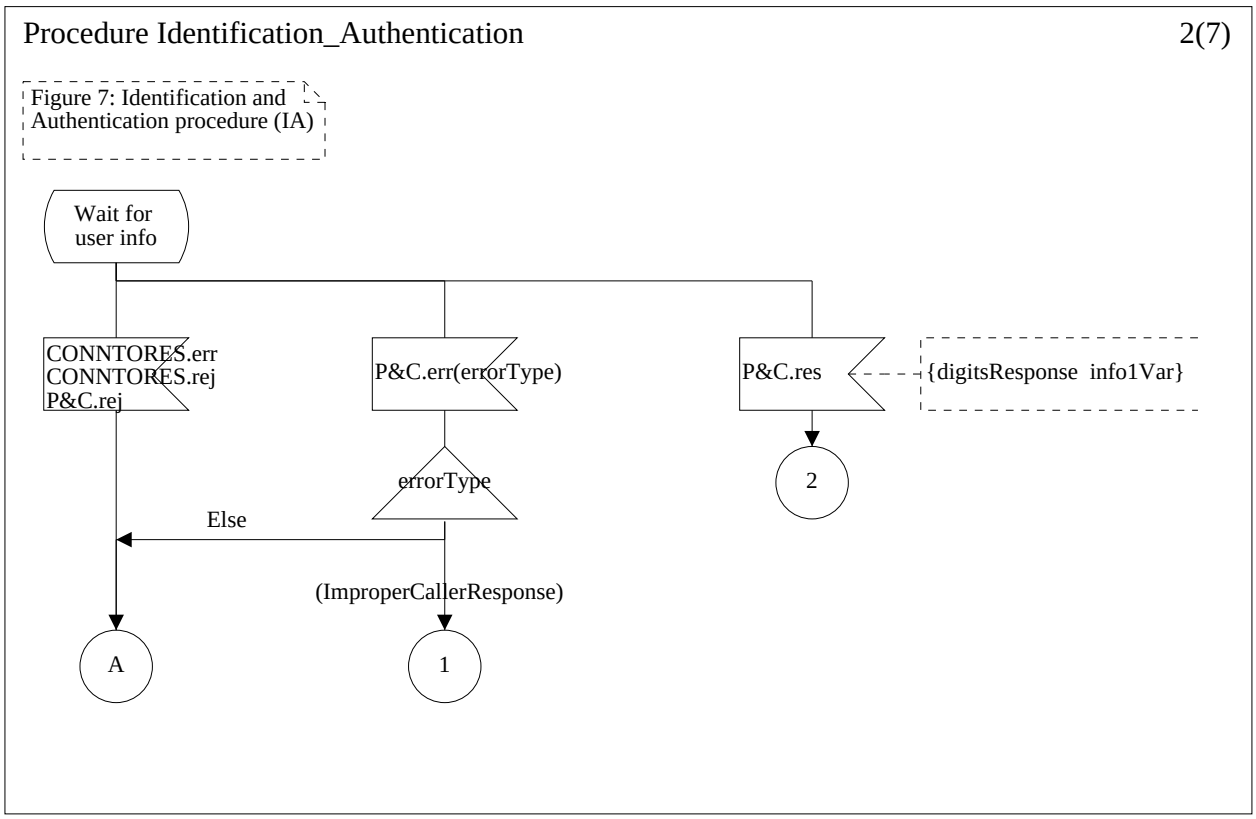


**Figure 7 (sheet 2 of 7): Identification and Authentication procedure (IA)**

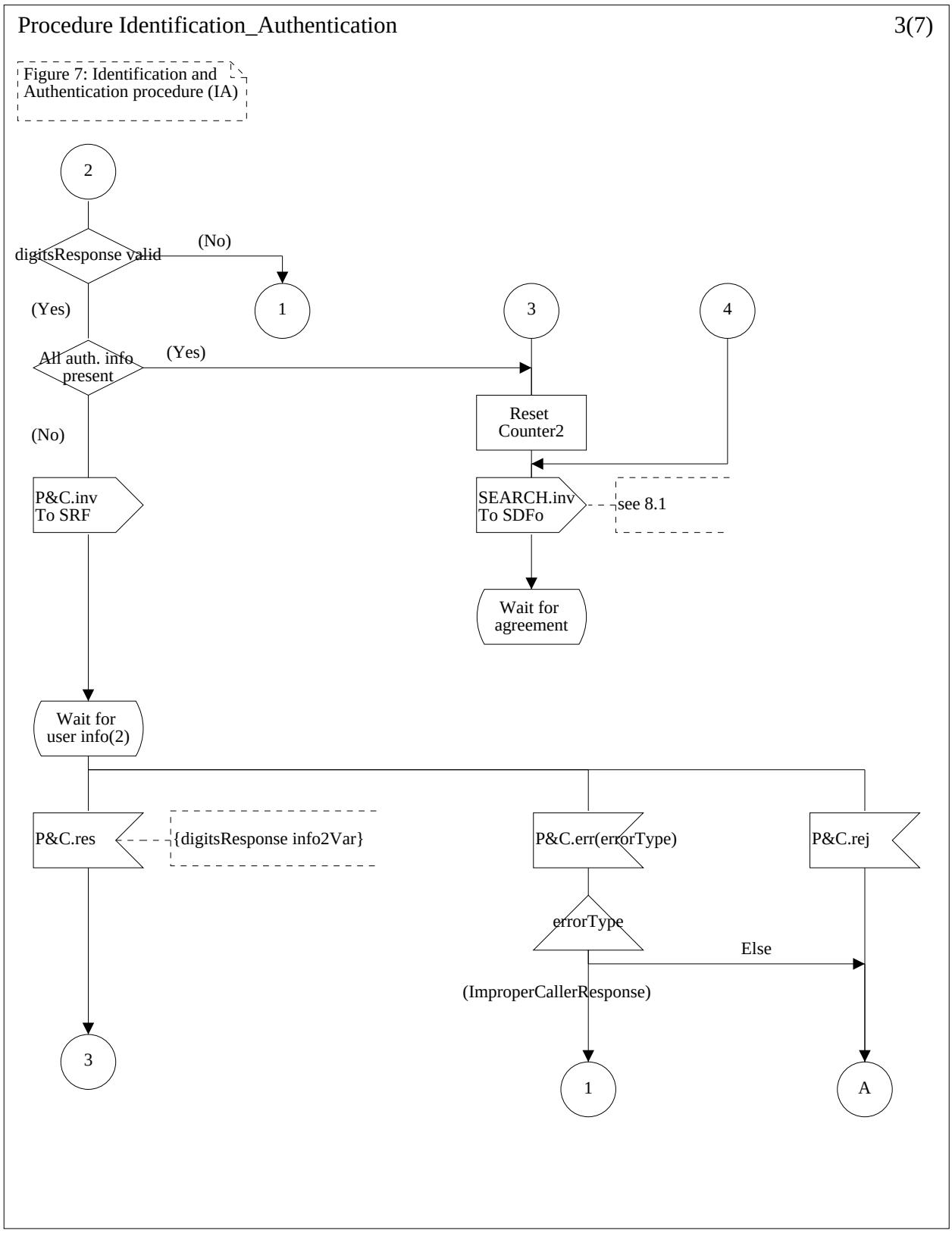


**Figure 7 (sheet 3 of 7): Identification and Authentication procedure (IA)**

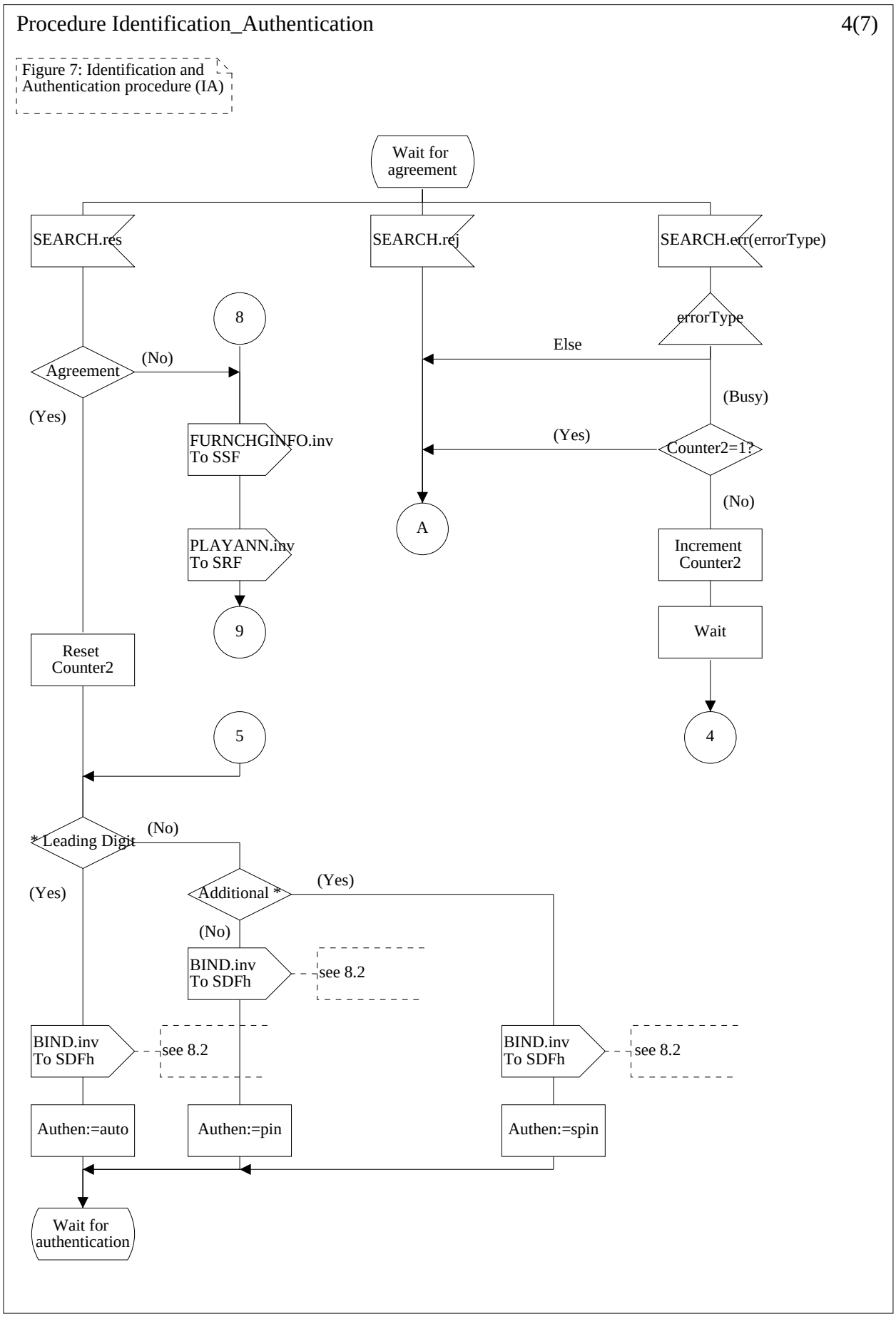


**Figure 7 (sheet 4 of 7): Identification and Authentication procedure (IA)**

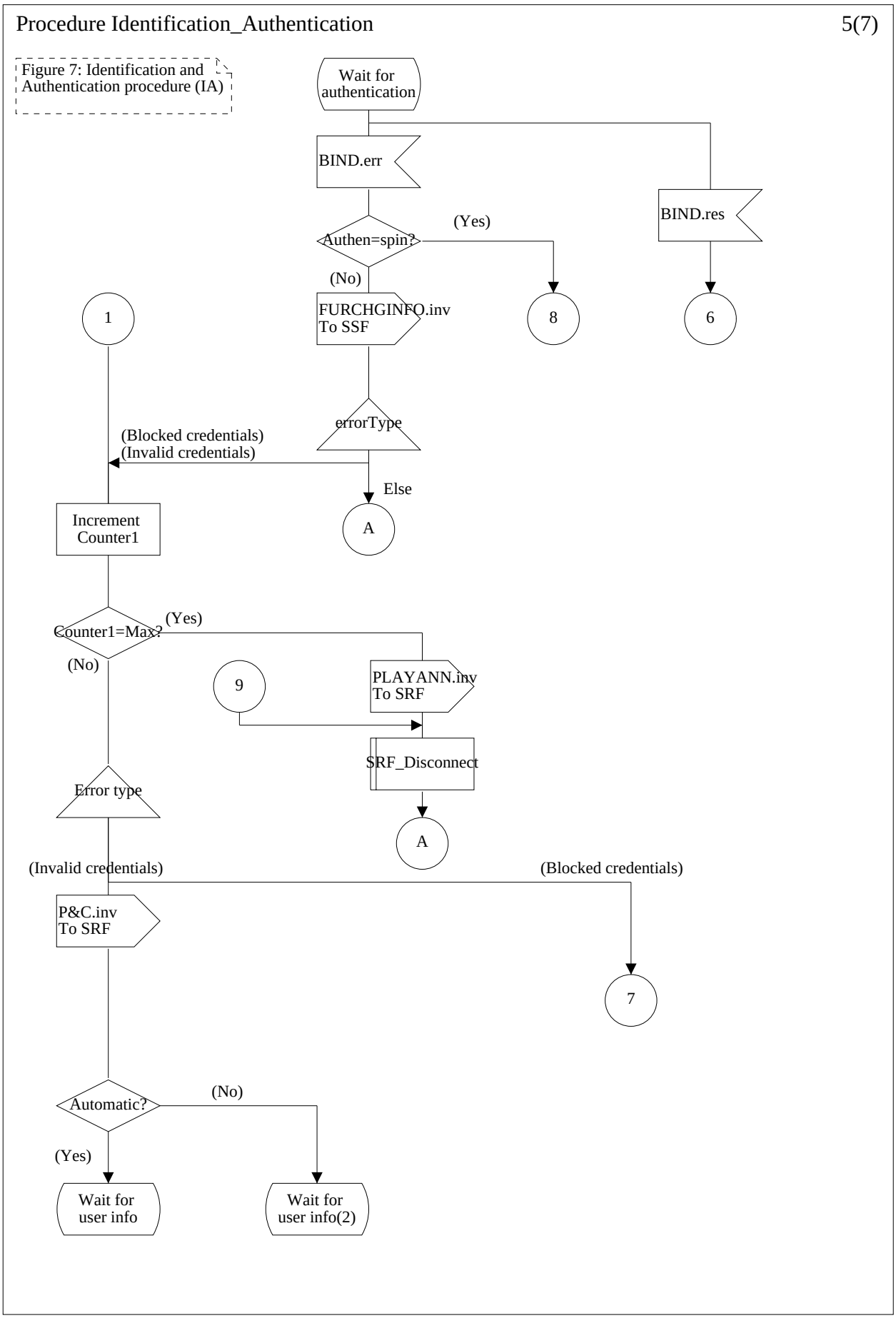


**Figure 7 (sheet 5 of 7): Identification and Authentication procedure (IA)**

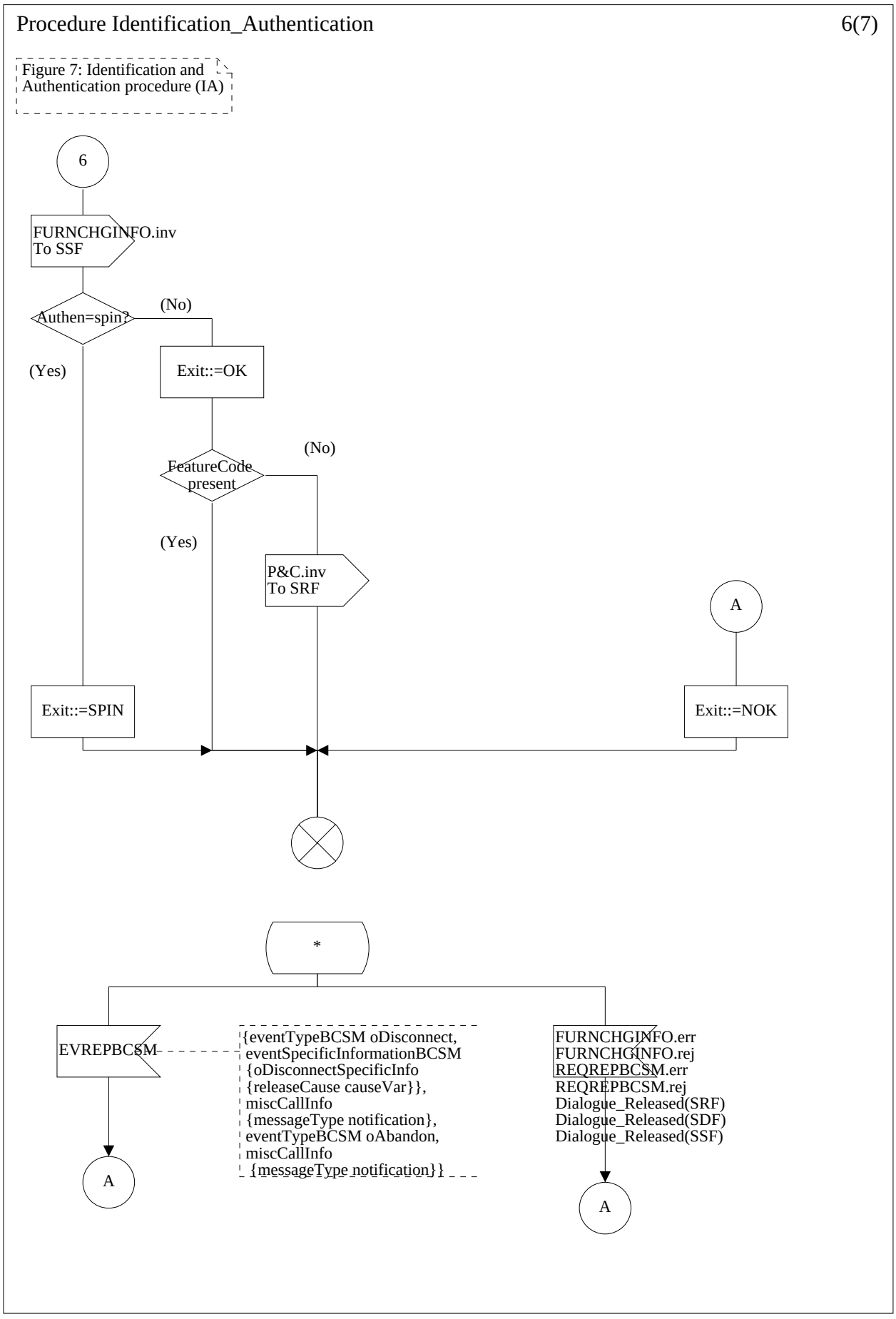


**Figure 7 (sheet 6 of 7): Identification and Authentication procedure (IA)**

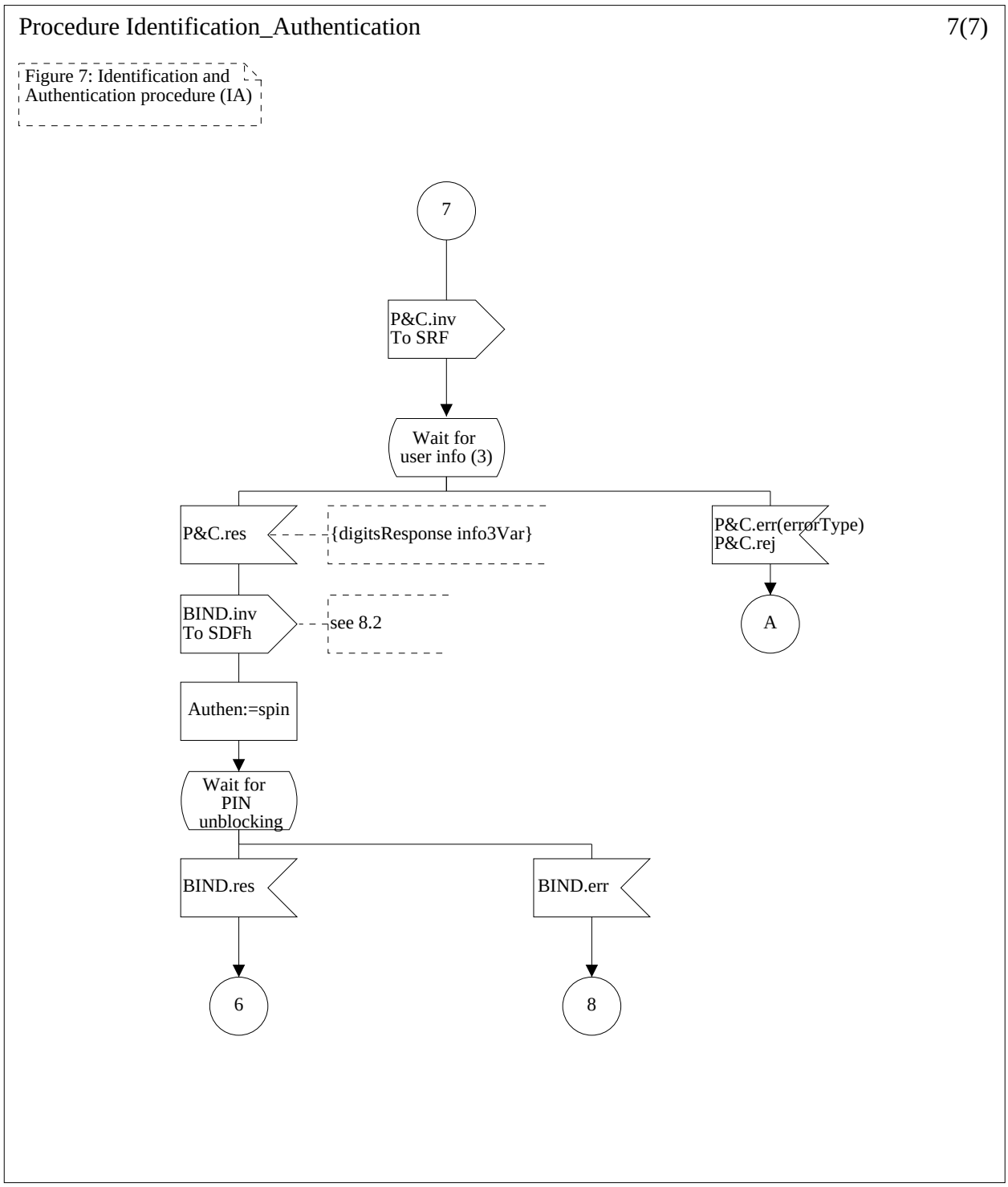


**Figure 7 (sheet 7 of 7): Identification and Authentication procedure (IA)**

### 7.2.2 Feature request identification

#### 7.2.2.1 General

The feature request identification (FRI) procedure takes places:

- after a successful authentication;

- within the follow-on procedure after any other procedure.

It is used to identify a feature request and check if the feature can be supported for the user.

The associated procedure is named Feature Request Identification (FRI). It has two logical outputs. They are identical to the ones for the IA procedure except that the SPIN output does not exist.

Like in the IA procedure, at any stage of the FRI procedure, the user can abandon, he can be disconnected or errors for pending operations can occur. Therefore the same "State *" applies to the FRI procedure.

#### 7.2.2.2 Detailed procedure

Figure 8 is the SDL diagram for this procedure.

**Identification of the service feature**

The SCF starts to check if the feature was already given by the user. If so, no P&C is sent to the user to ask him for the requested feature code. If not, the P&C is sent and the SCF moves to the state "Wait for feature code". It is waiting for the user response to a previous P&C. There are three responses to expect:

- The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user's release. The FRI procedure is terminated and followed by a release procedure.

- An error has occurred: The SCF receives an error indication. What follows is a typical error treatment for a P&C. Only ImproperCallerResponse is considered. It is followed by a procedure that informs the SSF of the failed attempt with a FURNCHGINFO and that allows the user to redial the feature code or that notifies him about a denied access to the UPT service after the maximum number of retries is reached.

- The user has answered the request: The SCF receives a P&C response.

**Service provider agreements**

Once the user information is collected, the user information is used to send a database operation to the SDFo. This operation checks if particular agreements exist between the local service provider and the user's home service provider for the requested feature. The three possible outcomes are:

a) The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user release. The FRI procedure is terminated and followed by a release procedure.

b) An error has occurred: The SCF receives an error indication:

   - If a service error of type Busy occurs, the implementer can resend the database operation to this same entity after a given time;

   - Otherwise the FRI procedure is terminated and the user is released by the network;

c) A response to the database operation has been received: The content of the response is checked. If there is no agreement between the service providers, the SSF is informed of the failure of the FRI procedure and is requested to modify the call record by a FURNCHGINFO. Depending on the number of retries available, the user is informed with a P&C that he can perform another FRI procedure or he is informed with a PLAYANN that he will be released by the network. If there is an agreement between the service providers, the user has access to the requested feature.

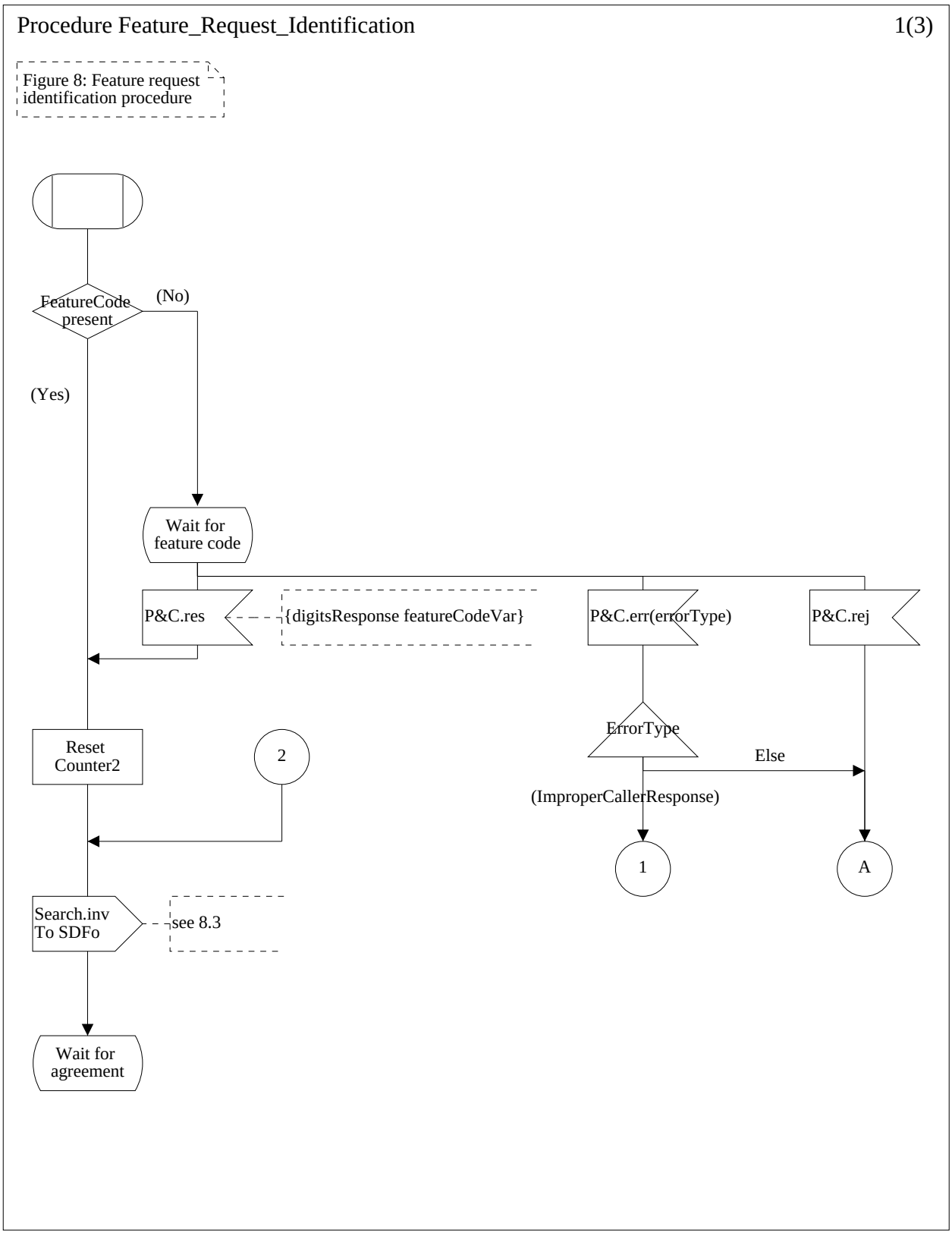


**Figure 8 (sheet 1 of 3): Feature request identification procedure**

Procedure Feature_Request_Identification 2(3)

Figure 8: Feature request identification procedure

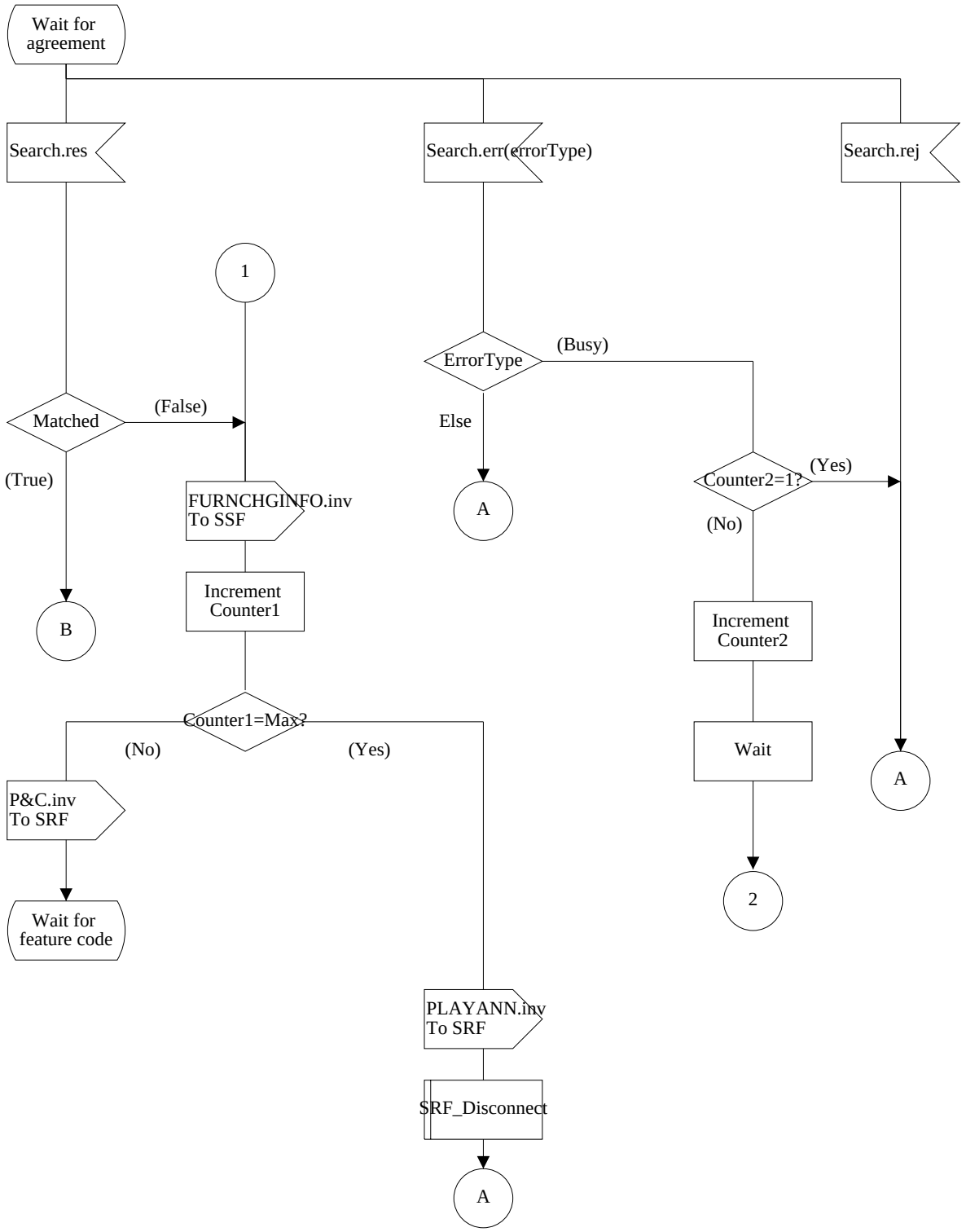


**Figure 8 (sheet 2 of 3): Feature request identification procedure**

Procedure Feature_Request_Identification 3(3)

Figure 8: Feature request identification procedure

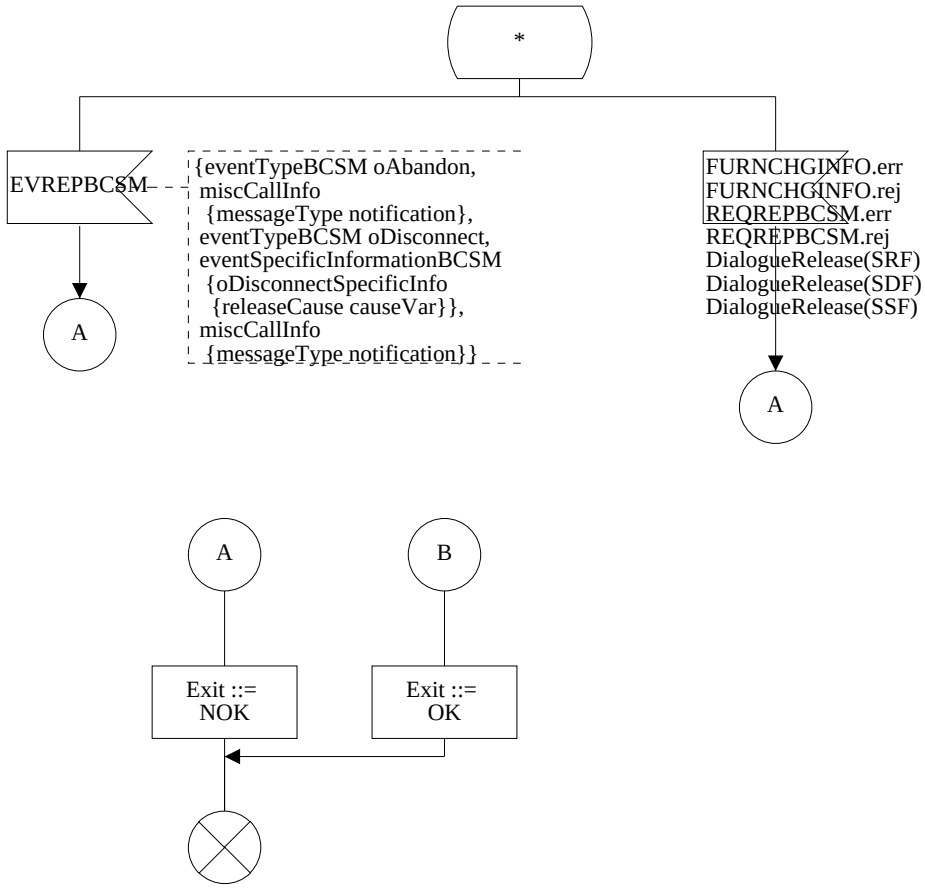


**Figure 8 (sheet 3 of 3): Feature request identification procedure**

### 7.2.3 Release of the calling user

#### 7.2.3.1 General

The release of the calling user takes place at the end of every UPT user request (including one or more feature requests).

The following situations may be encountered:

- release initiated by the user hanging up in any state. The release may also be initiated by the originating network;

- forced release initiated by the IN-node.

Even though the two types of released are triggered by different events, the release procedure is the same in both cases. The release procedure generally includes the release of all the external connections (e.g. lines, circuits) and the release of all engaged IN resources (e.g. intelligent peripheral).

#### 7.2.3.2 Detailed procedure

Figure 9 is the SDL diagram for this procedure.

At the beginning of the procedure, the SCF sends a Call Release to the SSF and closes the dialogues established with the other entities. The procedure is then terminated.

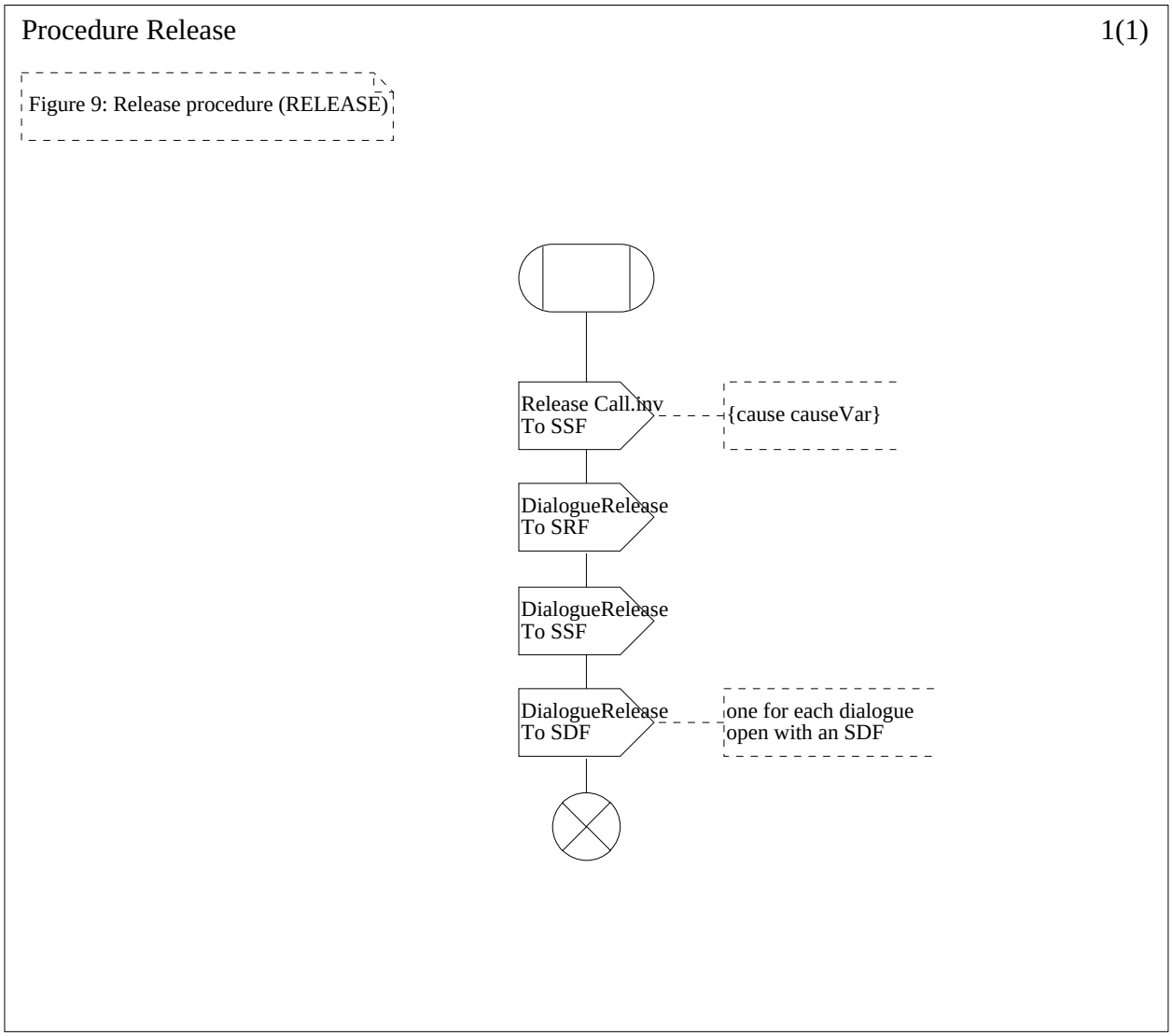


**Figure 9: Release procedure**

### 7.2.4 Connection of an SRF

#### 7.2.4.1 General

The connection of an SRF takes place when there is a need to send a message to a user. It is also used to collect information provided by the user. For example this procedure occurs at the beginning of the IA procedure to send messages to the user and to receive back provided information (like authentication information). It also occurs for the follow-on procedure following an outgoing call procedure. It allows the SCF to interact with the user.

Two technical options are available to connect an SRF. They depend on the type of interface used between the SCF and the SSF or on whether the operations from the SCF are relayed through the SSP (i.e. SSF and SRF integrated or not). Both options are presented in the procedure.

The associated procedure is named SRF connection (SRF_Connect). It has two logical outputs. They are identical to the ones for the FRI procedure. Like in the IA and in the FRI procedures, at any stage of the procedure, the user can abandon.

#### 7.2.4.2 Detailed procedure

Figure 10 is the SDL diagram for this procedure.

The technical choice concerning the relation between the IP and the SSP is left open in the SDL procedure. An attribute internal to the SCF called Op_Relayed contains indications on whether the operations are relayed or not by the SSP. To show this implementation choice, the value of the attribute is checked and the following sequence of the messages depends on the result of the check. However, this choice is not present in a given implementation where only one message sequence is selected. The two possible sequences are:

- Operations relayed: The SCF sends a CONNTORES to the SSF. The message orders the SSF to connect an SRF.

- Operations not relayed: The SCF sends an ETC to the SSF and waits for the confirmation of the connection with an SRF through an ARI message. If the confirmation is received, the procedure normally ends (exit=OK). If the confirmation is not received, this means that a problem arose with the previous ETC operation. Either the operation was rejected and the user should be released by the network or an error was detected in the processing of the operation. The only case of error recovery is for the ETCFailed error. Then the SCF has the possibility of contacting a new SRF (if any available) via a new ETC operation. The other error cases lead to a release (exit = NOK).

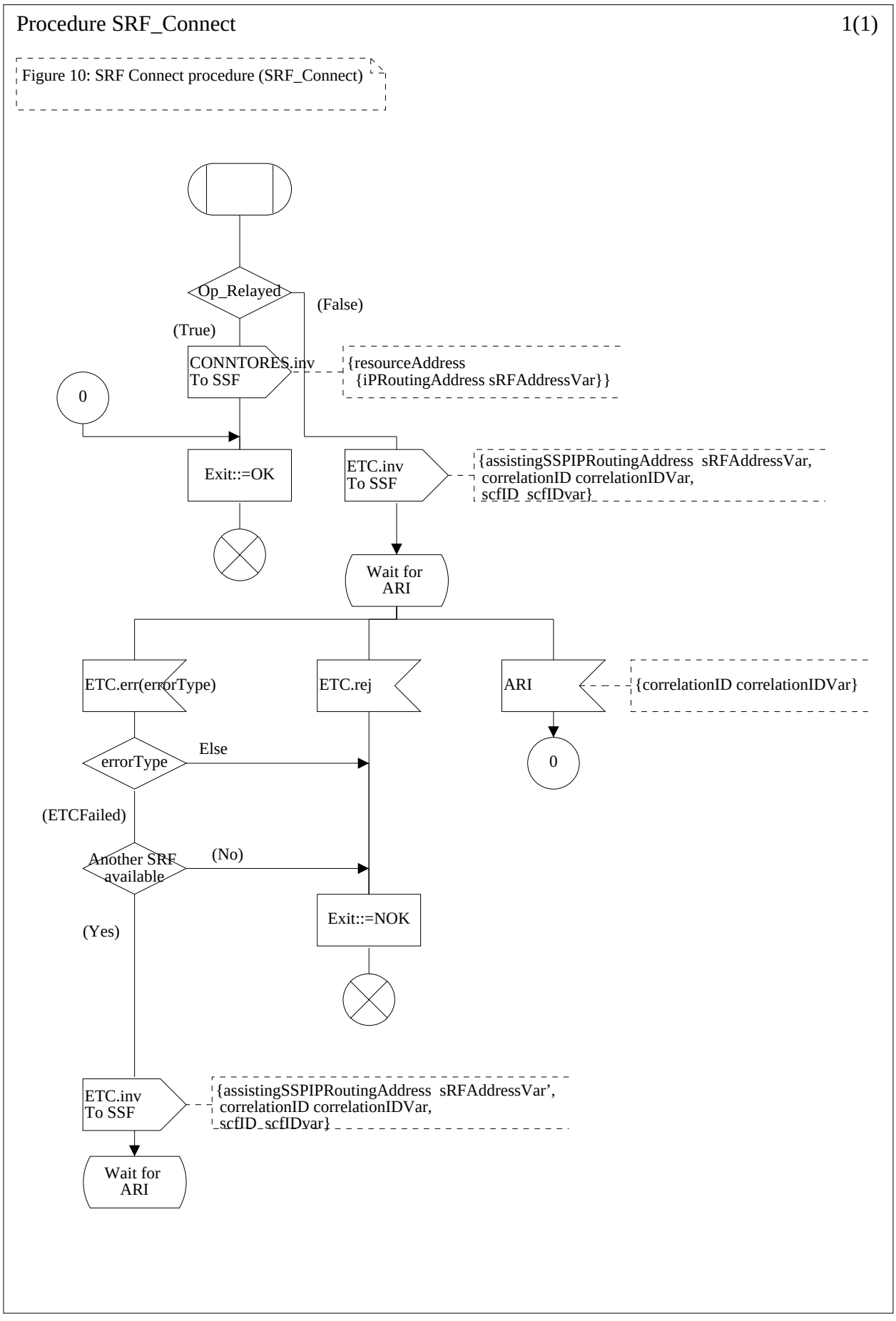


**Figure 10: SRF Connect procedure**

### 7.2.5 Disconnection of an SRF

#### 7.2.5.1 General

The disconnection of an SRF takes place either when a message has been sent to a user announcing him the end of his request or when a call initiated by the user is about to be processed. It is used to release the resources engaged in the dialogue between the SRF and the SCF. This procedure occurs at the end of a user request or when the resources are needed to set-up a call.

Two technical options are available to disconnect an SRF. They depend on the type of interface used between the SCF and the SSF or on whether the operations from the SCF are relayed through the SSP (i.e. SSF and SRF integrated or not). Both options are presented in the procedure.

The associated macro is named SRF disconnection (SRF_Disconnect). It has one logical output.

#### 7.2.5.2 Detailed procedure

Figure 11 is the SDL diagram for this procedure (macro).

Either the disconnection takes place after an announcement has been sent or after the abandon of the B user. In the first case an SRFRPT is expected whereas in the other case the disconnection can be executed right away. This is represented in the choice "direct disc". There are two possible ways to disconnect a SRF either by using the DISCFWDCONN operation or by a backward disconnect from the SRF. The choice between the two alternatives is controlled by the use of the "DisconnectFromIPForbidden" parameter of the PLAYANN and P&C operations. The variable "disconn" is used instead of the full parameter name to have shorter names on the diagram.

Macro SRF_Disconnect 1(1)

Figure 11: SRF Disconnection
Macro SRF_Disconnect

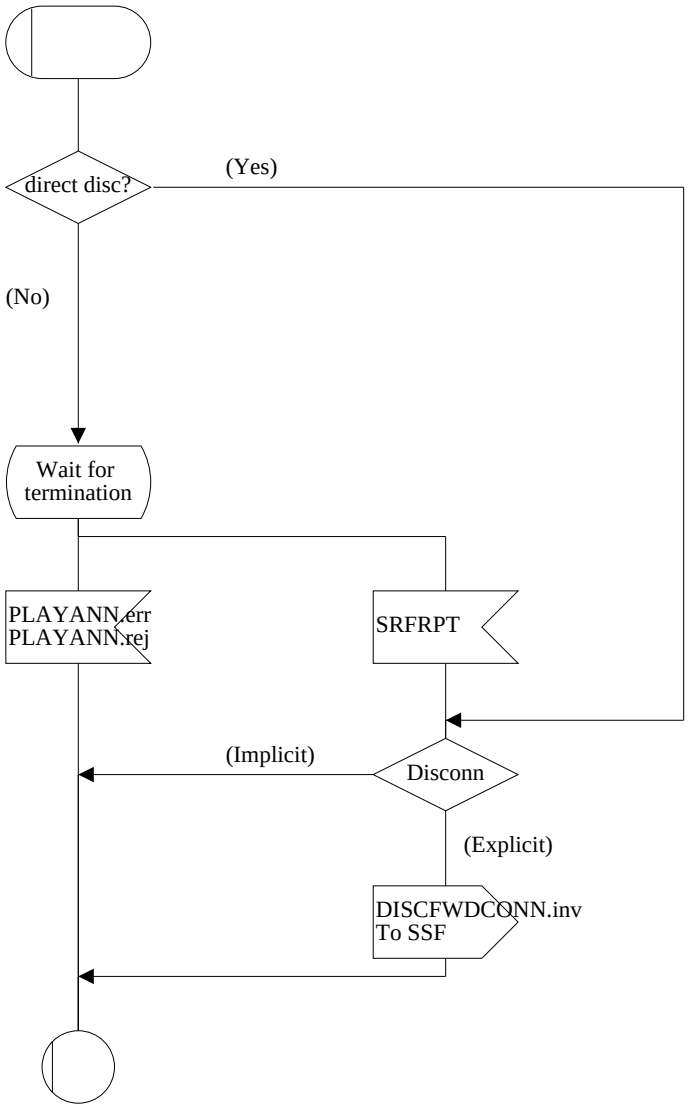


**Figure 11: Macro SRF Disconnect**

## 7.3 Personal Mobility

Personal Mobility Procedures are UPT procedures relating to the personal, or discrete, mobility of the UPT user, used in order to ensure that the UPT user is able to receive or make UPT calls. The personal mobility procedures do not involve actual making or receiving of calls.

### 7.3.1 Registration for incoming calls

#### 7.3.1.1 General

Registration for incoming calls is a feature by which a UPT user registers from the current terminal access for incoming UPT calls to be presented to that terminal access.

When registered, all incoming calls to UPT user will be presented to that terminal access for the duration specified by the UPT user (by a new registration or an explicit deregistration) or until a specified registration limitation. A UPT user's Incall Registration will cancel the previous one of that UPT user. Several UPT users may be registered for incoming calls to the same terminal access simultaneously.

The Identification and Authentication (IA) and Feature Request Identification (FRI) procedures must have been successfully completed before this procedure.

#### 7.3.1.2 Detailed procedure

Figure 12 shows the Registration for Incoming Calls (REG_IN) procedure.

The REG_IN procedure is invoked by the UPT_SLP process when the UPT user answers with the Incall Registration Code during the FRI procedure. In order to simplify the description of the procedure this subclause is structured in subclauses describing different part of the registration.

**Terminal ID available**

When the terminal ID is provided by the user with the feature code the SCF formulates and sends a P&C operation to the SRF in order to ask the user if he wants to register at the terminal specified by the terminal ID and waits for the answer (state "Wait for User ACK"). The processing continues as described in subclause "User's acknowledgement for terminal ID".

**CLI available**

When CLI (Calling Line Identity) is available, the SCF formulates and sends a P&C operation to the SRF and waits for the answer (state "Wait for CLI ACK"). The SRF receives and reacts to the P&C and plays to the user the requested announcement asking the UPT user to indicate whether the registration will be at that terminal or not.

The following events move the SCF out of "Wait for CLI ACK" state:

a) The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user release. The procedure is terminated with Exit = NOK. This is included in the "State *" of the SDL diagrams.

b) An error has occurred for the P&C operation (P&C.err):

   1) If the error is "ImproperCallerResponse", the SCF sends a FURNCHGINFO to request the SSF to create a record for the user request, the number of retries is incremented and, if not exceeded, the prompt is repeated to the user that is allowed to make another attempt. If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded";

   2) Else for any other error the procedure is terminated with Exit=NOK.

c) The P&C operation is rejected (P&C.rej): The procedure is terminated with Exit = NOK.

d) The user answers "yes" to the prompt: The processing continues as described in subclause "UPT user's permission screening".

e) The user answers "no" to the prompt: The processing continues as described in the subclause "CLI not available".

f) The user answers "esc" to the prompt: The SCF sends a FURNCHGINFO to request the SSF to create a record for the user request, the number of retries is incremented and, if not exceeded, the SCF formulates and sends a P&C operation indicating that the user's request has been cancelled and allowing him to return to the feature request identification (FRI) (follow-on). If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded".

**CLI not available**

If CLI is not available or the user's response to register on the current terminal is "no", the desired terminal identity is requested. The SCF formulates and sends a P&C operation to the SRF and waits for the answer (state "Wait for Terminal ID"). The SRF receives and reacts to the P&C and plays to the user the requested announcement to appeal for the terminal identity on which the user wants to register.

The following events move the SCF out of "Wait for Terminal ID" state:

a) P&C operation is successful: The SRF echoes the received terminal identity to the user and sends the dialled digits to the SCF. The SCF formulates and sends the P&C operation in order to prompt the user to confirm or cancel the input data (state "Wait for User ACK"). This is described in subclause "User's acknowledgement for terminal ID".

b) An error has occurred for the P&C operation (P&C.err):

   1) If the error is "ImproperCallerResponse", the SCF sends a FURNCHGINFO to request the SSF to create a record for the user request, the number of retries is incremented and, if not exceeded, the user is requested to enter valid information with a P&C operation. If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded";

   2) Else for any other error the procedure is terminated with Exit=NOK.

c) The P&C operation is rejected (P&C.rej): The procedure is terminated with Exit = NOK.

**Maximum number of retries exceeded**

If the maximum number of retries is exceeded a PLAYANN is sent to SRF in order to inform the user to hang up, the procedure is terminated with Exit = NOK and the calling user is released by the Release procedure.

**User's acknowledgement for terminal ID**

If the user confirms the echoed terminal ID the processing continues as described in subclause "UPT user's permission screening".

If the user dials "esc" the same treatment described in subclause "CLI available" is applied.

If the user does not confirm the echoed terminal ID the SCF sends a FURNCHGINFO to request the SSF to create a record for the user request and the number of retries counter is incremented. If the maximum number of retries is not exceeded, a new attempt is allowed. The SCF formulates and sends a P&C to the SRF in order to prompt the user to retry the registration procedure and collect again the registration input data and returns to the "Wait for Terminal ID" state. If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded".

**UPT user's permission screening**

The SCF formulates and sends a Search operation to the local SDF to verify the UPT user's permission to register on a specified network access and wait for the SDFo answer (state "Wait for Screen results").

The following events move the SCF out of "Wait for screen results" state:

a) An error has occurred for the Search operation (SEARCH.err):

   1) If the error is "Busy" and it is the first attempt (Counter2 = 0) the Search operation is sent again to the SDF after a time-out (timer) and the "UPT user's permission screening" subclause is repeated.

   2) If the error is "Insufficient Access Rights" the procedure is terminated as described in subclause "Unsuccessful registration".

   3) For any other error or after a second "Busy" error the procedure is terminated with Exit = NOK.

b) The Search operation is rejected (SEARCH.rej) the procedure is terminated with Exit = NOK;

c) A response to the Search operation has been received. The Counter2 is reset and the ResultArg received is checked:

   1) If the registration is allowed the processing continues as described in subclause "Collect limitation information".

   2) If the registration for incoming calls is not allowed, the SCF sends a FURNCHGINFO to request the SSF to create a record for the user request and the number of retries counter is incremented. If the maximum number of retries is not exceeded, a new attempt is allowed. The SCF formulates and sends a P&C to the SRF in order to prompt the user to retry the registration procedure and collect again the registration input data and returns to the "Wait for Terminal ID" state. If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded".

**Unsuccessful registration**

The SCF formulates and sends FURNCHGINFO to request the SSF to send a record for all user requests and the number of retries counter is incremented. If the maximum number of retries is not exceeded the SCF formulates and sends a P&C operation in order to inform the user that the request can not be handled and to terminate or request another procedure (this is done by the FRI procedure). If the maximum number is exceeded the processing continues as described in subclause "Maximum number of retries exceeded".

**Collect limitation Information**

If the registration is allowed, the SCF formulates and sends P&C operation to SRF to request the user to provide time limitation information and waits for the answer (state "Wait for limitation sequence"). The following events move the SCF out of "Wait for limitation sequence" state:

a) P&C operation is successful: The SRF echoes the received limitation sequence to the user and sends the dialled digits to the SCF. The SCF formulates and sends the P&C operation to SRF in order to prompt the user to confirm or cancel the input data (state "Wait for limitation ACK."). This is described in subclause "User's acknowledgement for limitation sequence".

b) An error has occurred for the P&C operation (P&C.err):

   1) If the error is "ImproperCallerResponse", the SCF sends a FURNCHGINFO to request the SSF to create a record for the user request, the number of retries is incremented and, if not exceeded, the user is requested to enter valid information with a P&C operation. If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded";

2) Else for any other error the procedure is terminated with Exit=NOK.

c) The P&C operation is rejected (P&C.rej): The procedure is terminated with Exit = NOK.

**User's acknowledgement for limitation sequence**

If the user confirms the echoed limitation sequence the processing continues as described in subclause "Collect limitation Information".

If the user dials "esc" the same treatment described in subclause "CLI available" is applied.

If the user does not confirm the echoed terminal ID the SCF sends a FURNCHGINFO to request the SSF to create a record for the user request and the number of retries counter is incremented. If the maximum number of retries is not exceeded, a new attempt is allowed. The SCF formulates and sends a P&C operation to the SRF in order to prompt the user to retry the registration procedure and collect again the limitation sequence and returns to the "Wait for limitation sequence" state. If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded".

**Update of UPT current location**

When the registration for incoming calls is allowed, the SCF formulates and sends a ModifyEntry operation to the SDF home to check if the given number does not violate input restrictions and if so, to update the UPT user current location with the given information. The SCF formulates and sends a P&C operation in order to inform the user that his request has been processed. The SCF waits for the SDFh answer (state "Wait for Update Confirmation"). The following events move the SDF out of this state:

a) The ModifyEntry operation is successful: The registration is accepted. The Counter2 is reset and the SCF formulates and sends FURNCHGINFO to the SSF. Afterwards the SCF sends a P&C operation in order to prompt the UPT user that the registration has been successfully executed and ask to terminate or request another procedure (this is done by the FRI procedure).

b) An error has occurred for the ModifyEntry operation (MODIFY.err):

1) If the error is "Insufficient Access Rights" the registration is denied and the processing continues as described in subclause "Unsuccessful registration".

2) If the error is "Busy" and it is the first attempt (Counter2 = 0) the ModifyEntry operation is sent again to the SDF after a time-out (timer) and the "Update of UPT current location" subclause is repeated.

3) For any other error or after a second "Busy" error the procedure is terminated with Exit = NOK.

c) The ModifyEntry operation is rejected (MODIFY.rej) the procedure is terminated with Exit = NOK.

Procedure REG_IN 1(8)

Figure 12: SCF Registration for Imcoming Calls Procedure Procedure REG_IN

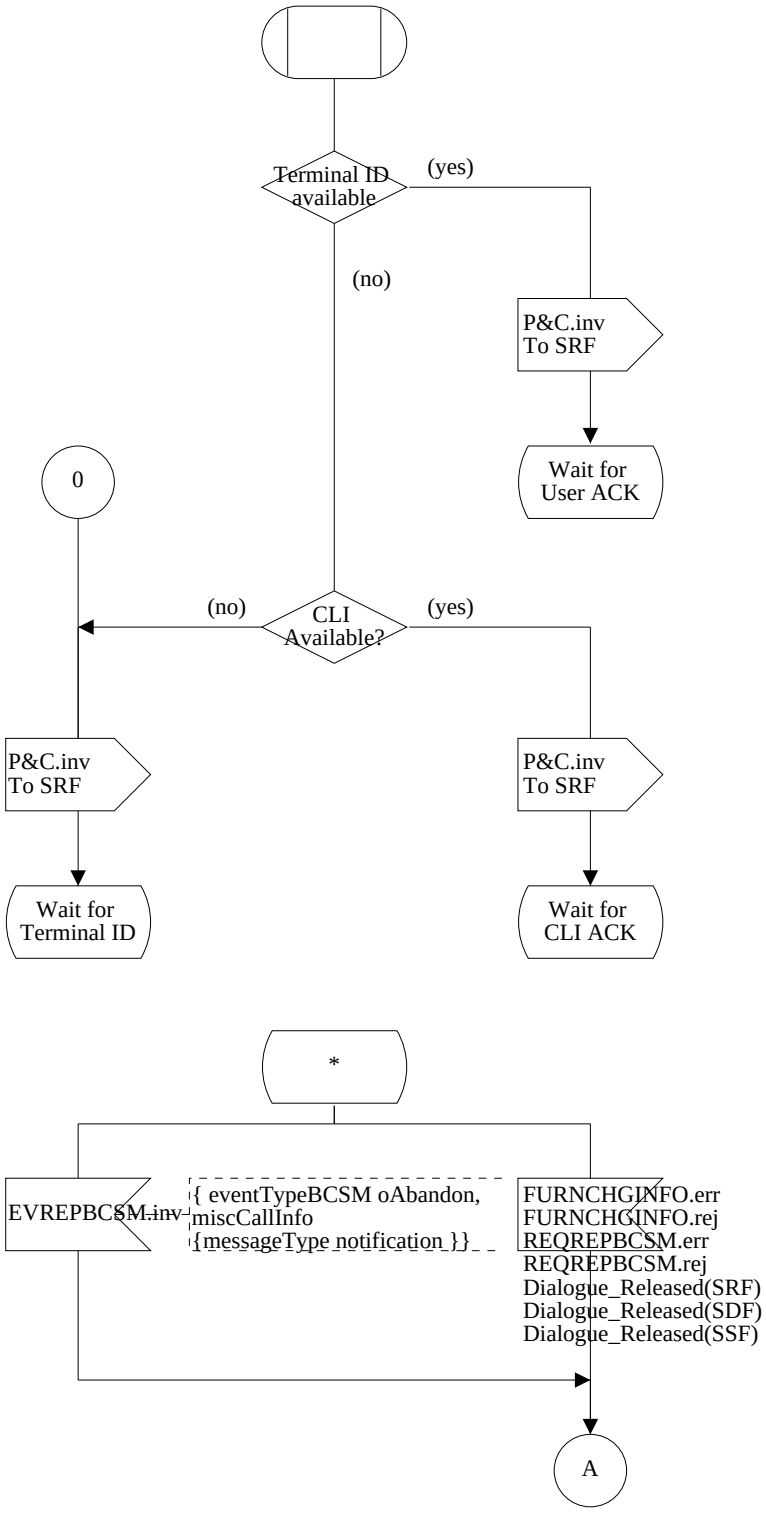


**Figure 12 (sheet 1 of 8): SCF Registration for Incoming Calls procedure**

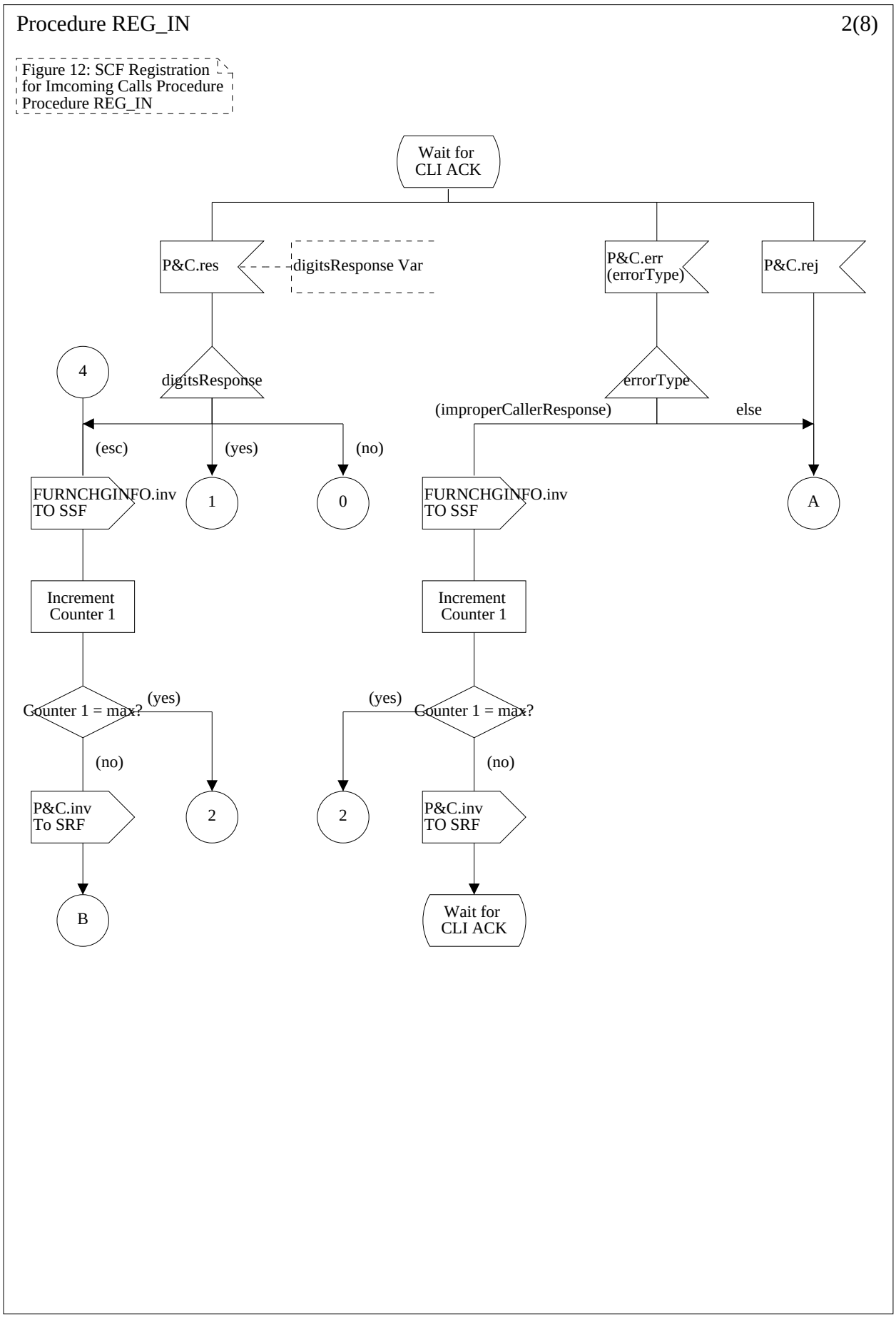


**Figure 12 (sheet 2 of 8): SCF Registration for Incoming Calls procedure**

Procedure REG_IN 3(8)

Figure 12: SCF Registration for Imcoming Calls Procedure Procedure REG_IN

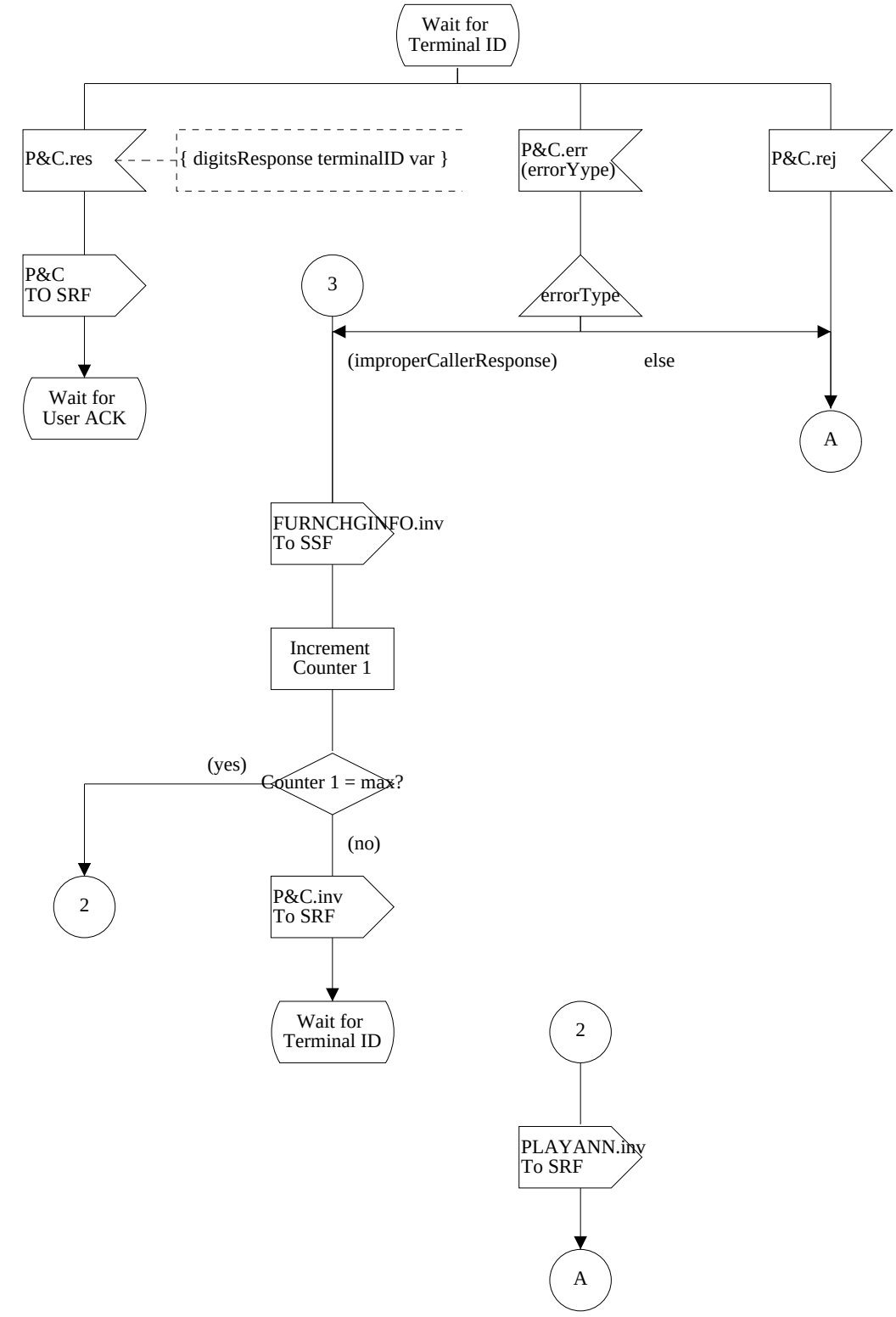


**Figure 12 (sheet 3 of 8): SCF Registration for Incoming Calls procedure**

Procedure REG_IN 4(8)

Figure 12: SCF Registration for Imcoming Calls Procedure Procedure REG_IN

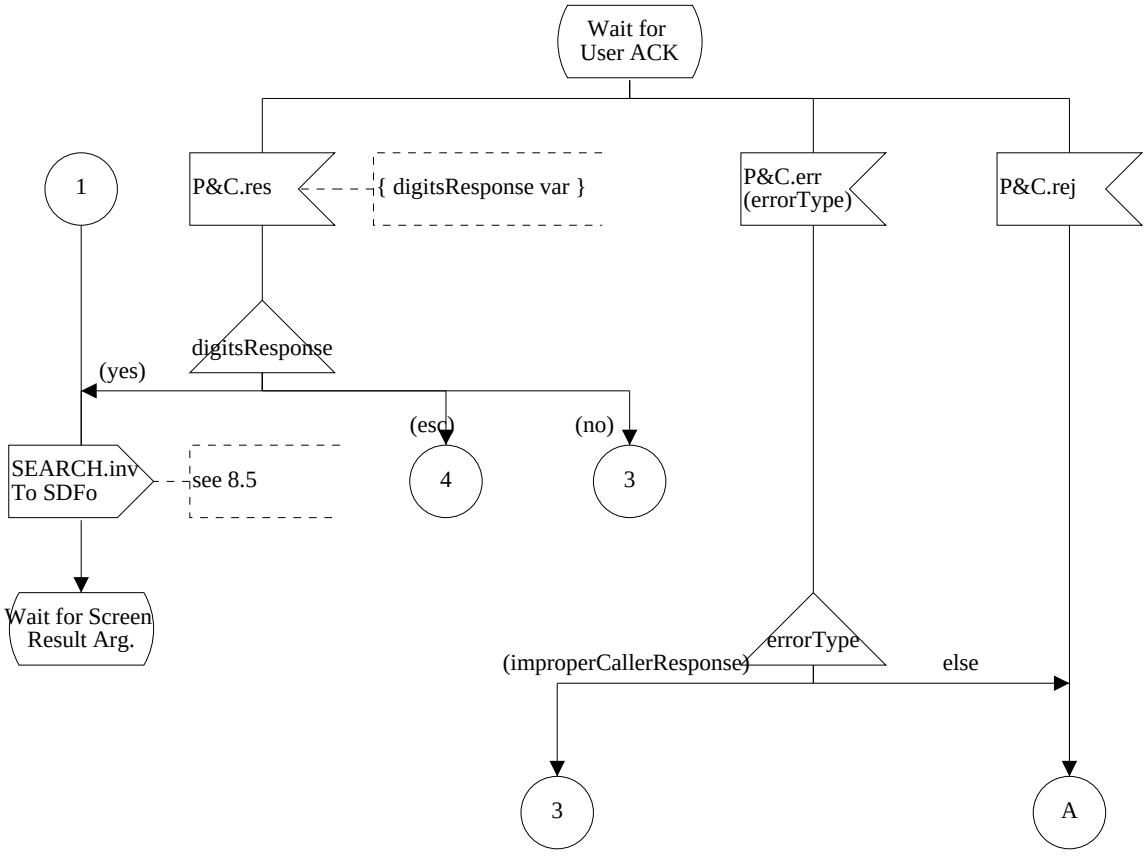


**Figure 12 (sheet 4 of 8): SCF Registration for Incoming Calls procedure**

Procedure REG_IN 5(8)

Figure 12: SCF Registration for Imcoming Calls Procedure Procedure REG_IN

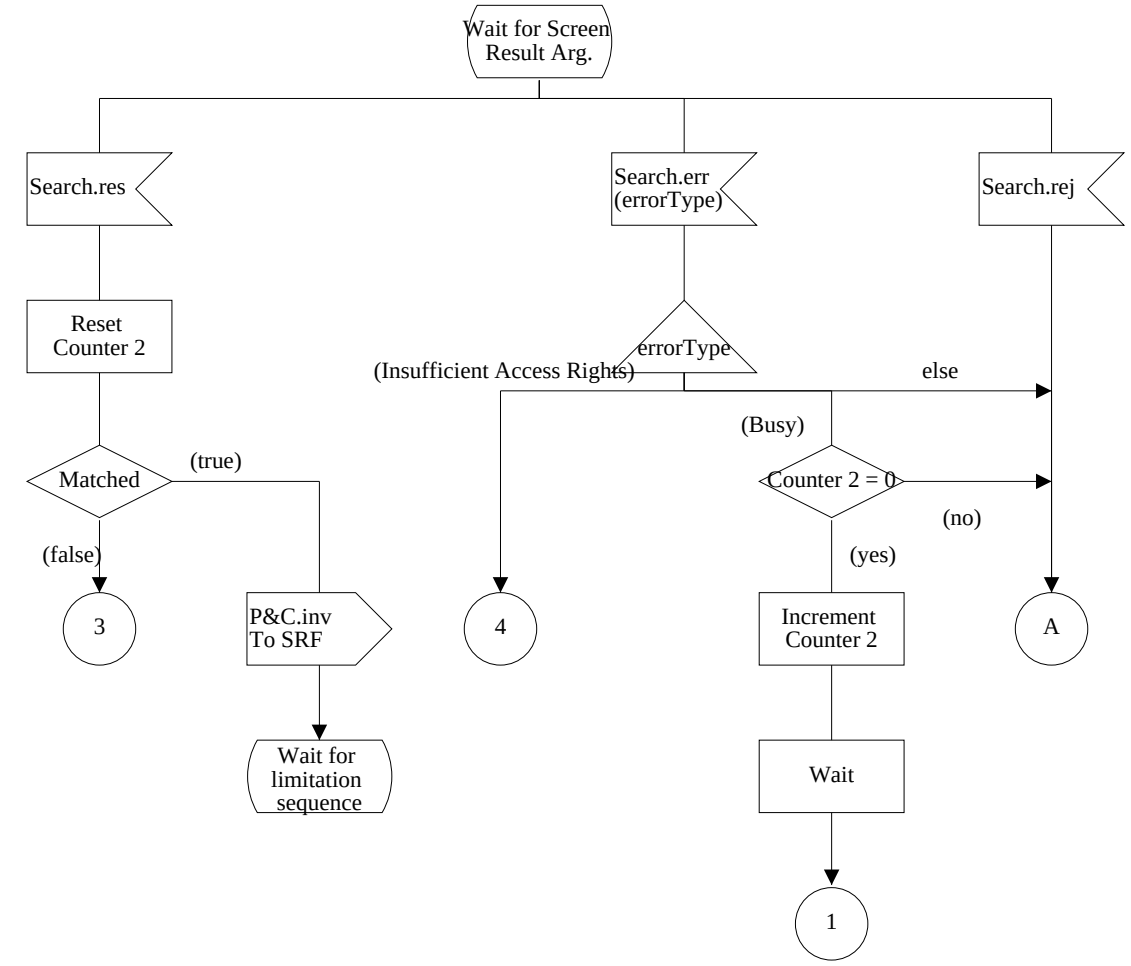


**Figure 12 (sheet 5 of 8): SCF Registration for Incoming Calls procedure**

Procedure REG_IN 6(8)

Figure 12: SCF Registration for Imcoming Calls Procedure Procedure REG_IN

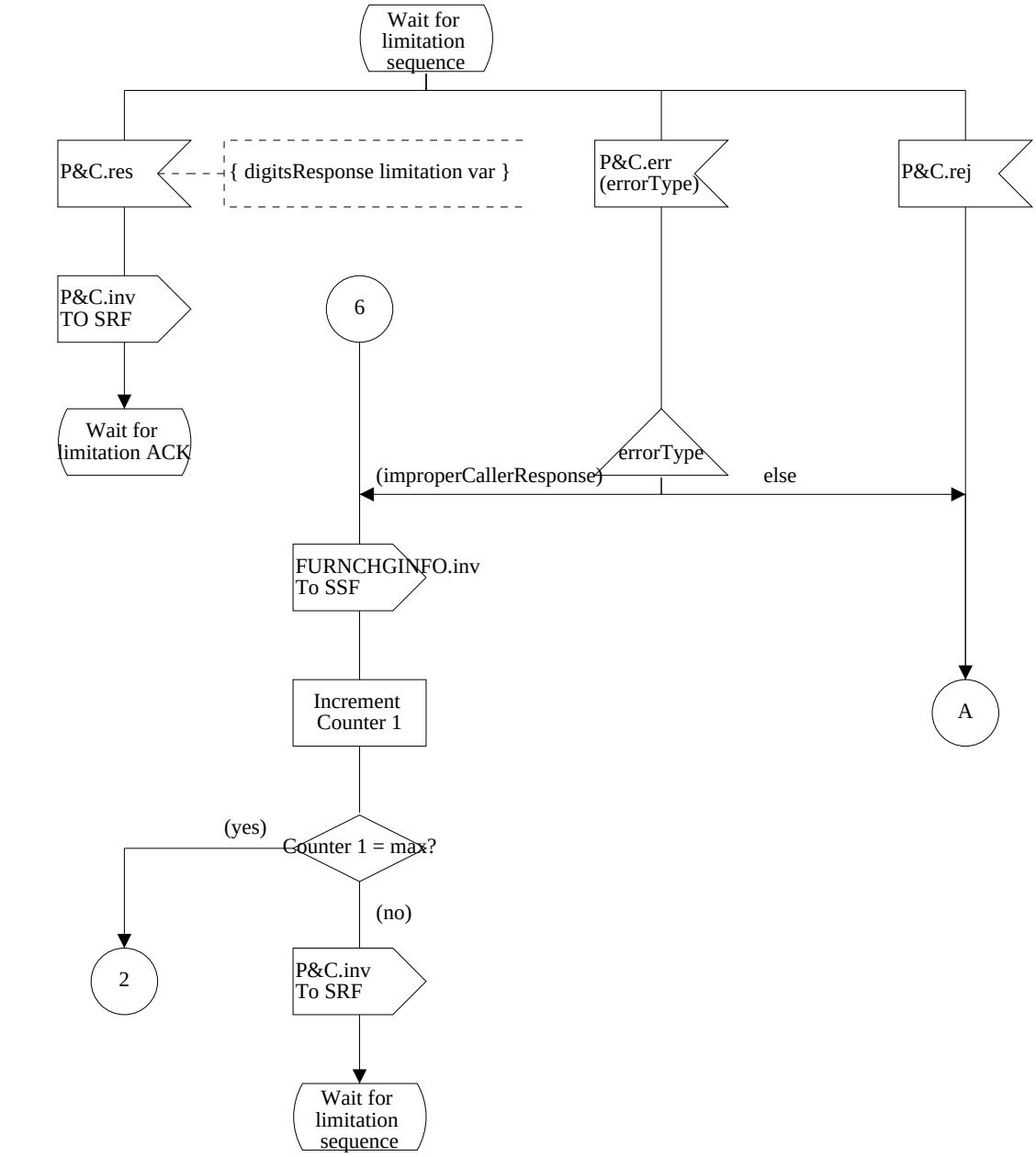


**Figure 12 (sheet 6 of 8): SCF Registration for Incoming Calls procedure**

Procedure REG_IN 7(8)

Figure 12: SCF Registration for Imcoming Calls Procedure Procedure REG_IN

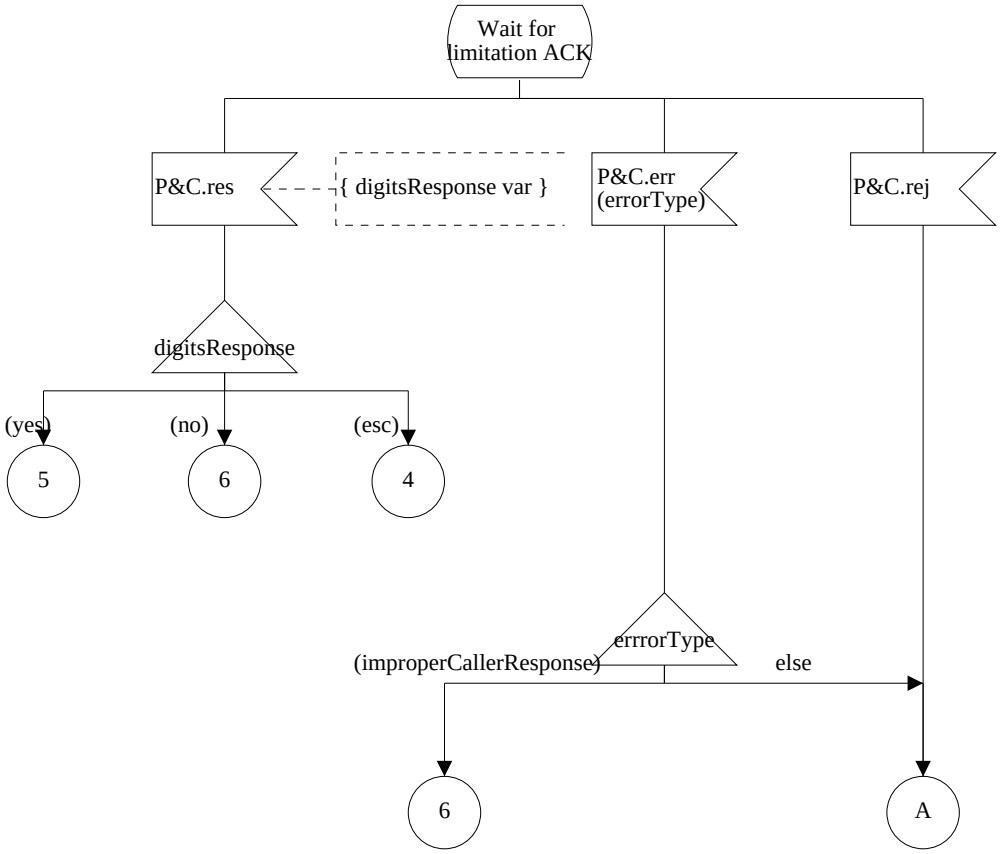


**Figure 12 (sheet 7 of 8): SCF Registration for Incoming Calls procedure**

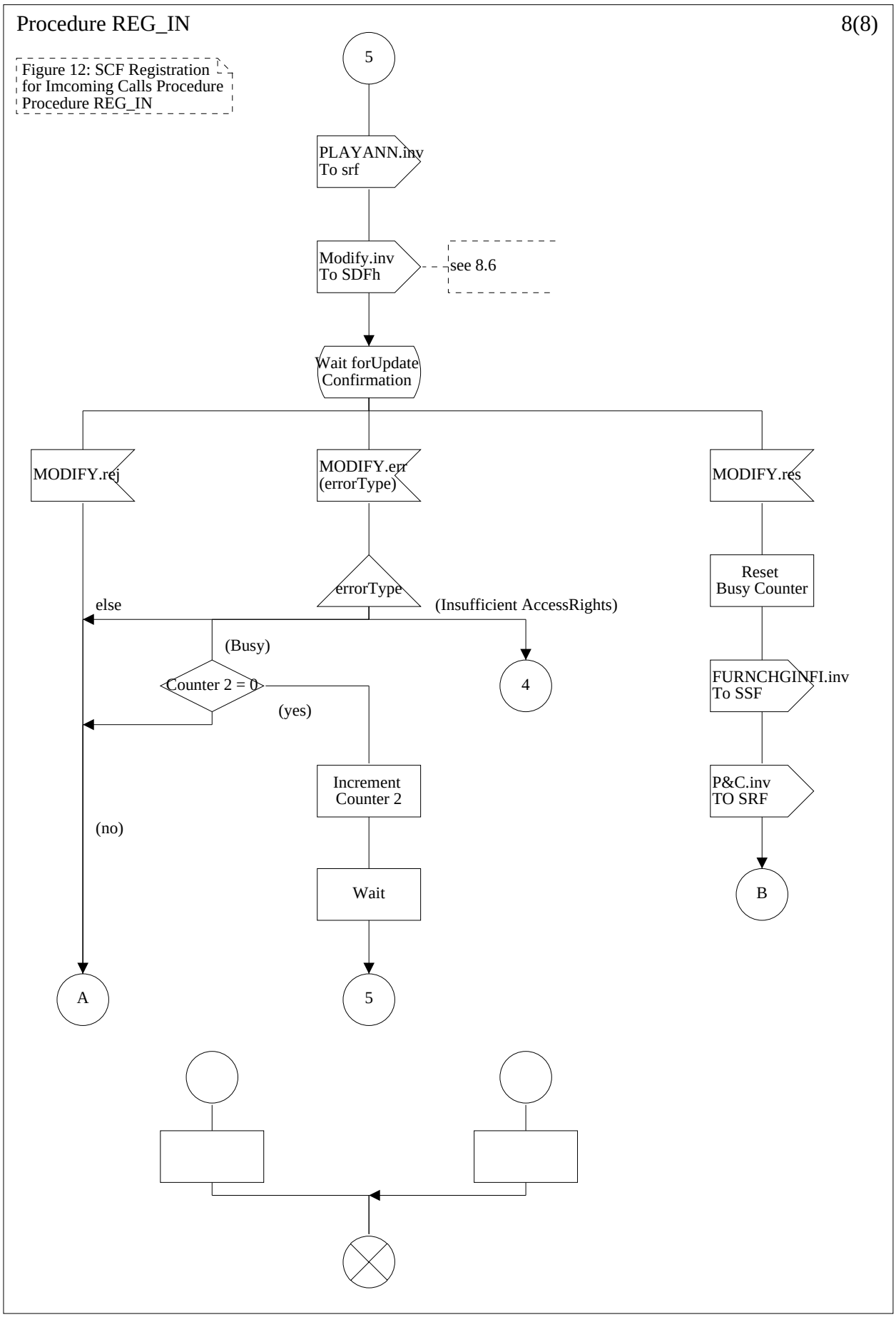


**Figure 12 (sheet 8 of 8): SCF Registration for Incoming Calls procedure**

### 7.3.2 Deregistration for Incoming Calls

#### 7.3.2.1 General

The Deregistration for Incoming Calls procedure is used when the UPT user explicitly deregisters.

The IA and FRI procedures must have been successfully completed before this procedure.

#### 7.3.2.2 Detailed procedure

Figure 13 shows the Deregistration for Incoming Calls (DEREG_IN) procedure

The DEREG_IN procedure is invoked by the UPT_SLP process when the UPT user answers with the Incall Deregistration Code during the FRI procedure.

**Retrieve of registration information**

As the retrieval of registration information is optional two cases may occur:

**Option 1:** If the data retrieval is necessary for obtaining the current registration information the SCF formulates and sends a Search operation to the home SDF in order to retrieve the terminal address where the user is currently registered for incoming calls. After that it waits for the SDF answer (state "Wait for Retrieve res.").

**Option 2:** Otherwise the procedure continues as described in subclause "User Ack. For Deregistration"

The following events move the SCF out of "Wait for Retrieve res." state:

a) an error has occurred for the Search operation (SEARCH.err):

   1) If the error is "Busy" and it is the first attempt (Counter2 = 0) the ModifyEntry operation is sent again to the SDF after a time-out (timer) and the "Retrieve of registration information" subclause is repeated.

   2) If the error is "Insufficient Access Rights" the procedure is terminated as described in subclause "Unsuccessful deregistration".

   3) For any other error the procedure is terminated with Exit = NOK.

b) The Search operation is rejected (SEARCH.rej) the procedure is terminated with Exit = NOK.

c) A response to the Search operation has been received. If a previous registration was done by the user (terminal ID available) the procedure continues as described in the subclause "User Ack. for Deregistration". Otherwise the procedure is terminated as described in subclause "Unsuccessful deregistration".

**User Ack. for Deregistration**

The SCF formulates and sends a P&C operation to SRF. The SRF receives and reacts to P&C in two different ways according to options described below:

**Option 1:** If the data retrieval of registration information has occurred the SRF plays to the user to appeal for his acknowledgement of the deregistration from the terminal where the user is currently registered for incoming calls and waits for the answer (state "Wait for user dereg. Ack.").

**Option 2:** Otherwise the SRF plays to the user for his acknowledgement of the deregistration without expliciting the terminal identity from where the user is currently registered for incoming calls and waits for the answer (state "Wait for user dereg. Ack.).

The following events move the SCF out of "Wait for user dereg. ack." state:

a) The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user release. The procedure is terminated with Exit = NOK. This is included in the "State *" of the SDL diagrams.

b) An error has occurred for the P&C operation (P&C.err):

   1) If the error is "ImproperCallerResponse", the SCF sends a FURNCHGINFO to request the SSF to create a record for the user request, the number of retries is incremented and, if not exceeded, the prompt is repeated to the user that is allowed to make another attempt. If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded";

   2) Else for any other error the procedure is terminated with Exit=NOK.

c) The P&C operation is rejected (P&C.rej): The procedure is terminated with Exit = NOK.

d) The user answers "yes" to the prompt: The processing continues as described in subclause "Set to default registration address".

e) The user answers "no" or "esc" to the prompt the procedure is terminated as described in subclause "Unsuccessful deregistration".

**Set to default registration address**

If the user accepts the deregistration the SCF formulates and sends a P&C operation to SRF in order to inform the user that his request has been processed. The SCF formulates and sends an ModifyEntry operation to the SDFh to set the UPT user current location to the default value. The SCF waits for the SDFh answer (state "Wait for Update Confirmation").

The following events move the SDF out of this state:

a) The ModifyEntry operation is successful: The deregistration is accepted. The SCF formulates and sends FURNCHGINFO to the SSF. Afterwards the SCF sends a P&C operation in order to prompt the UPT user that the deregistration has been successfully executed and ask to terminate or request another procedure (this is done by the FRI procedure).

b) An error has occurred for the ModifyEntry operation (MODIFY.err):

   1) If the error is "Insufficient Access Rights" the deregistration is denied and the processing continues as described in subclause "Denied Update".

   2) If the error is "Busy" and it is the first attempt (Counter2 = 0) the ModifyEntry operation is sent again to the SDF after a time-out (timer) and the "Set to default registration address" subclause is repeated.

   3) For any other error or after a second "Busy" error the procedure is terminated with Exit = NOK.

c) The ModifyEntry operation is rejected (MODIFY.rej) the procedure is terminated with Exit = NOK.

**Denied update**

The SCF formulates and sends FURNCHGINFO to request the SSF to send a record for all user requests and the number of retries counter is incremented. If the maximum number of retries is not exceeded the SCF formulates and send a P&C operation in order to inform the user that the request can not be handled and to terminate or request another procedure (this is done by the FRI procedure). If the maximum number is exceeded the processing continues as described in subclause "Maximum number of retries exceeded".

**Unsuccessful deregistration**

The SCF sends a FURNCHGINFO to request the SSF to create a record for the user request, the number of retries is incremented and, if not exceeded, the SCF formulates and sends a P&C operation allowing the user to return to the feature request identification (FRI) (follow-on). If the number of retries is exceeded the processing continues as described in subclause "Maximum number of retries exceeded".

**Maximum number of retries exceeded**

If the maximum number of retries is exceeded a PLAYANN operation is played to the user for asking him to hang up; the procedure is terminated with Exit = NOK and the calling user is released by the Release procedure.

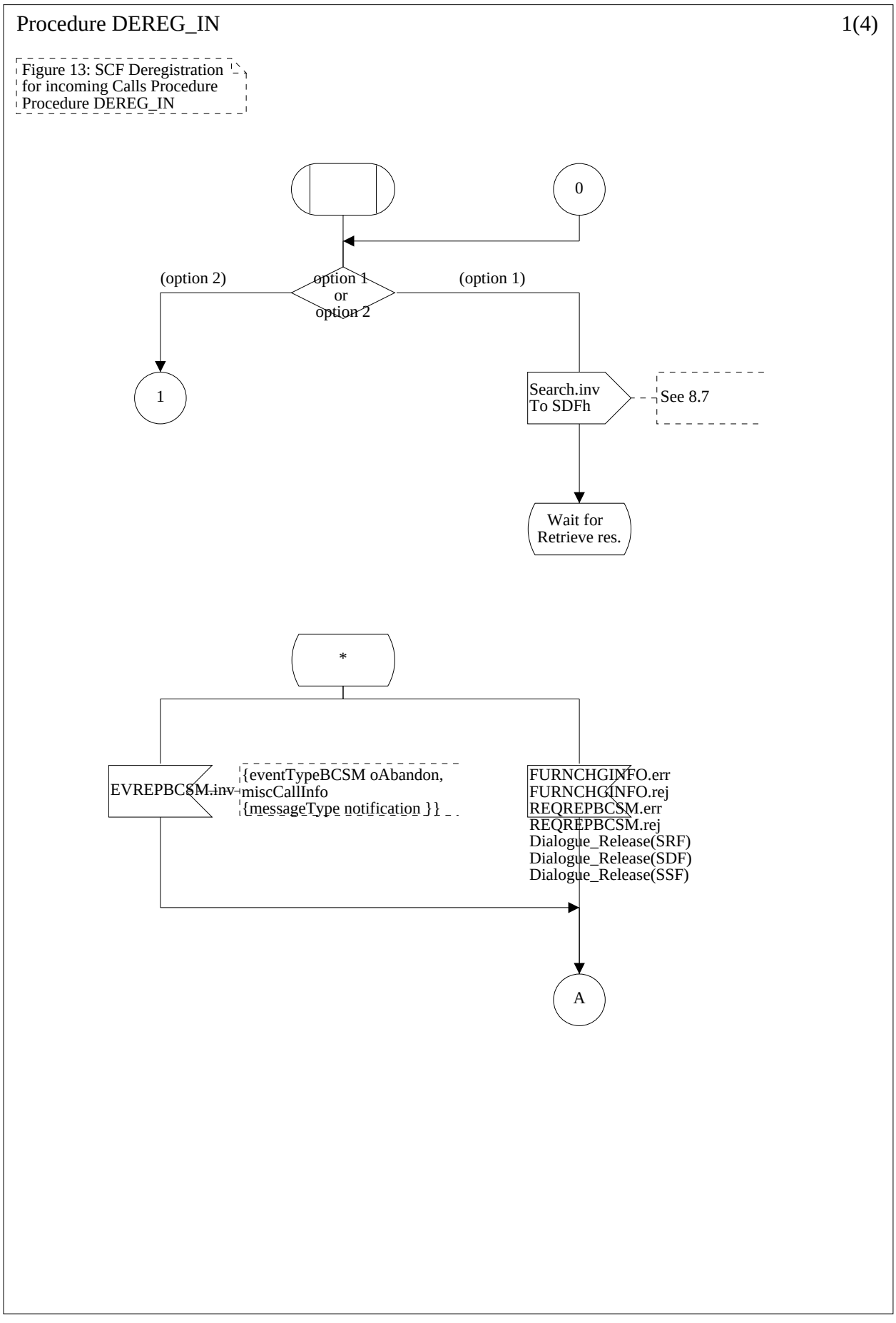


**Figure 13 (sheet 1 of 4): SCF Deregistration for incoming Calls procedure**

Procedure DEREG_IN 2(4)

Figure 13: SCF Deregistration for incoming Calls Procedure Procedure DEREG_IN

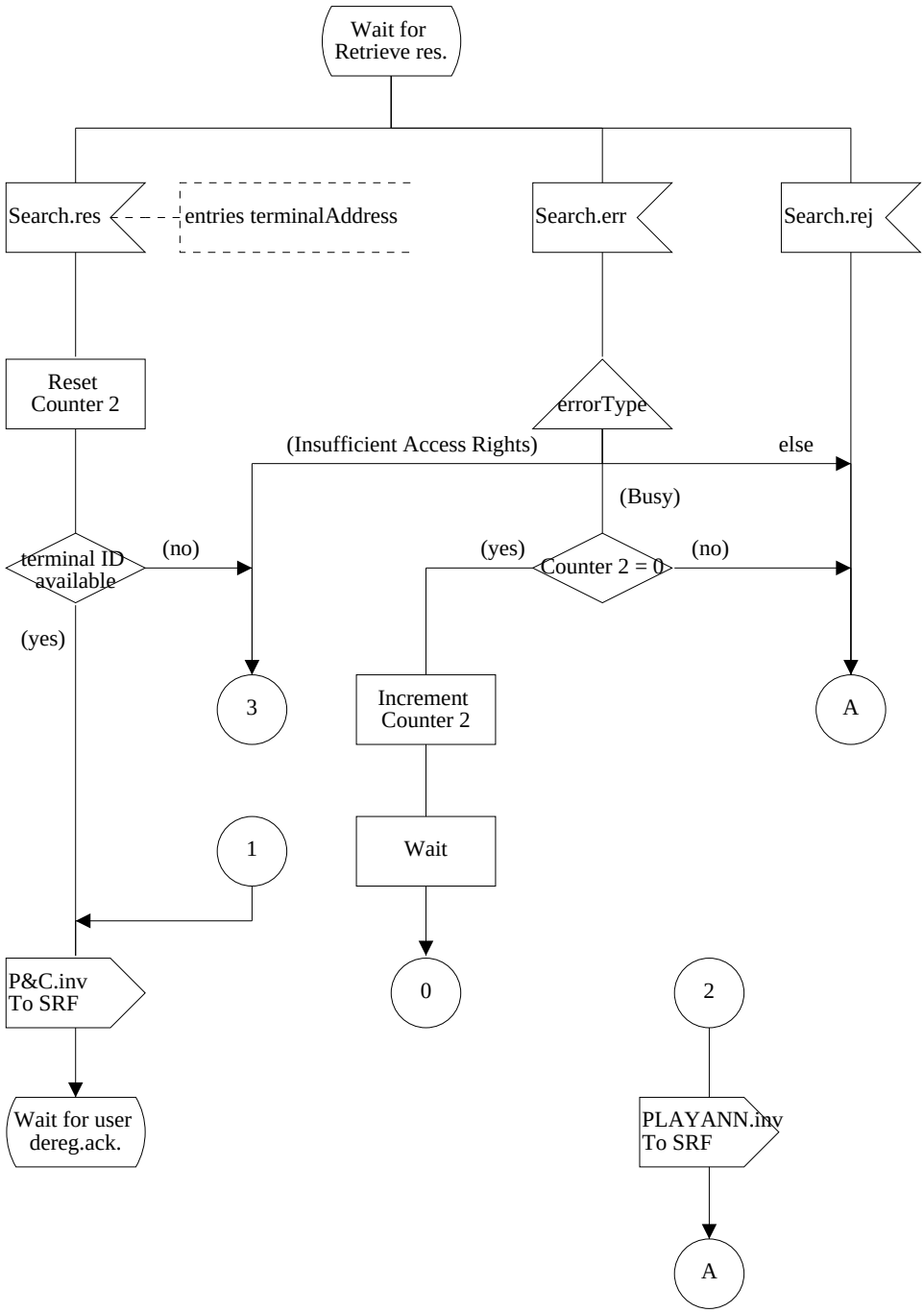


**Figure 13 (sheet 2 of 4): SCF Deregistration for incoming Calls procedure**

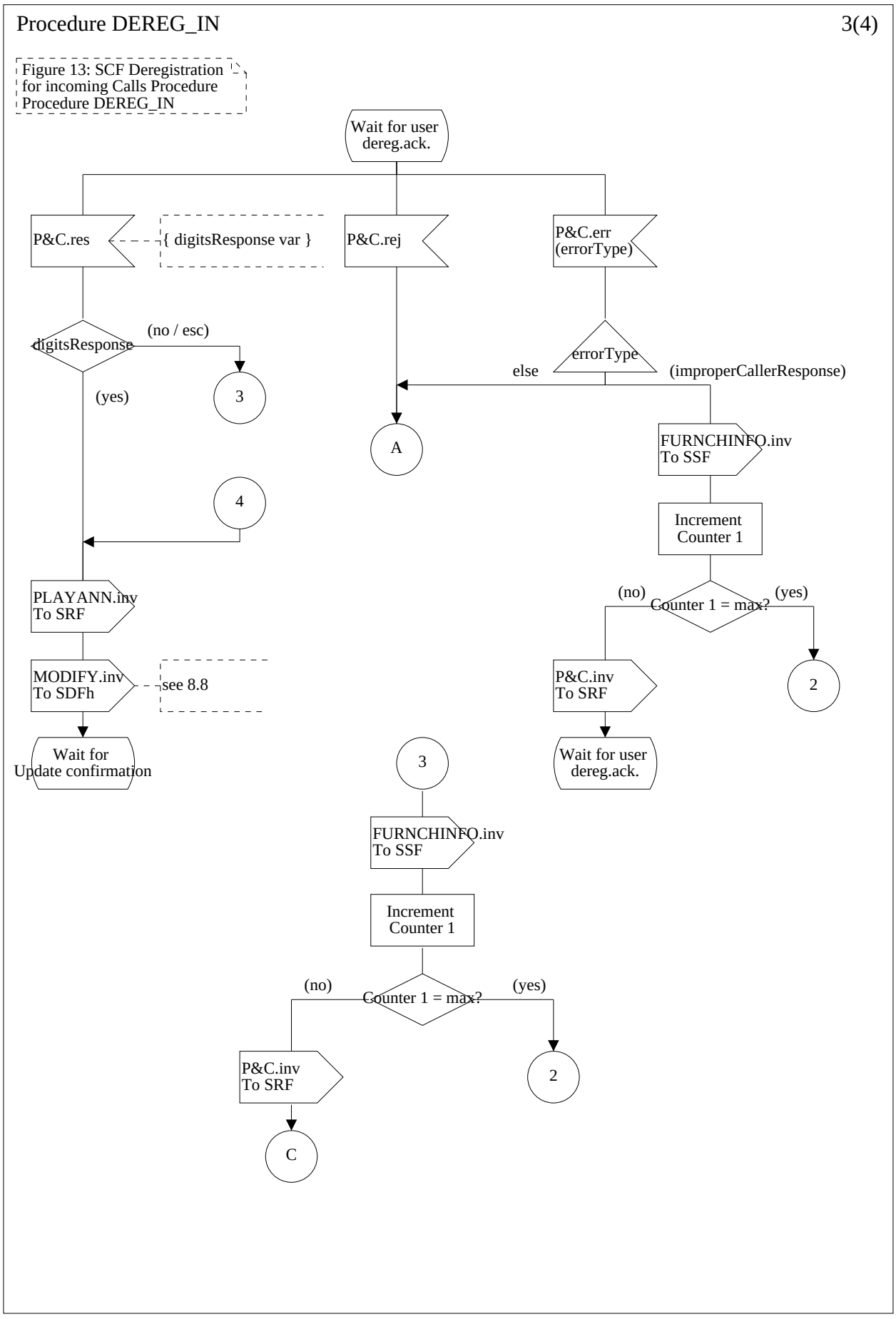


**Figure 13 (sheet 3 of 4): SCF Deregistration for incoming Calls procedure**

Procedure DEREG_IN 4(4)

Figure 13: SCF Deregistration for incoming Calls Procedure Procedure DEREG_IN

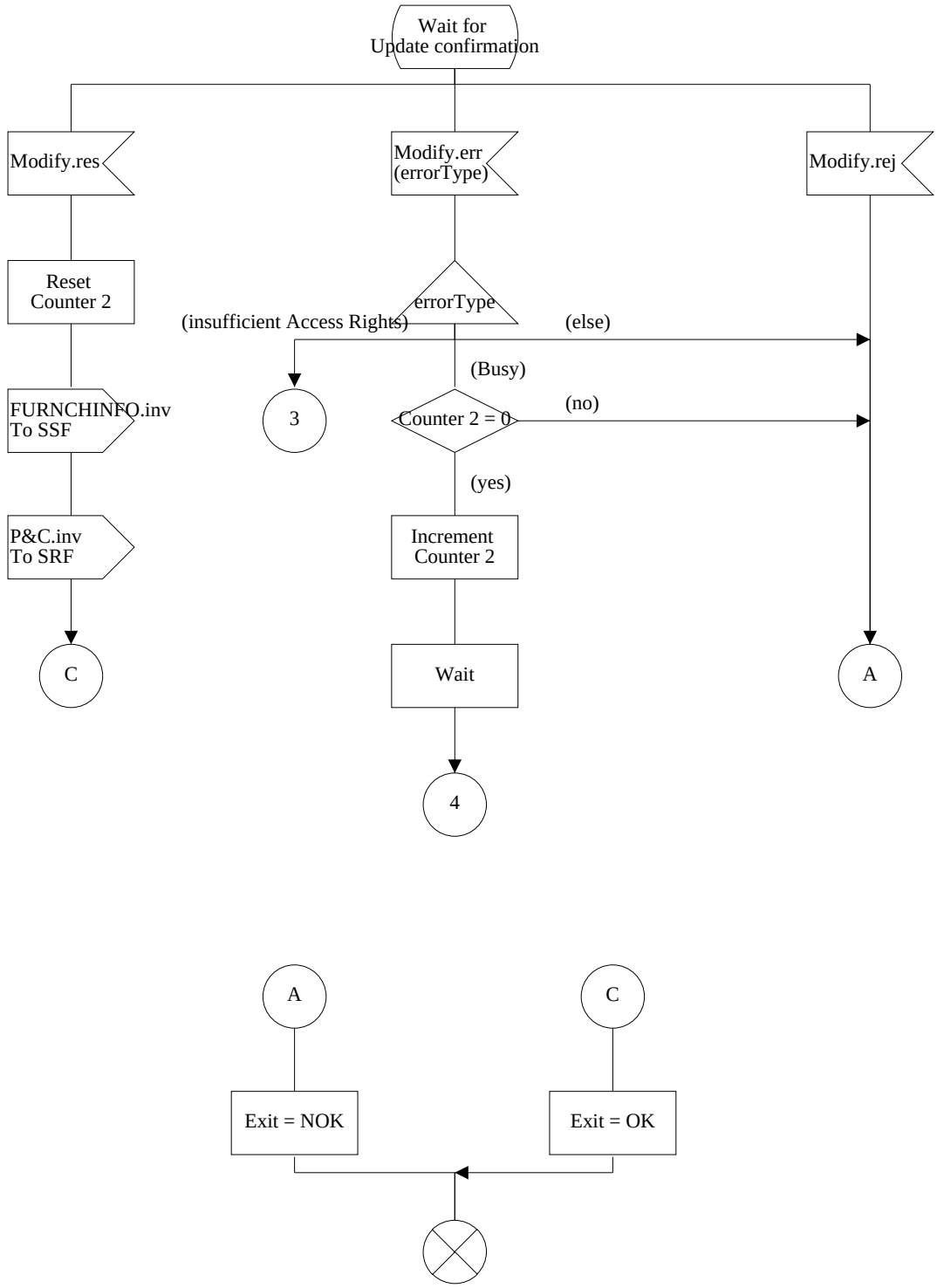


**Figure 13 (sheet 4 of 4): SCF Deregistration for incoming Calls procedure**

## 7.4 Call Handling

UPT call handling procedures are the procedures related to the making and receiving of UPT calls. The description of the procedures assumes that:

a) ISUP signalling will be used for the network signalling;

b) the limit for the redirection counter is a network provider option with an upper limit of 5 redirections;

c) as the default registration address will always be present in the UPT user profile (see subclause 6.2.1.4) the UPT to UPT Call Forwarding on Not Reachable service is not considered as it cannot be used;

d) interaction with ISDN fixed network call diversion services and IN based UPT call diversion services will follow the procedures described in ISUP/INAP interaction ETR 164 [10];

e) to enable UPT to UPT call forwarding to be detected it will be possible to differentiate between a UPT and non UPT number;

f) numbers not recognized as UPT numbers will result in the call being treated as a normal call;

g) as the cost of forwarding a call is at the expense of the forwarding user no service restriction or credit limit checks will be performed on the forwarded-to leg of the call.

### 7.4.1 Outgoing UPT Call

This subclause describes how the UPT user can make a single outgoing UPT call independent of any previous registrations by the UPT user or any other UPT user for incoming and/or outgoing calls to the used terminal access.

#### 7.4.1.1 General

Outgoing calls from a UPT user may be single calls, in which the procedure terminates at the end of the call, or may allow follow-on. The follow-on may be of another call or of another UPT procedure. Follow-on will be offered to the UPT user after the B party disconnects at the end of a conversation or following call set-up failure. The procedures for follow-on are described in more detail in subclause 7.1.3.

The IA and FRI procedures (see subclause 7.2) must have been successfully completed before this procedure.

#### 7.4.1.2 Detailed Procedure

The procedure for outgoing call handling for UPT calls is described in figure 14. The related macro is shown in figure 15 The outgoing call procedure is called by the process UPT_SLP as described in subclause 7.1.3. of this specification.

If Redirection Information is provided in the INITIALDP operation, the Call Forwarding Counter (CFCOUNTER) is set to the same value as the Redirection Counter (see ETS 300 356-15 [1] for a description of this ISUP information element), otherwise the Call Forwarding Counter (CFCOUNTER) is set to zero.

**Destination number**

If the calling UPT user has already provided a destination number the Call Handling procedures continue with screening of the UPT users home data base otherwise, the Destination number is requested via invocation of the operation P&C which is sent to the SRF. The result of this operation will be one of the following events:

a) Dialogue released by IN node (P&C.rej or dialogue_released): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-1 [2]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call set-up procedure as described below in the subclause User Initiated Release.

c) Operation error returned (P&C.err): The possible error causes returned in the P&C operation (.err) are described in ETS 300 374-1 [2]. If the error "Improper Caller Response" is returned, the user is given another opportunity to enter the destination number, providing the users retry counter (COUNTER1)has not been exceeded. If the retry counter (COUNTER1) has been exceeded the calling user is informed that the maximum number of denied call attempts is reached and is asked to hang up, the SCF will instruct the SRF to send this announcement by invoking the PLAYANN operation. The call handling procedure is then terminated and the RELEASE procedure is activated, see subclause 7.2.3. This also applies for all other error causes which may be returned by the P&C operation.

d) Successful result: A destination number is provided by the calling UPT user. The procedure continues as described in the subclause Screening of the home database.

**Screening of the home database**

The SCF checks the calling UPT users home database (SDFhA) for any restrictions which may apply to the dialled number. If the call has already been forwarded but has not been forwarded more times than permitted by the network (the maximum number of times a single call can be diverted is a network provider option, there is an upper limit of 5, refer to ETS 300 356-15 [1] for further information) this check will not be performed. If the call has already been forwarded more times that permitted by the network, this call will be terminated and the calling UPT user will be given an opportunity to enter a different destination number as described below in the subclause User Retry.

To screen the dialled number the SCF will invoke the SEARCH operation (as defined in subclause 8.10). The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the Release procedure (as described in subclause 7.2.3.) is invoked;

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation (.err) is described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries (COUNTER2) is exceeded, the Release procedure is performed.

   2) For all other error types returned by the SDF, the procedure is terminated with Exit = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDF. If there are no restrictions the procedure will continue as described below. If restrictions apply, the calling UPT user will be given an opportunity to enter a different destination number as described below in the subclause User retry.

A further query may then be made on the UPT users home database (SDFhA) to check the Credit Limit to see if there is any credit available to make the call. This check will not be repeated for subsequent call set-up attempts for the same call (i.e. UPT to UPT Call Forwarding has occurred). The SCF will invoke another SEARCH operation (as defined in subclause 8.9) for this purpose. The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the Release procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation (.err) is described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

   2) For all other error types returned by the SDF, the procedure is terminated with Exit = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDF. If credit is available to make the call the procedure will continue as described below. If there is no credit available the calling UPT user will be given an opportunity to select another feature, the procedure will continue as described in the subclause User retry.

For UPT to non-UPT user calls the call handling procedures continue with the Call Set-up procedures.

**UPT to UPT call**

For UPT to UPT user calls (i.e the called number is recognized as a UPT number), the originating network database (SDFo) is interrogated to check if agreements between the local service provider and the called UPT users (or the forwarded-to UPT users) home provider exist for establishing outgoing calls. An analysis of the number provided by the calling user should be performed to avoid retriggering the same SCF (the problem of interaction between IN and other services is to be solved). This interrogation is performed by invoking the SEARCH operation (as defined in subclause 8.3). The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation (.err) are described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries (COUNTER2) is exceeded, the Release procedure is performed.

   2) For all other error types returned by the SDF, the procedure is terminated with Exit = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDF. If the result is a match (i.e. agreement exists between the local service provider and the called UPT users home provider for establishing outgoing calls from the current location) then the procedure will continue as described below. In the case of no match (i.e. there is no agreement to establish outgoing calls) the call will be treated as a normal call and routed to another network for completion, for this the procedure continues as described in the subclause call set-up.

Following the successful service provider check, the database of the home network of the called or forwarded UPT user (SDFhB) is then interrogated to retrieve the location of the called user. The dialogue is opened with an "empty" Directory BIND operation (i.e the Credentials parameter will not be present), the outcome of this operation can be one of the following events:

a) Dialogue released by IN node (Dialogue_released or BIND.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (BIND.err): The possible error causes returned in the SEARCH operation (.err) are described in ETS 300 374-5 [3] regardless of the reason for the error the call handling procedure is terminated and the calling UPT user will be given an opportunity to enter a different destination number or select another feature, the procedure will continue as described in the subclause User retry.

d) Successful result (BIND.res): This means that SDFhB accepts the dialogue and the procedure continues as described below.

To retrieve the location of the called user the operation SEARCH (as defined in subclause 8.11) is invoked by the SCF, the outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the Release procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation result (.err) are described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

   2) For all other error types returned by the SDF, the procedure is terminated with EXIT = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDFhB. The result is the routeing address(es) applicable at the time the request is made. If more than one routeing address is returned the SCF will select the address to use based on the following priority:

   1) The routeing address for the Call Forwarding Unconditional service, if this service is active.

   2) If the Call Forwarding Unconditional service is not active but the registration is still valid, the routeing address used will be the registration address.

   3) If the Call Forwarding Unconditional service is not active but the Variable Routeing service is active, the address used will depend on the time or on the calling user.

   4) Default registration address if none of the above criteria apply.

**Retrieve Default Charging Reference Point**

If the call is being forwarded the retrieval of the default charging reference point is not required. The charging for the forwarded leg of the call is a matter for the original called user and is not described here.

To retrieve the default charging reference point the SDFhB is interrogated by the SCF using a SEARCH operation (as defined in subclause 8.16) the outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation result (.err) are described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

   2) For all other error types returned by the SDF, the procedure is terminated with Exit = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDFhB. The procedure then continues as described below.

If split charging (see ETR 055-3 [7]) is not to be applied the procedure continues with the UPT charging notification subclause below. If split charging is to be applied then the called UPT user's credit limit is checked to determine if there is sufficient credit available to receive the call. The SCF will invoke another SEARCH operation (as defined in subclause 8.9) for this purpose. The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation result (.err) are described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

   2) For all other error types returned by the SDF, the procedure is terminated with Exit = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDFhB. If credit is available to make the call the procedure will continue as described in subclause UPT Charging Notification. If there is no credit available the calling UPT user will be given an opportunity to select another feature, the procedure will continue as described in the subclause User retry.

**UPT Charging Notification**

The calling user is informed that "UPT Charging is Applicable" via invocation of the PLAYANN operation. This notification will not be repeated for subsequent call set-up attempts (i.e. UPT to UPT call forwarding has occurred and the calling user was notified that UPT charging was applicable from a previous attempt to set-up this call). Following the instruction for the PLAYANN operation the macro SRF_Disconnect is called, this macro will handle the operation errors and the disconnection of the SRF.

**Retrieve supplementary service information**

The purpose of this part of the procedure is to query the called UPT users home database (SDFhB) for the status of supplementary services and for those call forwarding services which are active, retrieve the relevant conditional forwarding parameters (e.g. No Reply Condition Timer). The SCF will invoke a SEARCH operation (as defined in subclause 8.13) for this. The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation result (.err) are described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

   2) For all other error types returned by the SDF, the procedure is terminated with Exit = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDFhB. The call forwarding parameters retrieved (if any) are stored for future use. The procedure continues as described below.

**Call set-up**

The SCF can now instruct the SSF to set-up the call, several operations are invoked for this purpose:

a) DISCFWDCONN: This instructs the SSF to release the SRF, this operation will not be repeated for subsequent call set-up attempts (i.e. UPT to UPT Call Forwarding has occurred). This operation is invoked from within the macro SRF_Disconnect.

b) REQREPBCSM: This requests the SSF to monitor for a call-related event (e.g. busy, no answer, release...) and report back to the SCF when the event has been detected. If a No Reply Condition Timer value was retrieved from the called UPT users home database (SDFhB) when searched for supplementary service information, the value will be provided with this operation.

c) FURNCHGINFO: This requests the SSF to generate call record information for the following event.

d) APPLYCHG: This operation requests the SSF to report back to the SCF when a charging related event has been detected.

e) CONNECT: This instructs the SSF to set-up the call (i.e. generate the IAM). If any redirection information was provided in the INITIALDP operation the information will be returned in this operation, the redirection counter may have been updated as a result of further call forwarding. If no redirection information was provided in the INITIALDP operation but call forwarding has occurred, then the redirection information will be constructed by the SCF. Redirection information will not be provided if no call forwarding has occurred. The SSF will determine how to handle this information.

The outcome of this procedure will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or <operation_name>.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-1 [2]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Called Party busy: This state is reported to the SCF by the SSF returning the operation EVREPBCSM. There are three possible outcomes:

   1) If call forwarding on busy service is not active, the calling UPT user is informed that the call cannot be connected and is provided with an opportunity to enter another destination number or select another feature. The SCF will first ask the SSF to establish a temporary connection to the SRF by invoking the SRF_Connect procedure (see subclause 7.2.4). The procedure continues as described in the subclause User retry.

   2) If call forwarding on busy service is active but the Call Forwarding Counter (CFCOUNTER) has exceeded the network redirection limit (note that the upper limit for this counter is 5), then the calling UPT user is informed that the call cannot be connected and provide the UPT user with an opportunity to enter another destination number or select another feature. The SCF will first ask the SSF to establish a temporary connection to the SRF by invoking the SRF_Connect procedure (see subclause 7.2.4). The procedure continues as described in the subclause User retry.

   3) If call forwarding on busy service is active, and the Call Forwarding Counter (CFCOUNTER) has not exceeded the network redirection limit, the call can be forwarded. The Call Forwarding Counter (CFCOUNTER) is incremented and the outgoing call procedure restarted.

d) Called Party no answer: This state is reported to the SCF by the SSF returning the operation EVREPBCSM. There are three possible outcomes:

   1) If call forwarding on no reply service is not active, the calling UPT user is informed that the call cannot be connected and is provided with an opportunity to enter another destination number or select another feature. The SCF will first ask the SSF to establish a temporary connection to the SRF by invoking the SRF_Connect procedure (see subclause 7.2.4). The procedure continues as described in the subclause User retry.

   2) If call forwarding on no reply service is active but the Call Forwarding Counter (CFCOUNTER) has exceeded the network redirection limit (note that the upper limit for this counter is 5), then the calling UPT user is informed that the call cannot be connected and provide the UPT user with an opportunity to enter another destination number or select another feature. The SCF will first ask the SSF to establish a temporary connection to the SRF by invoking the SRF_Connect procedure (see subclause 7.2.4). The procedure continues as described in the subclause User retry.

   3) If call forwarding on no reply service is active and the Call Forwarding Counter (CFCOUNTER) has not exceeded the network redirection limit, the call can be forwarded. The Call Forwarding Counter (CFCOUNTER) is incremented and the outgoing call procedure restarted.

e) Route select failure: This state will be reported to the SCF by the SSF returning the operation EVREPBCSM. This indicates that it was not possible to complete set up of the call due to either congestion, unsubscribed number or number blocked. The calling UPT user is informed that the call cannot be connected and provide the UPT user with an opportunity to enter another destination number or select another feature. The SCF will first ask the SSF to establish a temporary connection to the SRF by invoking the SRF_Connect procedure (see subclause 7.2.4). The procedure continues as described in the subclause User retry.

f) Operation error returned (CONNECT.err, REQREPBCSM.err, FURNCHGINFO.err, APPLYCHGRPT.err or DISCFWDCONN.err): The possible error causes returned by these operations are described in ETS 300 374-1 [2]. Regardless of the reason for the error, the call set-up procedure is terminated and the calling user is then given an opportunity to select another feature, the procedure continues as described in the subclause User retry.

g) Called Party Answers: The SCF is notified of this event by an EVREPBCSM operation.

**Call release and follow-on**

The set-up having been completed, the SCF waits for the release of the call. The SCF is notified of the release by the SSF sending an EVREPBCSM operation. If the B party releases the call and the A party does not release the call immediately this means that a follow-on call is required.

Before invoking the follow-on procedure, the SCF sends the FURNCHGINFO operation to the SSF, this requests the SSF to update the call record. The SCF then waits for the call record from the SSF, during this period the calling UPT user may choose to release the call, the SCF is notified of this event by an EVREPBCSM operation from the SSF.

The SSF will provide the call record in the APPLYCHGRPT operation. This information is then used to update the Calling Party's home database (SDFhA) by the SCF invoking the MODIFY operation (as defined in subclause 8.12) to store the call record in the SDF. It should be noted that it is not possible to directly use the call record to modify the users credit as the SDF is not able to calculate the charge that corresponds to the call record. The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or MODIFY.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure as described below in the subclause User Initiated Release.

c) Operation error returned (MODIFY.err): The possible error causes returned in the MODIFY operation (.err) are described in ETS 300 374-5 [3]:

   1) If the error is "Busy", the MODIFY operation can be attempted again, after a time delay. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed but to avoid the loss of the call record an implementation dependant action could be performed at this stage.

   2) For any other error the procedure is terminated. To avoid loss of the call record an implementation dependant action could be performed at this stage.

d) Successful result (MODIFY.res): The SDF will report the update of the database with MODIFY.res.

In the case of UPT to UPT calls the original called UPT user's home database (SDFhB) will also be updated in the same manner as described above.

To offer the follow-on to the user the SCF will instruct the SSF to reconnect an SRF through the SRF_Connect procedure described in subclause 7.2.4.

Once the connection to the SRF is confirmed the SCF will instruct the SRF to inform the UPT user that another request can be made or terminate, by sending the operation P&C. The follow-on procedure then continues with the feature request identification procedures as described in subclause 7.2.2.

The dialogue to the original SDFhB is released. There may be more than one SDFhB dialogues open, these should also be released.

**User retry**

Firstly, the SCF will request the SSF to create a call record for the following call event by invoking the operation FURNCHGINFO. The retry counter (COUNTER1) is incremented.

If the retry counter (COUNTER1) has been exceeded the calling user is informed that the maximum number of denied call attempts is reached and is asked to hang up, the SCF will instruct the SRF to send this announcement by invoking the PLAYANN operation. The call handling procedure is then terminated and the Release procedure is activated, see subclause 7.2.3.

If the retry counter (COUNTER1) has not been exceeded the calling user is informed that the request is denied and is requested to either hang up or to make another feature request. The SCF will instruct the SRF to send this announcement by invoking the PLAYANN operation. The procedure then continues with the Feature Request Identification procedure as described in subclause 7.2.2, call handling procedures are terminated.

**User Initiated Release**

The SCF, following notification of early release of the call by the calling party, waits for the APPLYCHGRPT operation which returns to the SCF the call record. Once this message is received, the SCF sends a MODIFY operation (as defined in subclause 8.12) to SDFhA to store the call record in the SDF. It should be noted that it is not possible to directly use the call record to modify the user credit as the SDF is not able to calculate the charge that corresponds to the call record. The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or MODIFY.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3) is invoked.

b) Operation error returned (MODIFY.err): The possible error causes returned in the MODIFY operation ration (.err) are described in ETS 300 374-5 [3]:

   1) If the error is "Busy", the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries is exceeded (COUNTER2) the procedure is terminated. To avoid loss of the call record an implementation dependant operation could be performed at this stage.

   2) For any other error the procedure is terminated. To avoid loss of the call record an implementation dependant operation could be performed at this stage.

c) Successful result (MODIFY.res): This confirms that the SDF has been successfully updated.

If the call is a UPT to UPT call the same procedure as described for SDFhA will be performed on the original SDFhB.

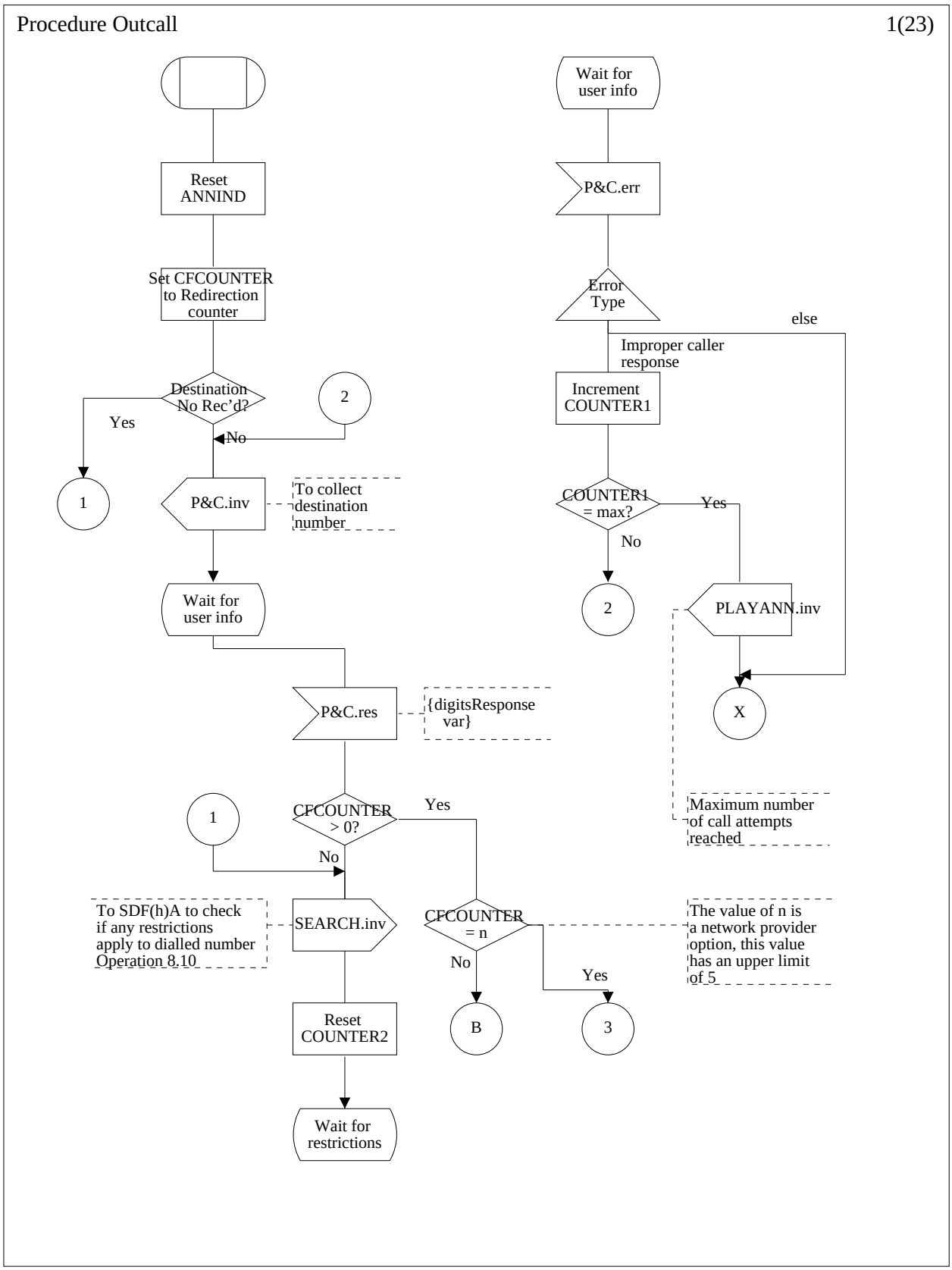


**Figure 14 (sheet 1 of 23): Outgoing UPT Call procedure**

Procedure Outcall 2(23)

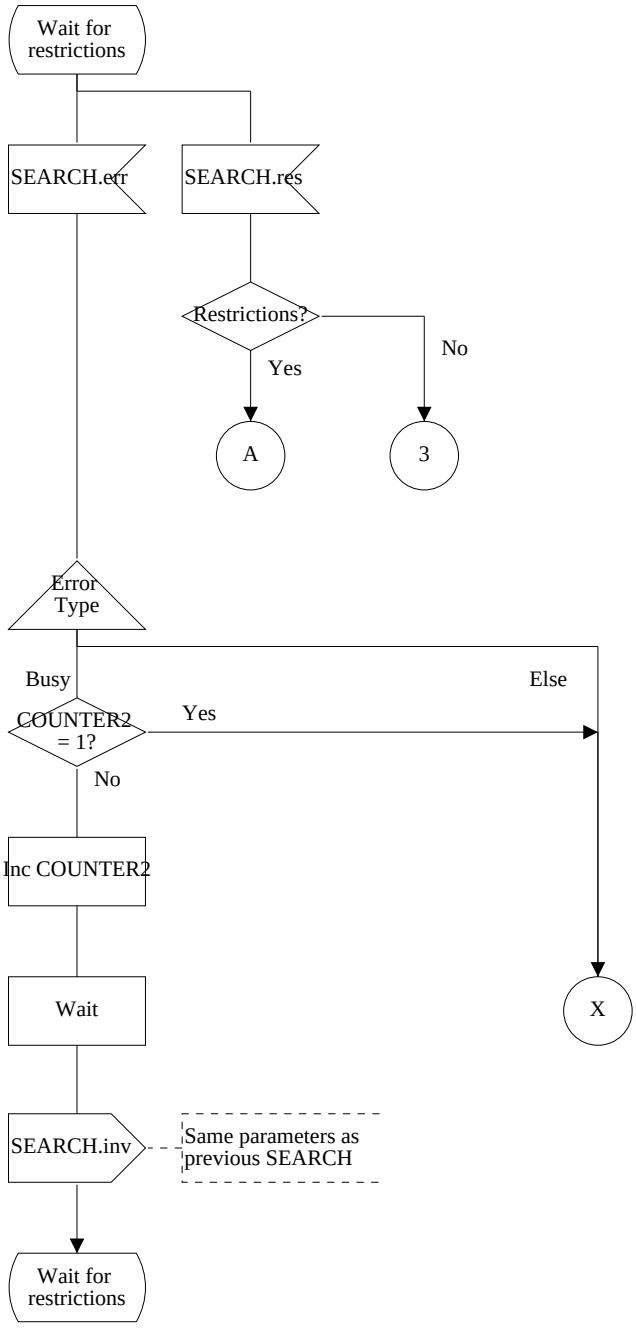


**Figure 14 (sheet 2 of 23): Outgoing UPT Call procedure**

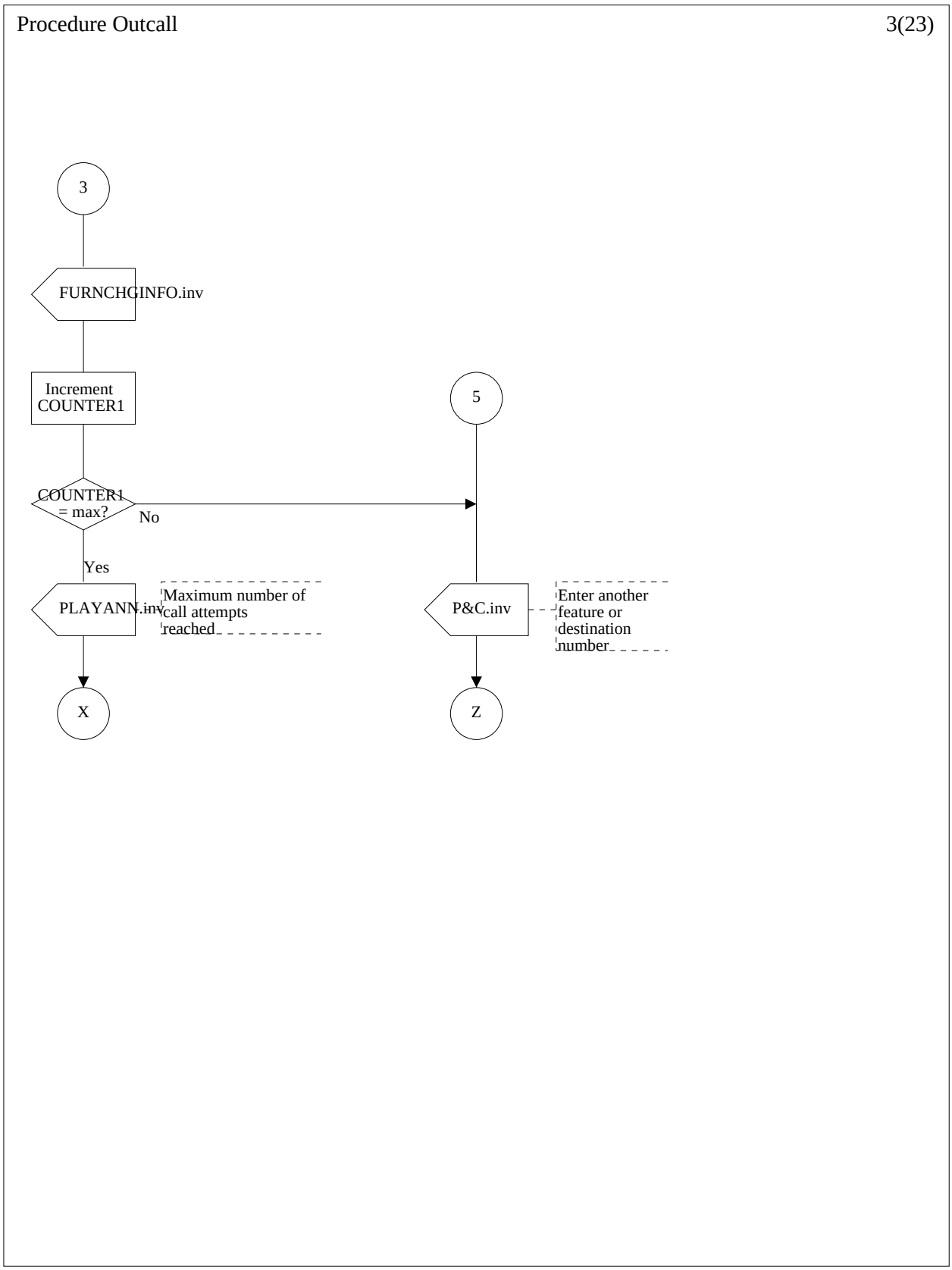


**Figure 14 (sheet 3 of 23): Outgoing UPT Call procedure**

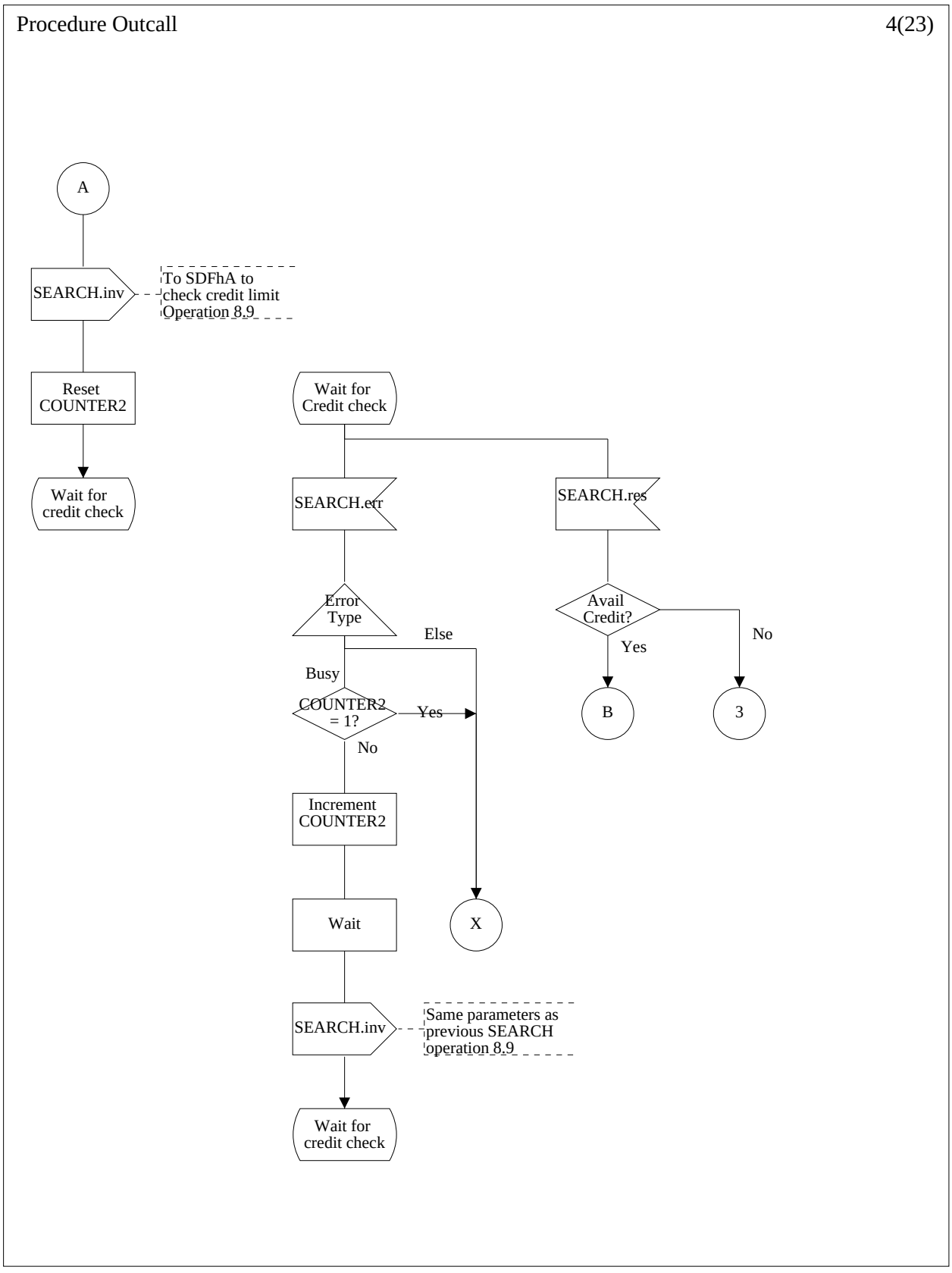


**Figure 14 (sheet 4 of 23): Outgoing UPT Call procedure**

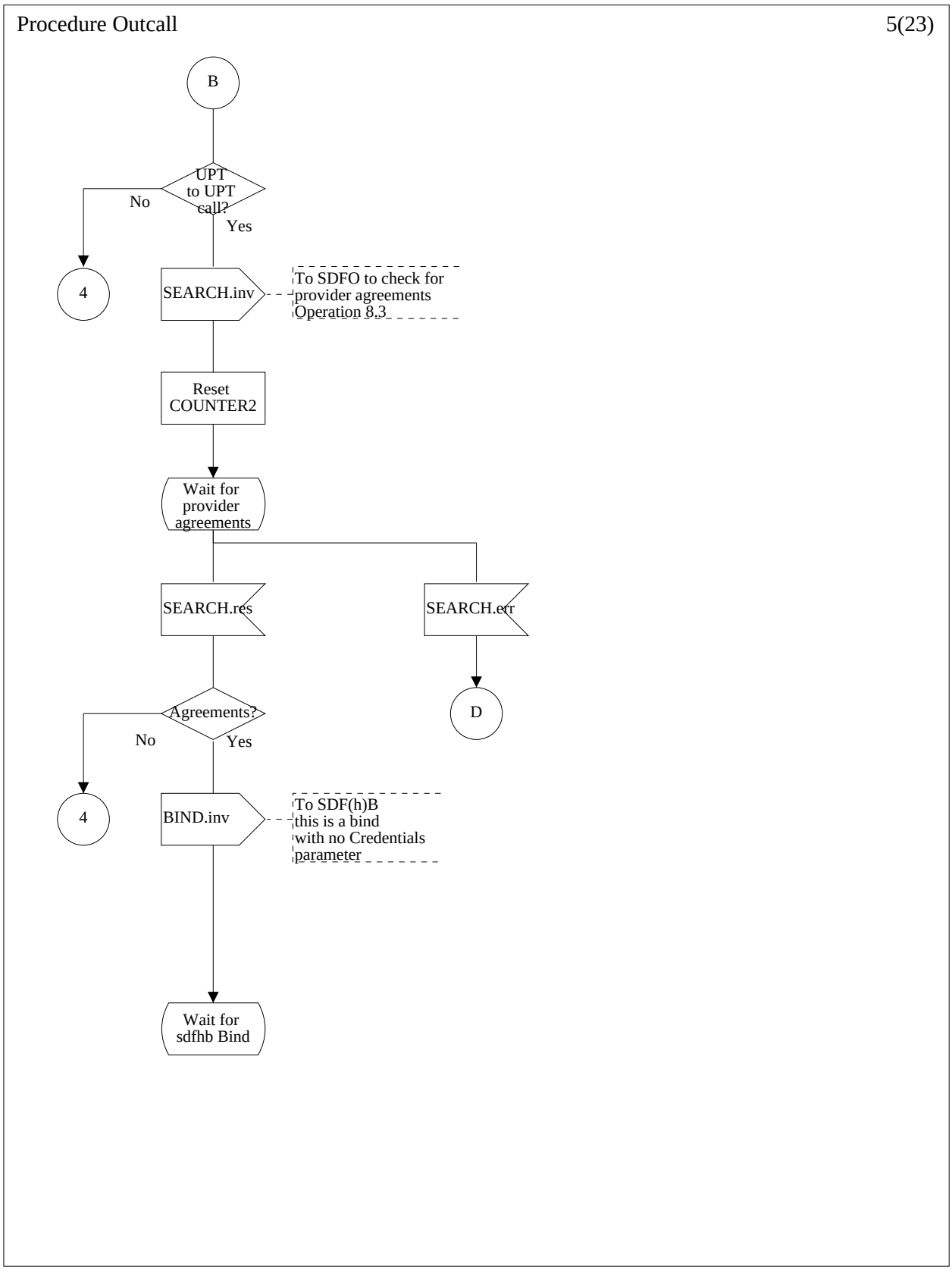


**Figure 14 (sheet 5 of 23): Outgoing UPT Call procedure**

Procedure Outcall 6(23)

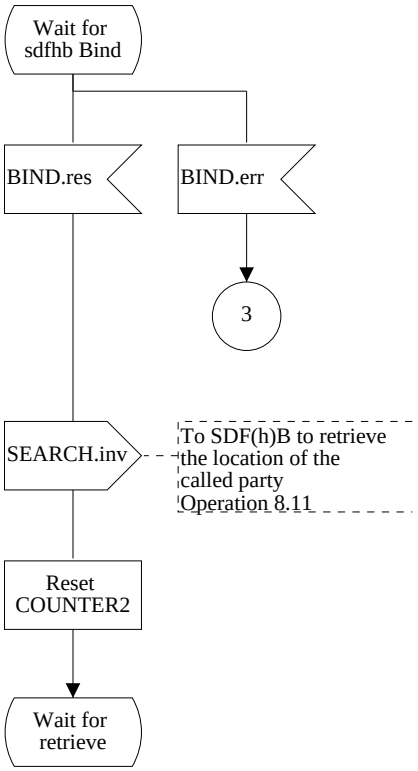


**Figure 14 (sheet 6 of 23): Outgoing UPT Call procedure**

Procedure Outcall 7(23)

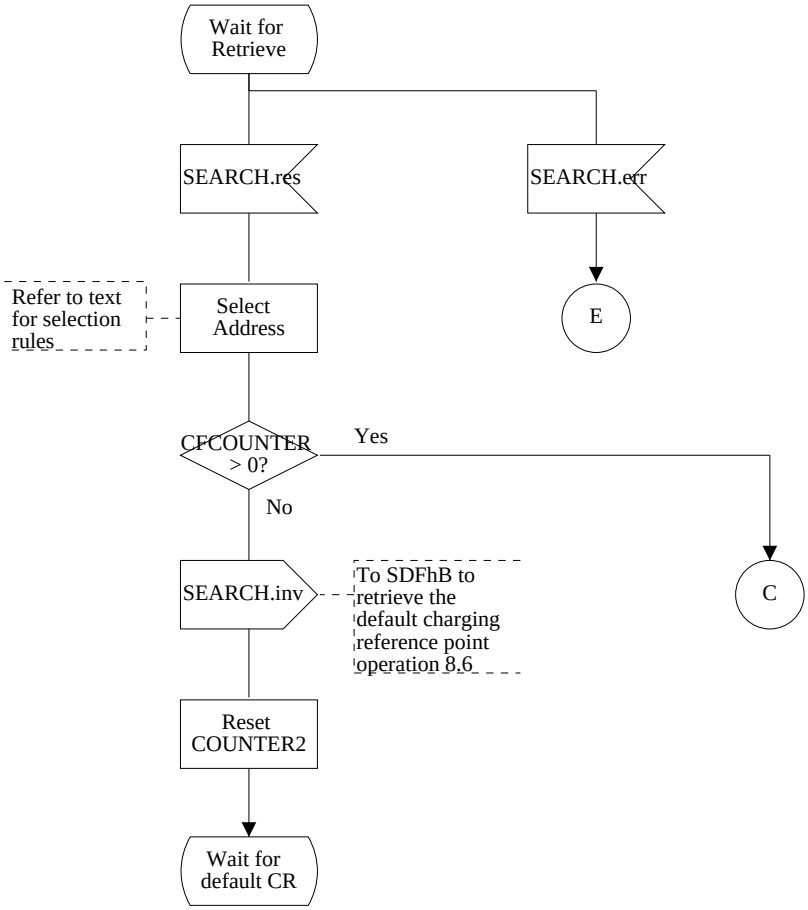


**Figure 14 (sheet 7 of 23): Outgoing UPT Call procedure**

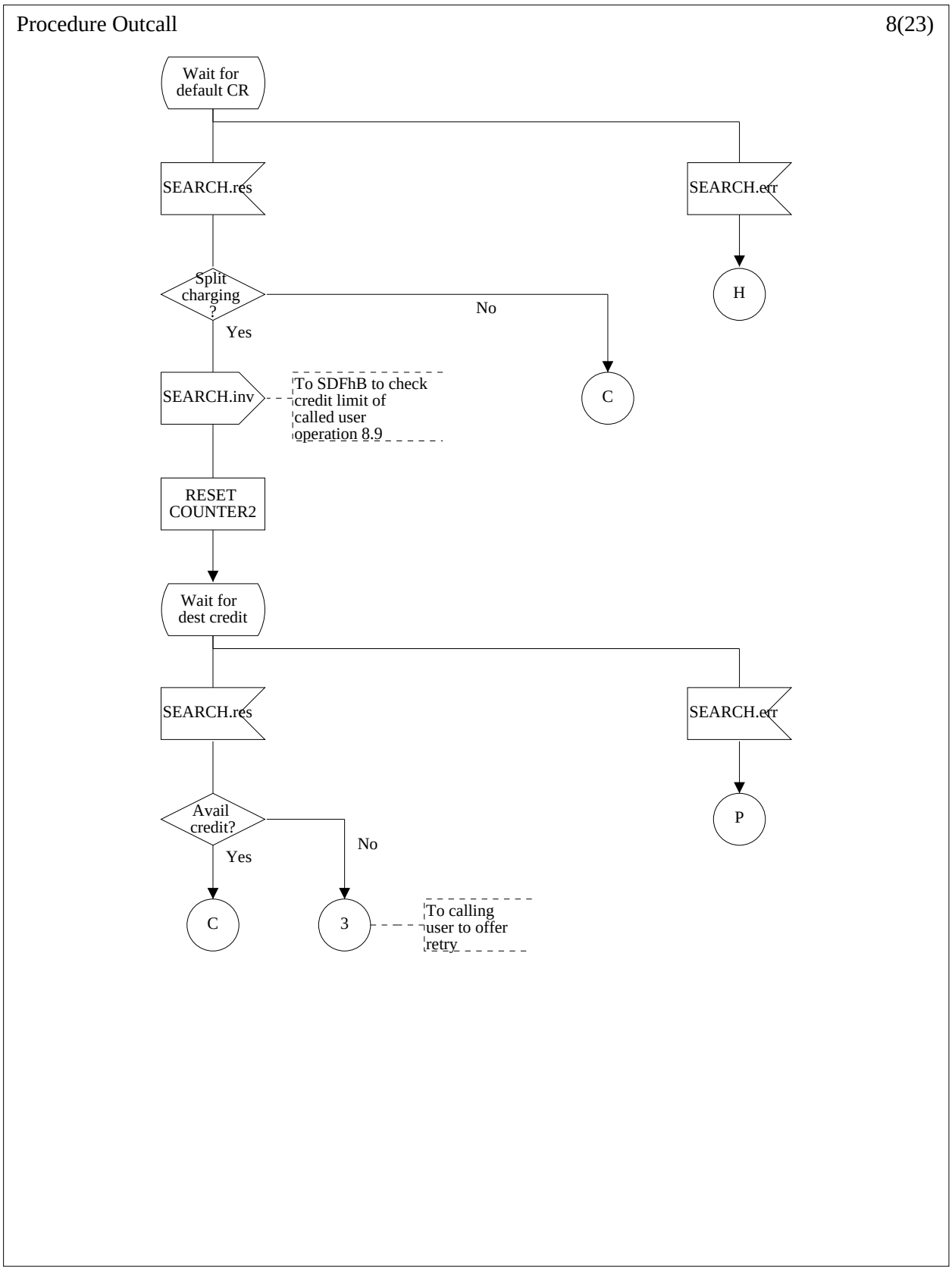


**Figure 14 (sheet 8 of 23): Outgoing UPT Call procedure**

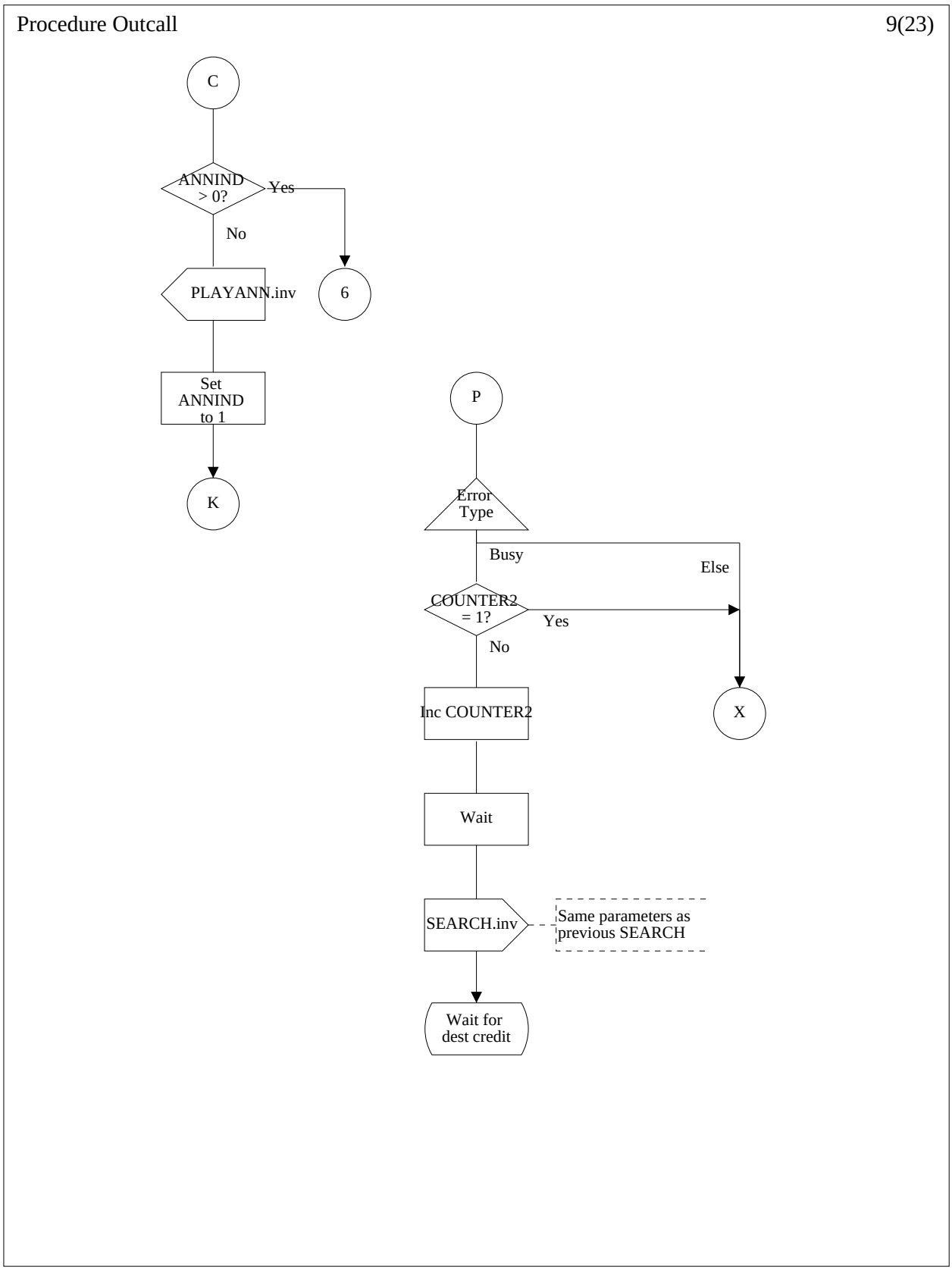


**Figure 14 (sheet 9 of 23): Outgoing UPT Call procedure**

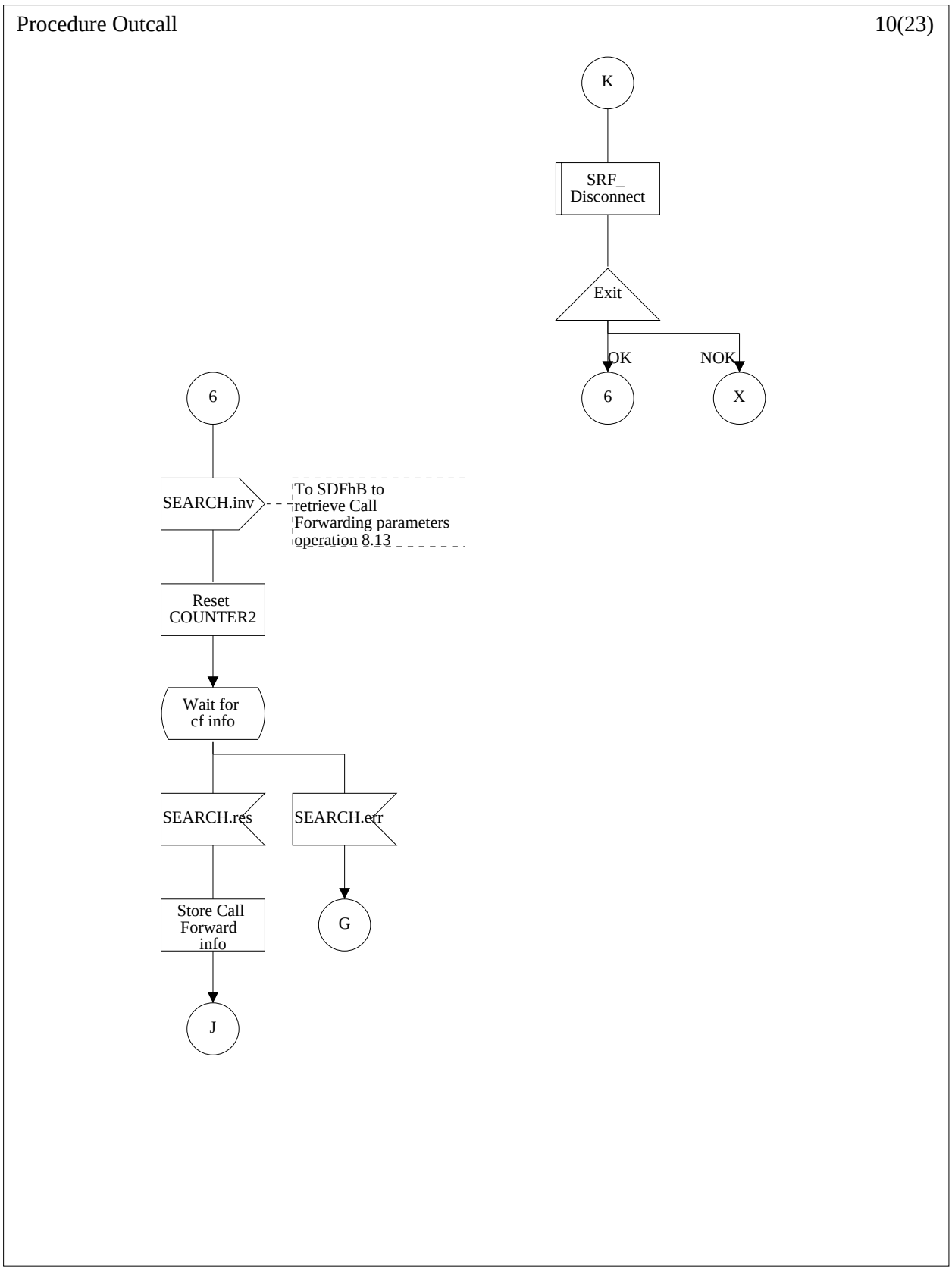


**Figure 14 (sheet 10 of 23): Outgoing UPT Call procedure**

Procedure Outcall 11(23)

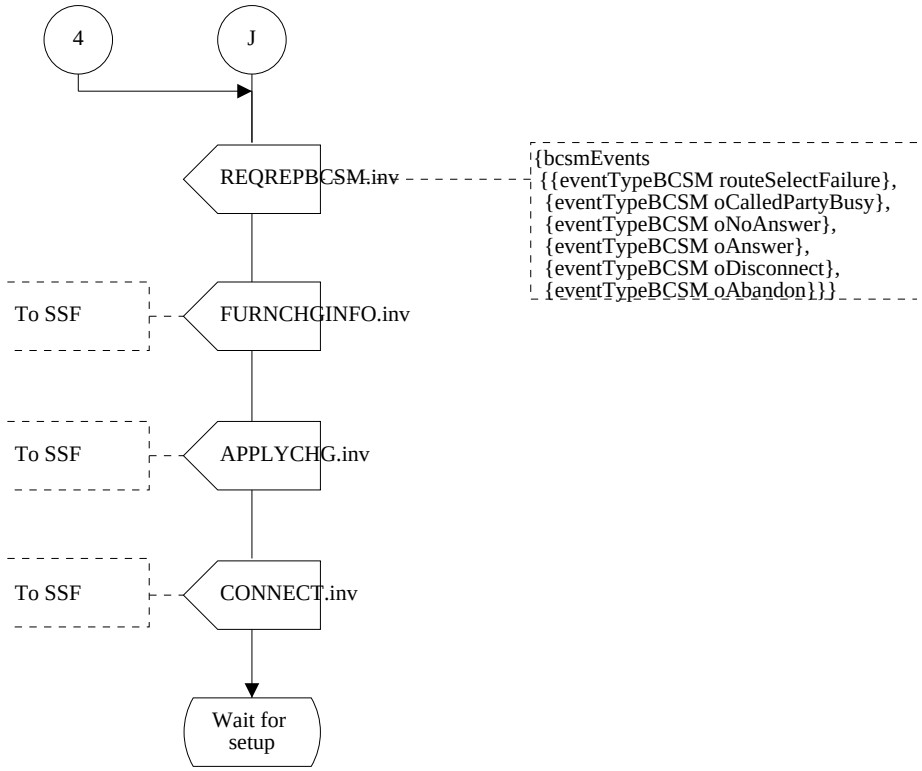


**Figure 14 (sheet 11 of 23): Outgoing UPT Call procedure**

Procedure Outcall 12(23)

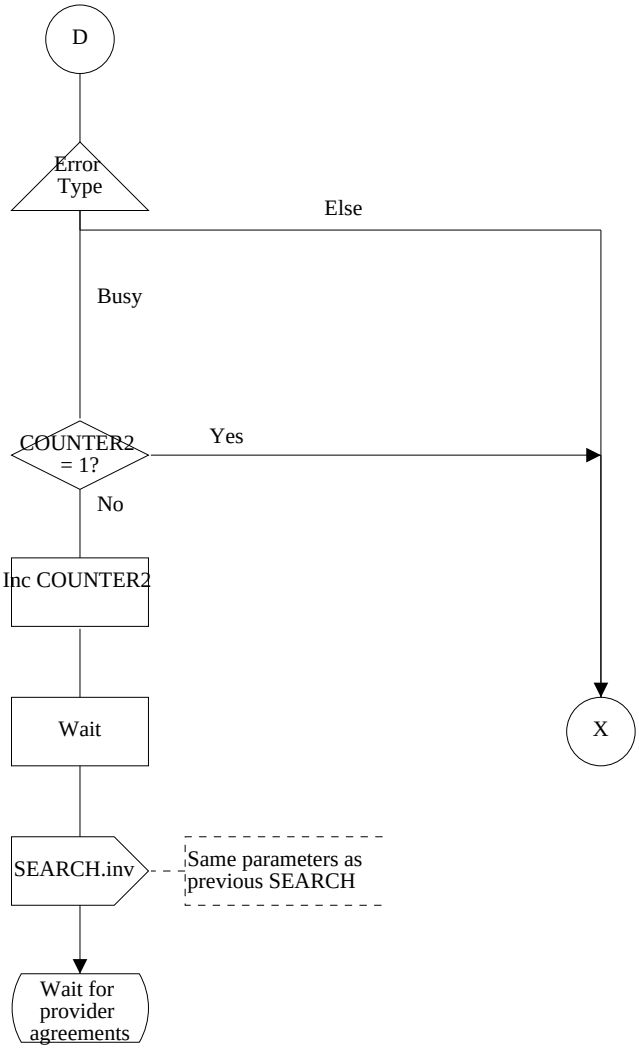


**Figure 14 (sheet 12 of 23): Outgoing UPT Call procedure**

Procedure Outcall 13(23)

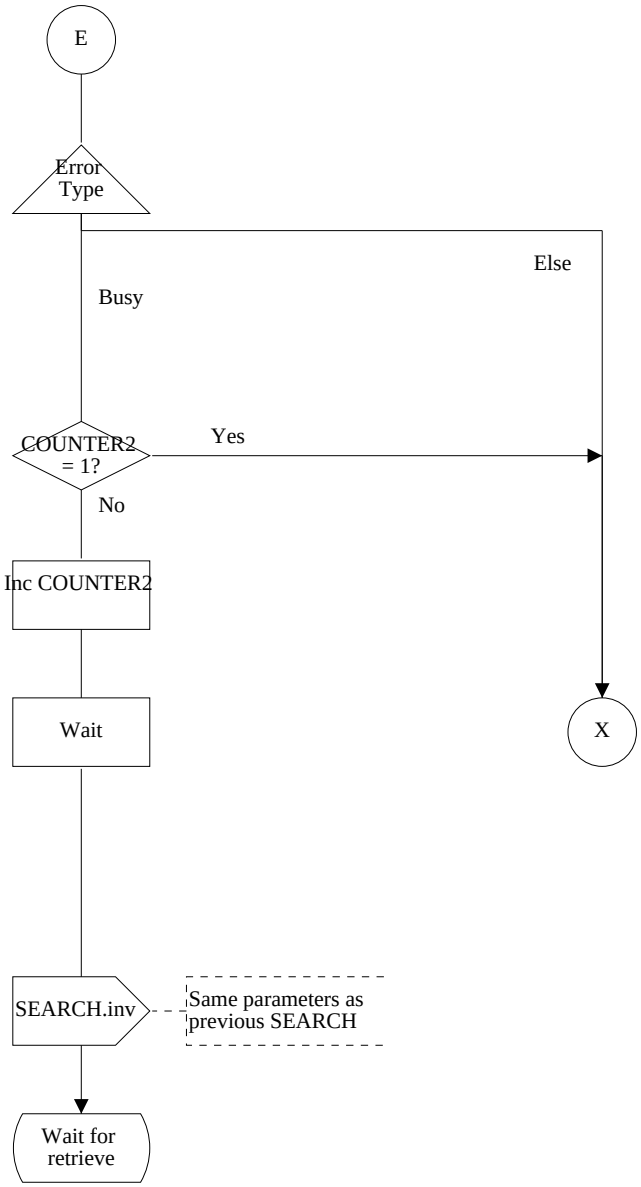


**Figure 14 (sheet 13 of 23): Outgoing UPT Call procedure**

Procedure Outcall 14(23)

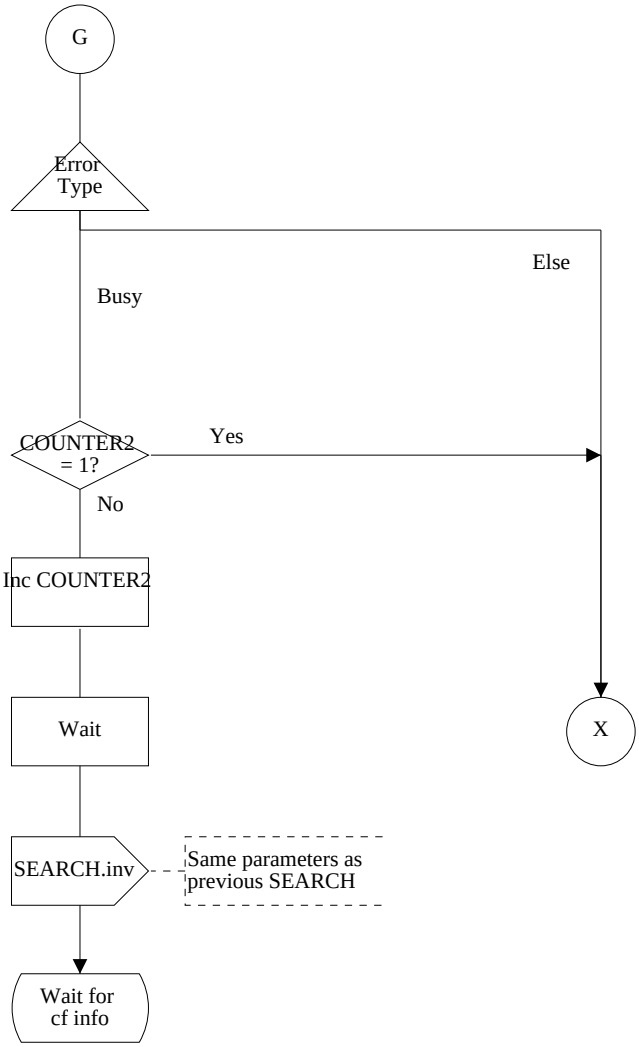


**Figure 14 (sheet 14 of 23): Outgoing UPT Call procedure**

Procedure Outcall 15(23)

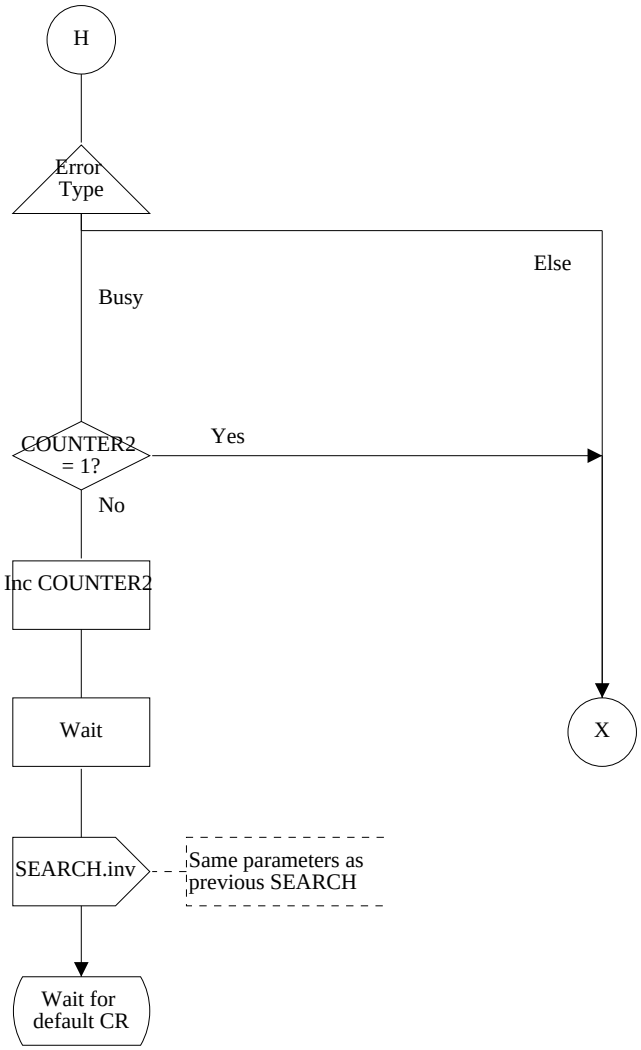


**Figure 14 (sheet 15 of 23): Outgoing UPT Call procedure**

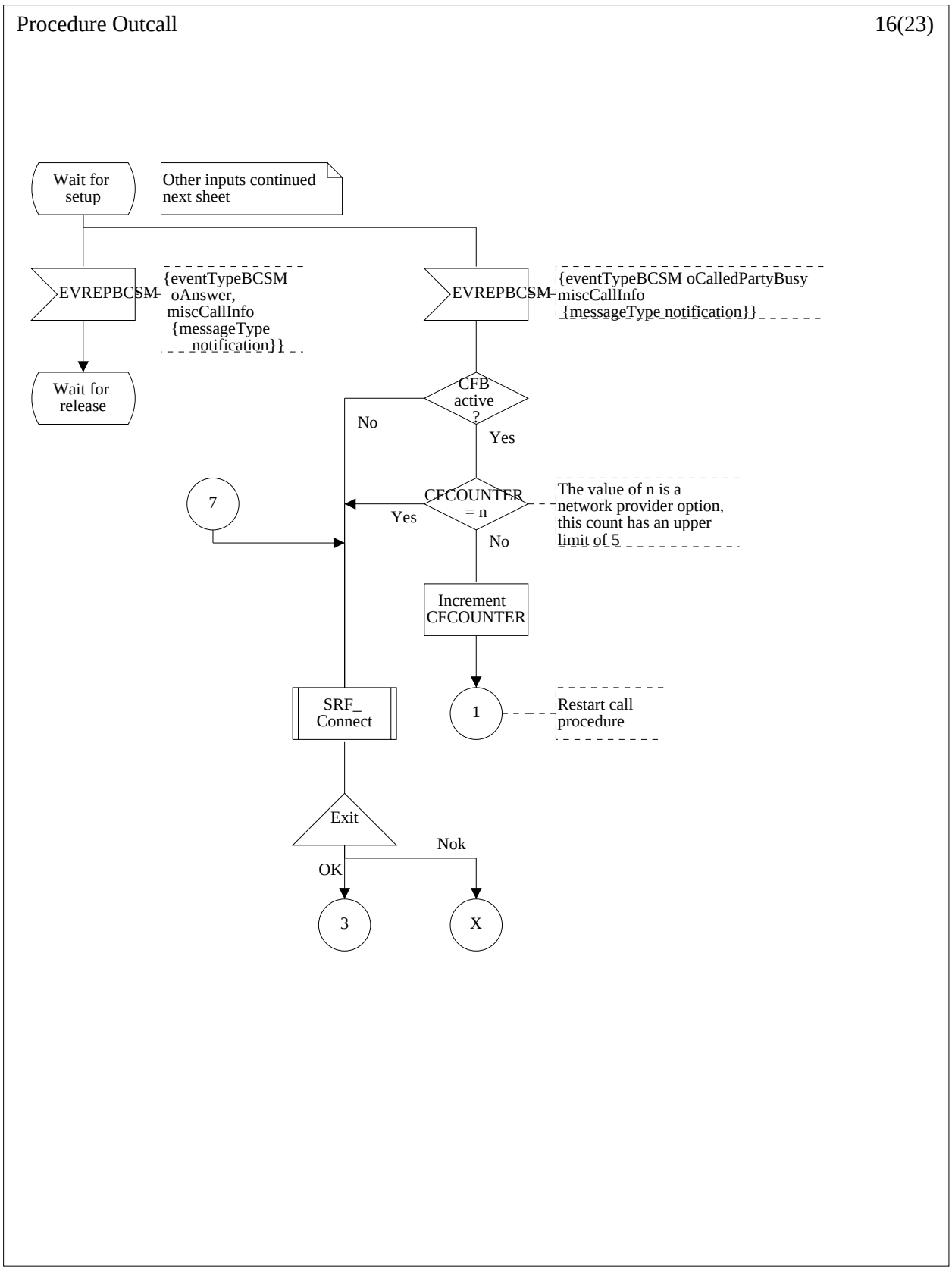


**Figure 14 (sheet 16 of 23): Outgoing UPT Call procedure**

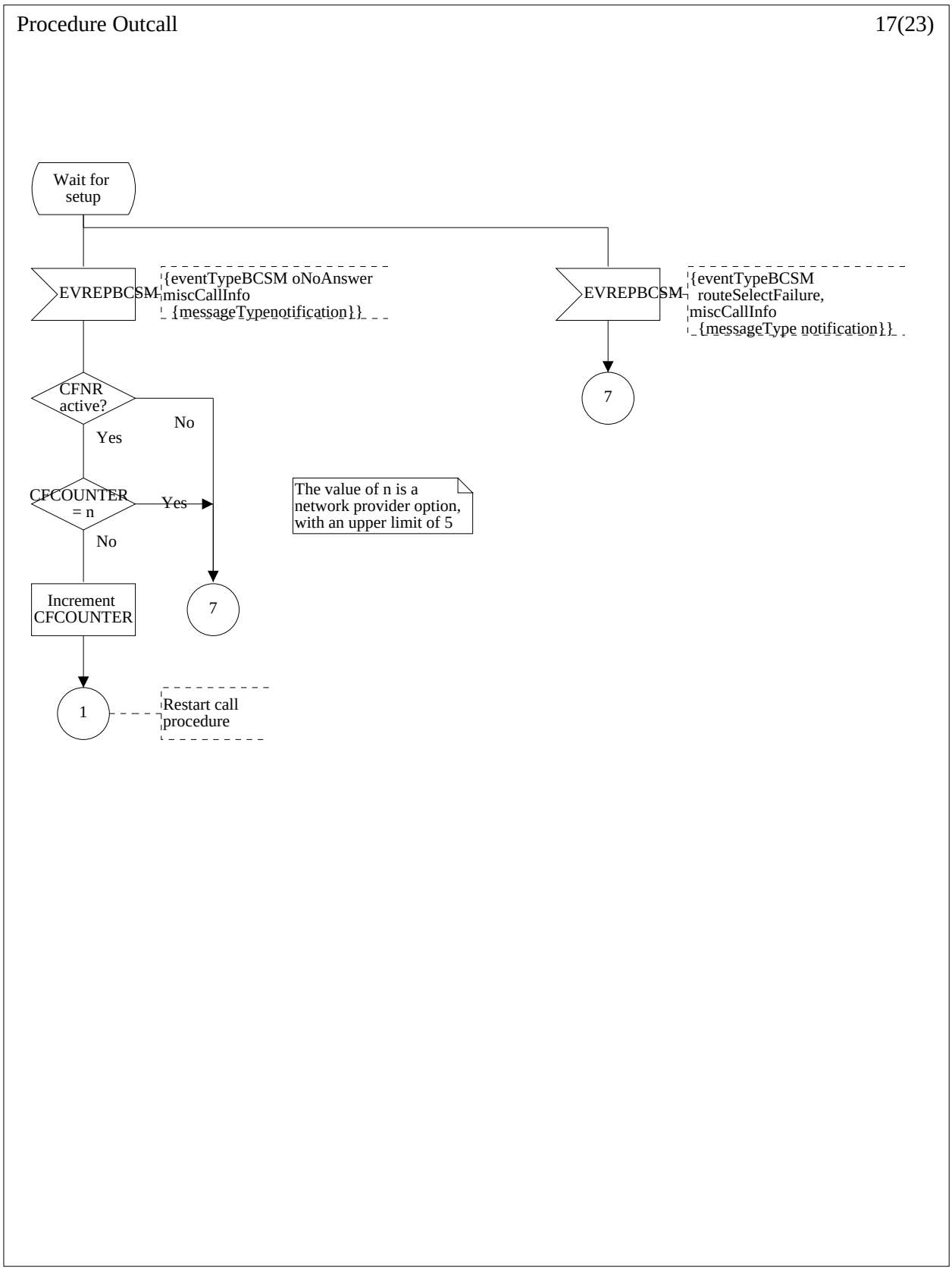


**Figure 14 (sheet 17 of 23): Outgoing UPT Call procedure**

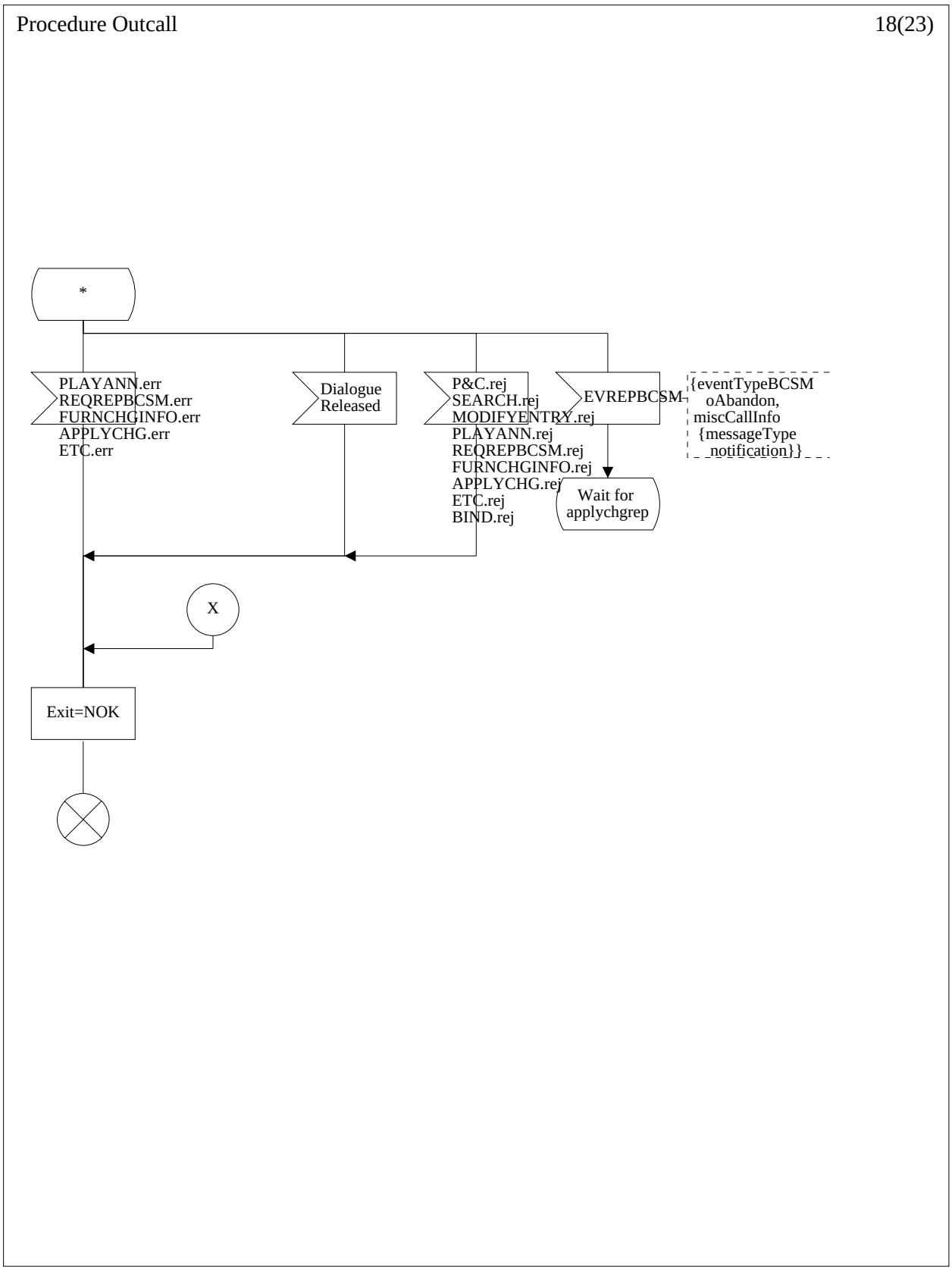


**Figure 14 (sheet 18 of 23): Outgoing UPT Call procedure**

Procedure Outcall 19(23)

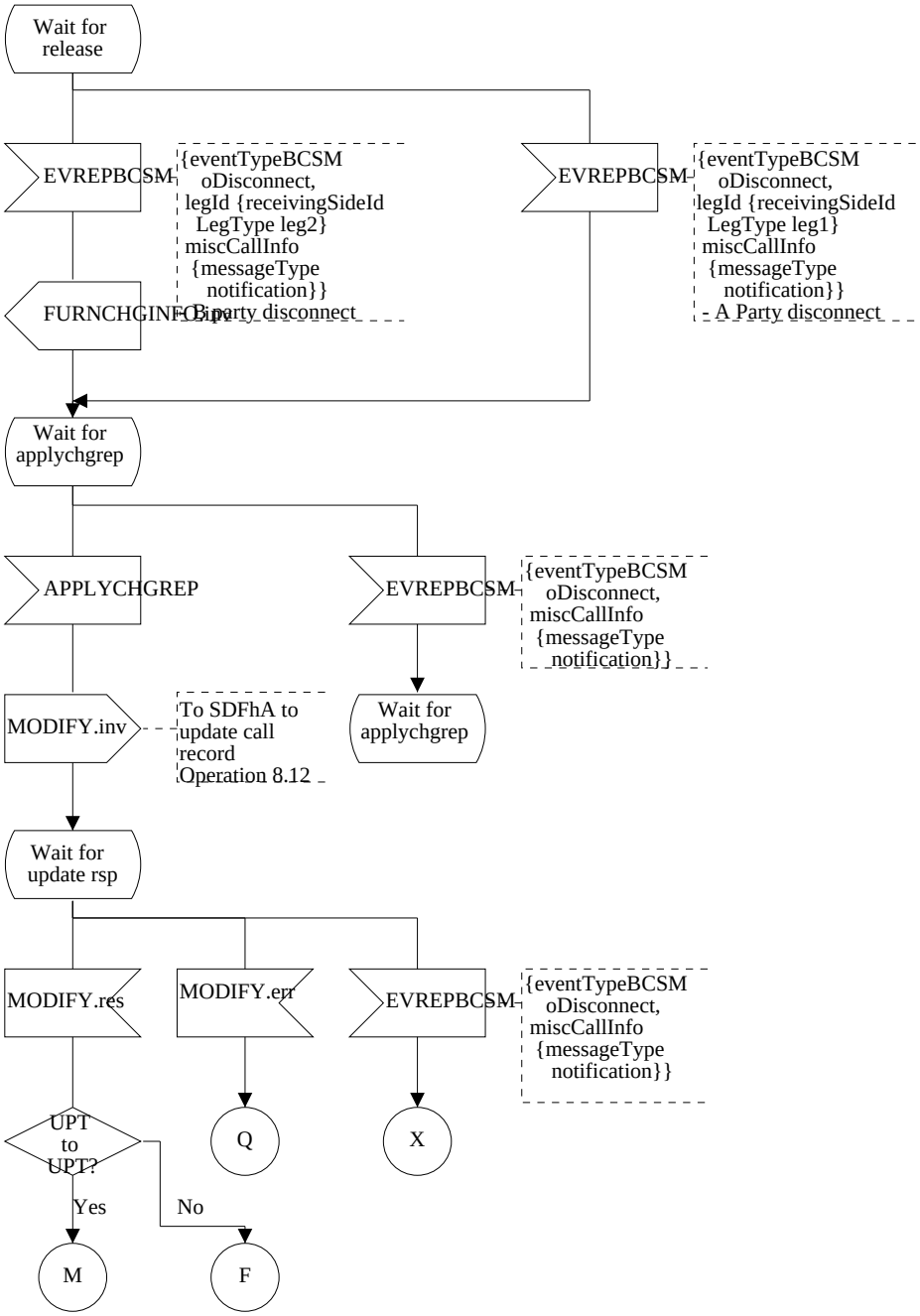


**Figure 14 (sheet 19 of 23): Outgoing UPT Call procedure**

Procedure Outcall 20(23)

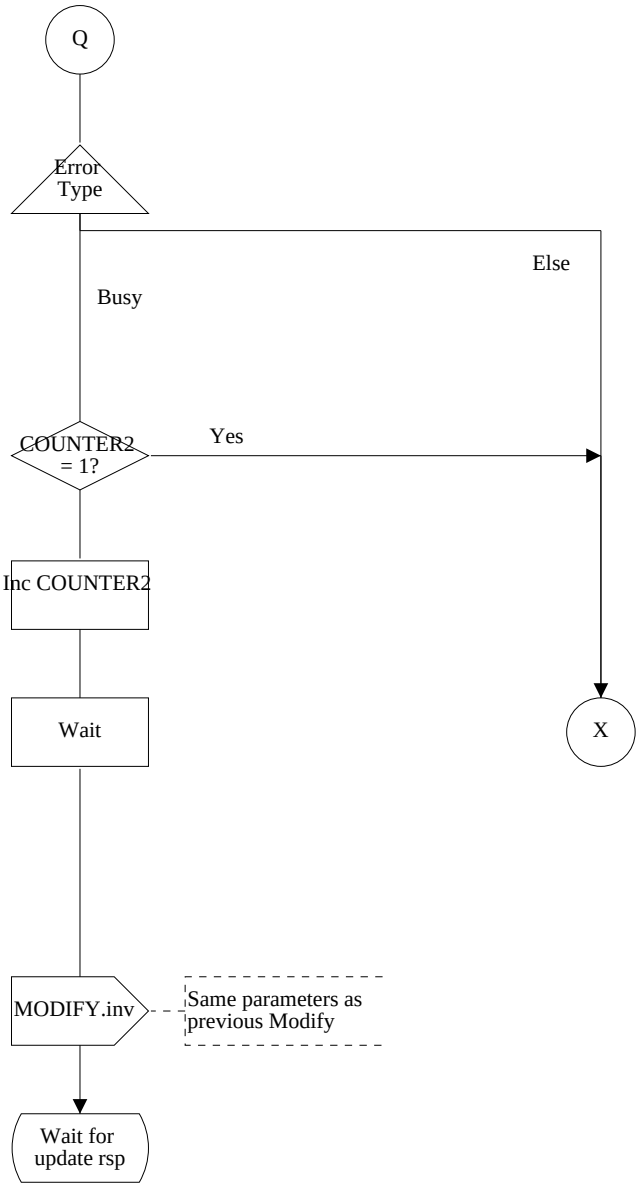


**Figure 14 (sheet 20 of 23): Outgoing UPT Call procedure**

Procedure Outcall 21(23)

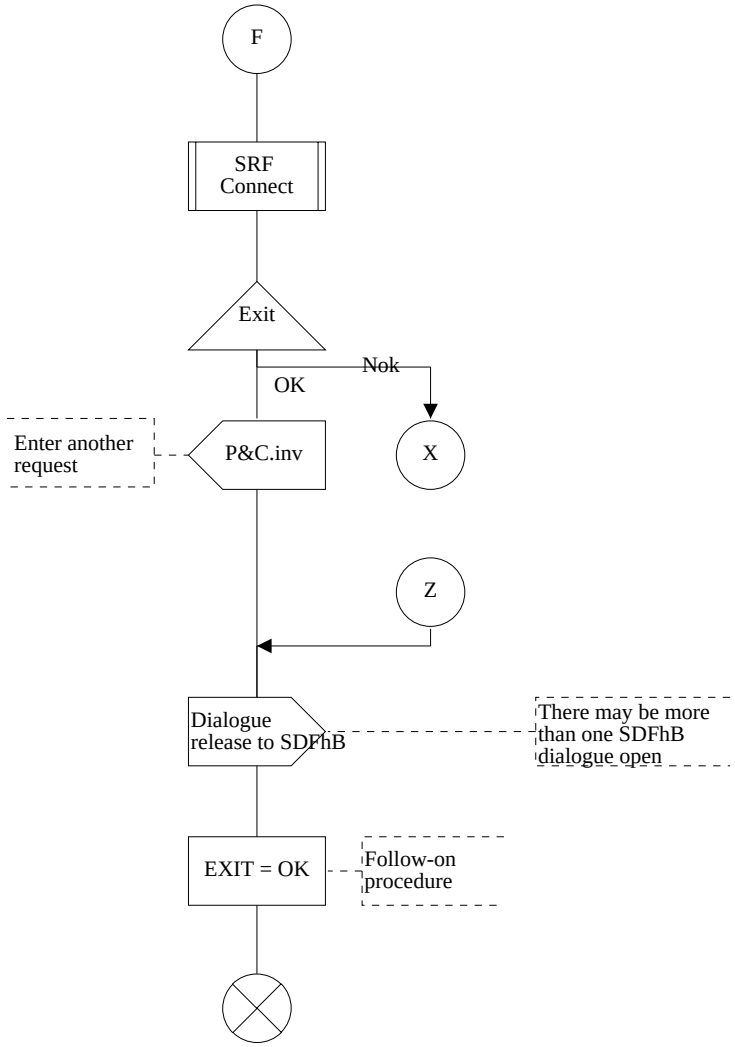


**Figure 14 (sheet 21 of 23): Outgoing UPT Call procedure**

Procedure Outcall 22(23)

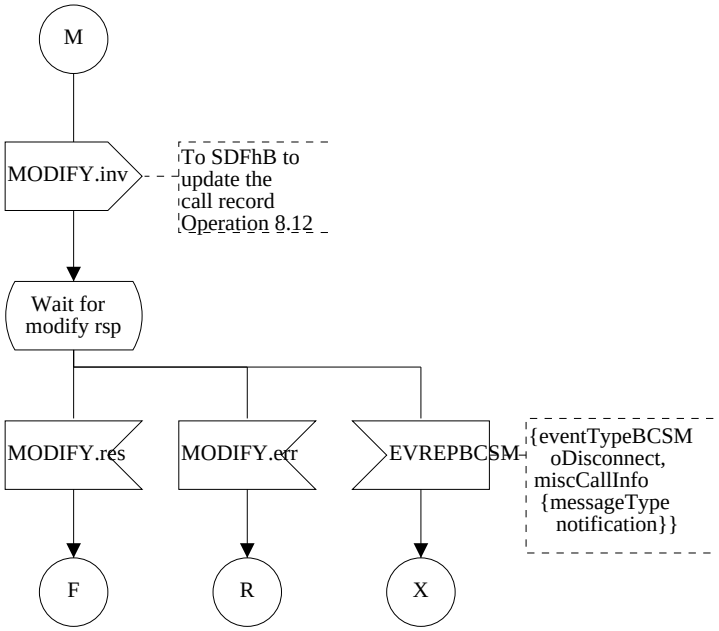


**Figure 14 (sheet 22 of 23): Outgoing UPT Call procedure**

Procedure Outcall 23(23)

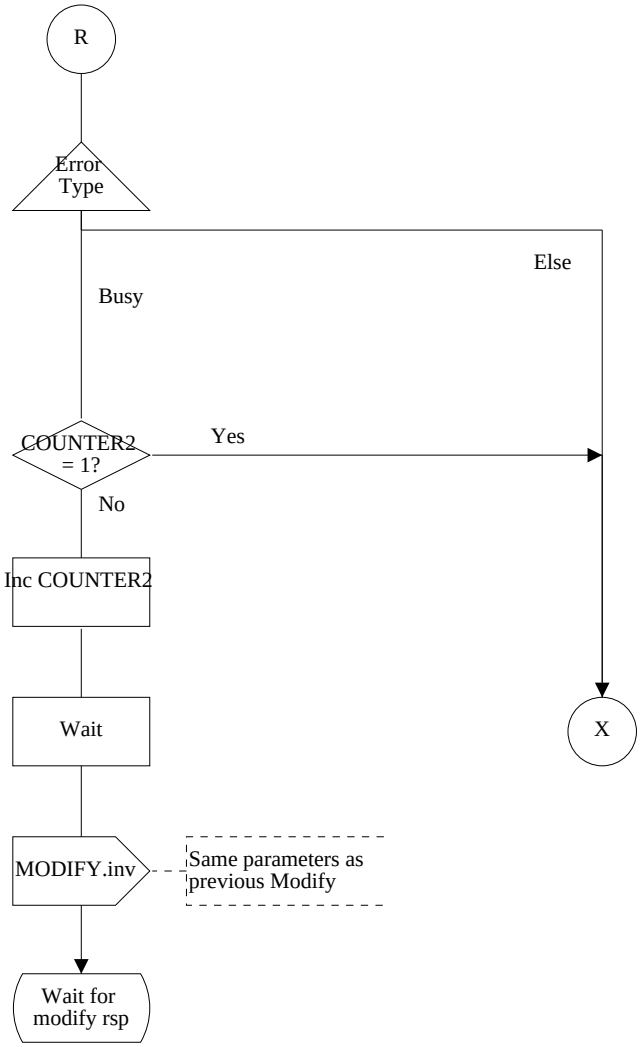


**Figure 14 (sheet 23 of 23): Outgoing UPT Call procedure**

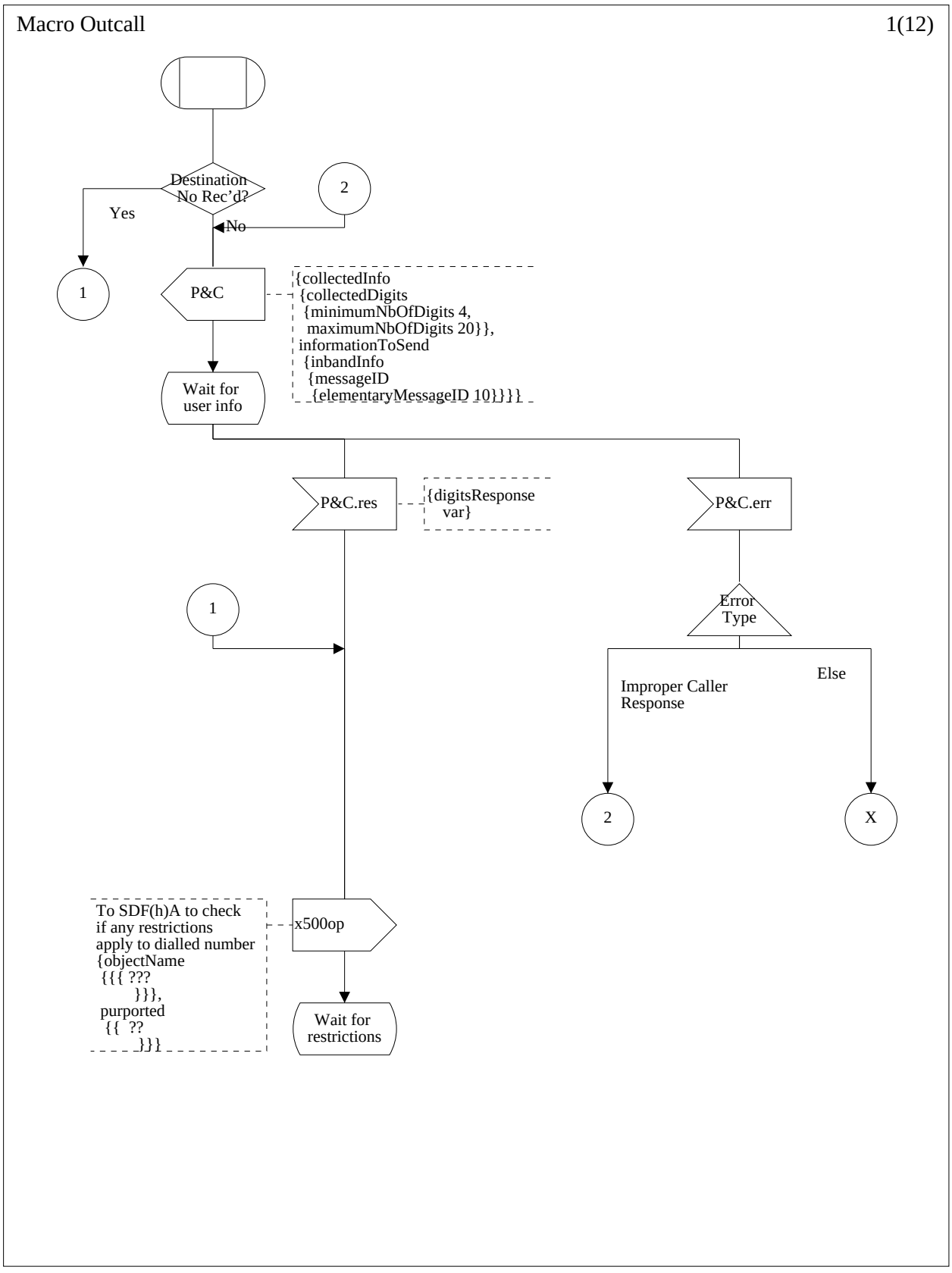


**Figure 15 (sheet 1 of 12): Outgoing UPT Call macro**

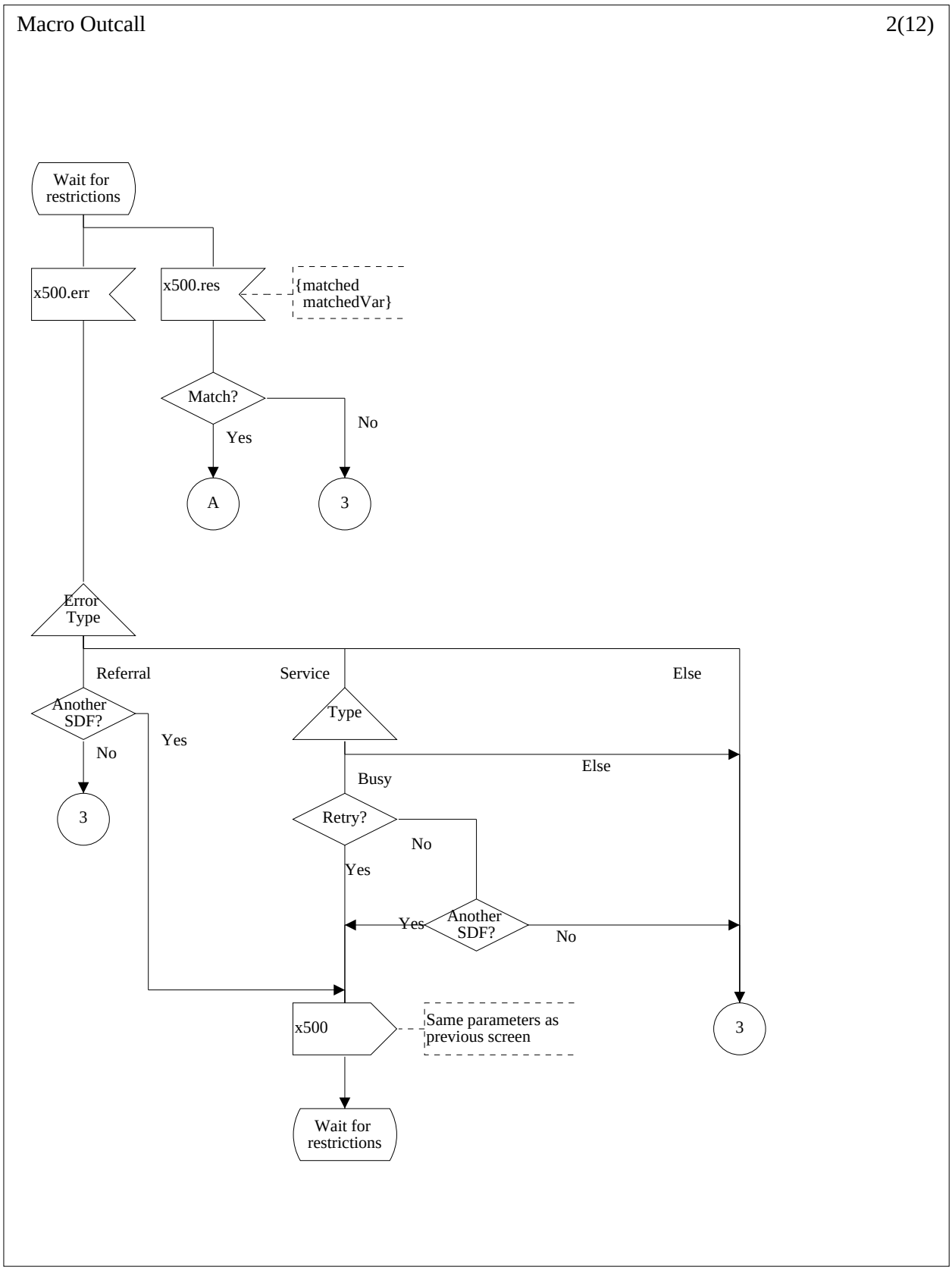


**Figure 15 (sheet 2 of 12): Outgoing UPT Call macro**

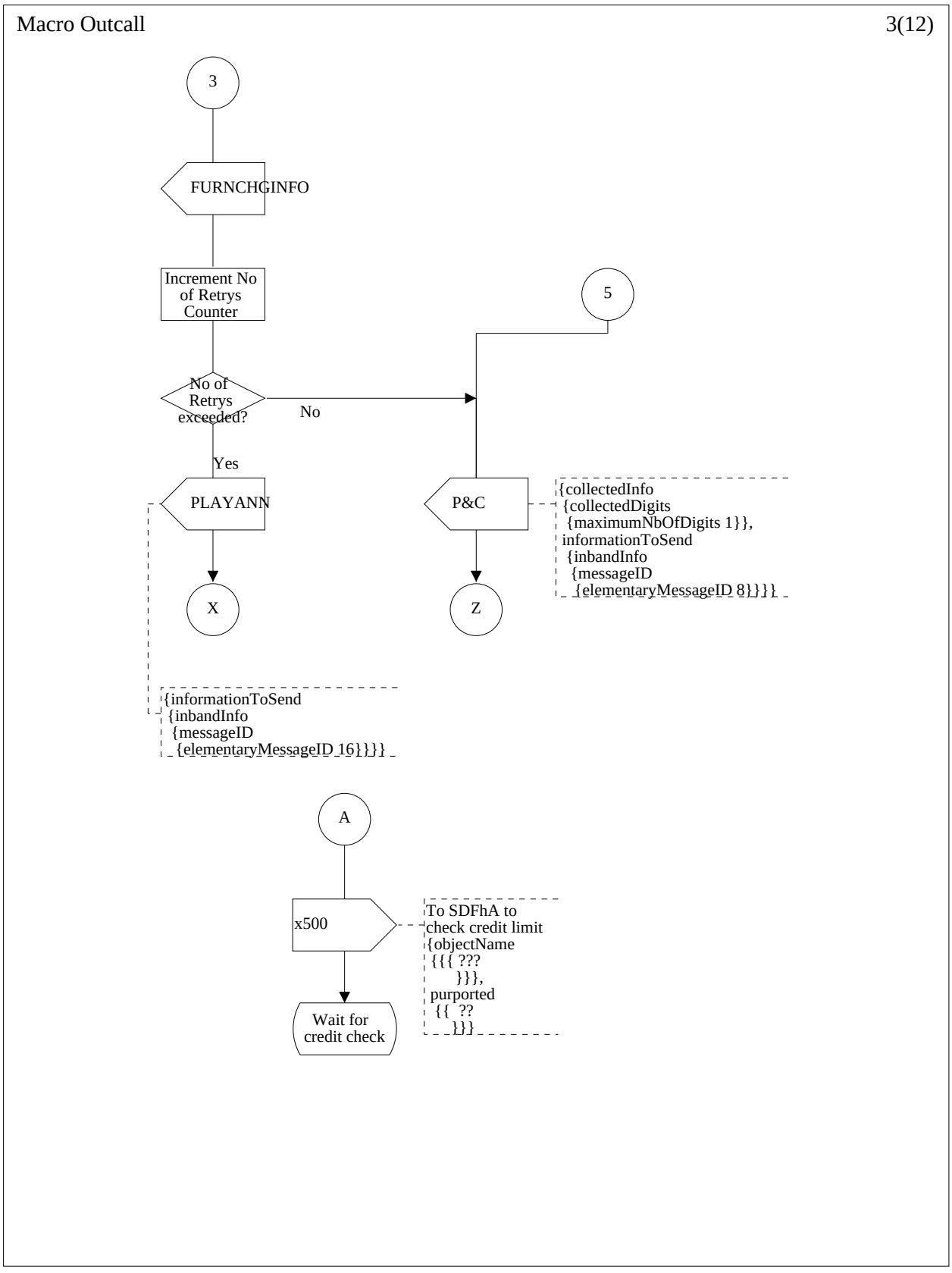


**Figure 15 (sheet 3 of 12): Outgoing UPT Call macro**

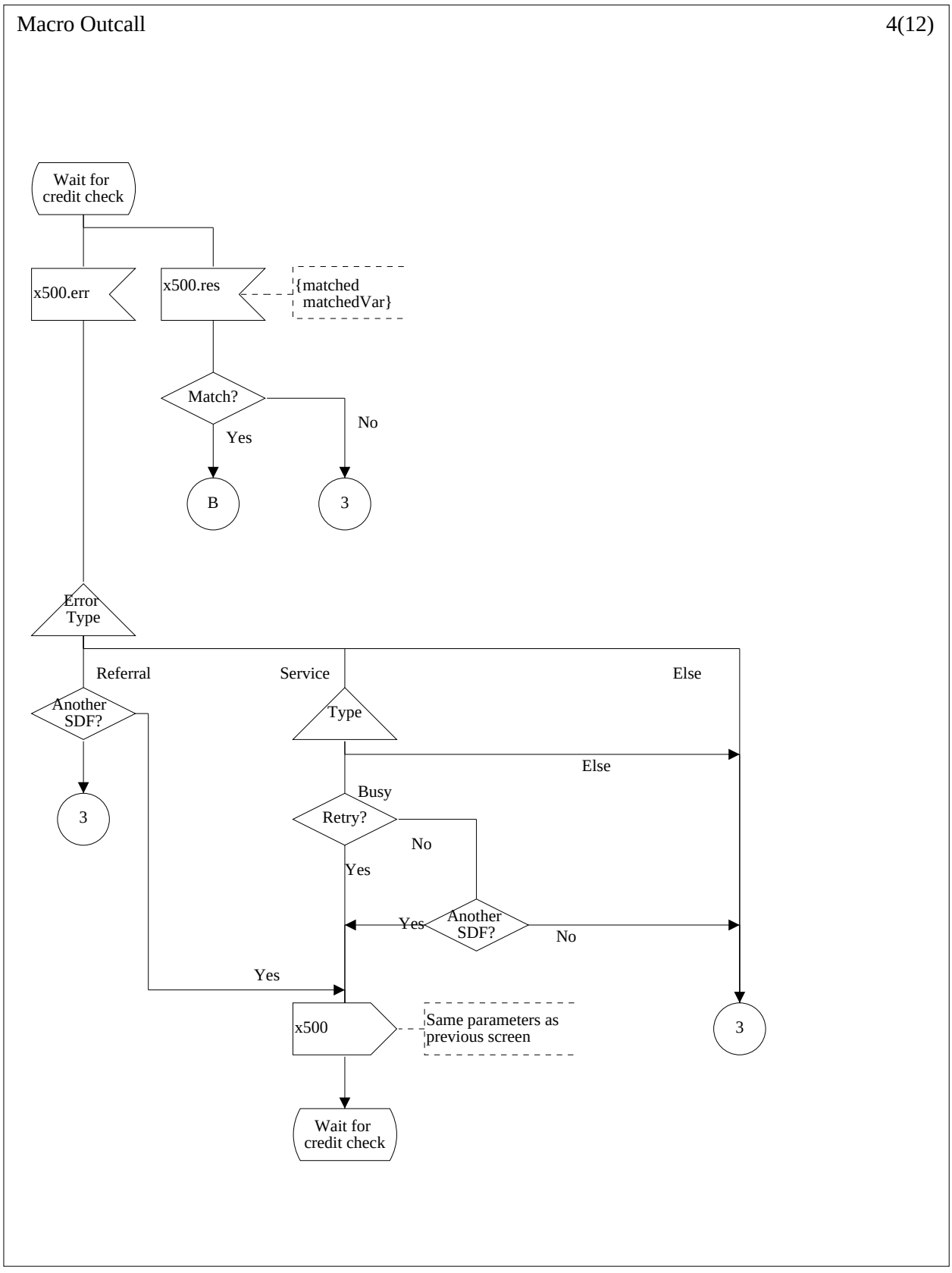


**Figure 15 (sheet 4 of 12): Outgoing UPT Call macro**

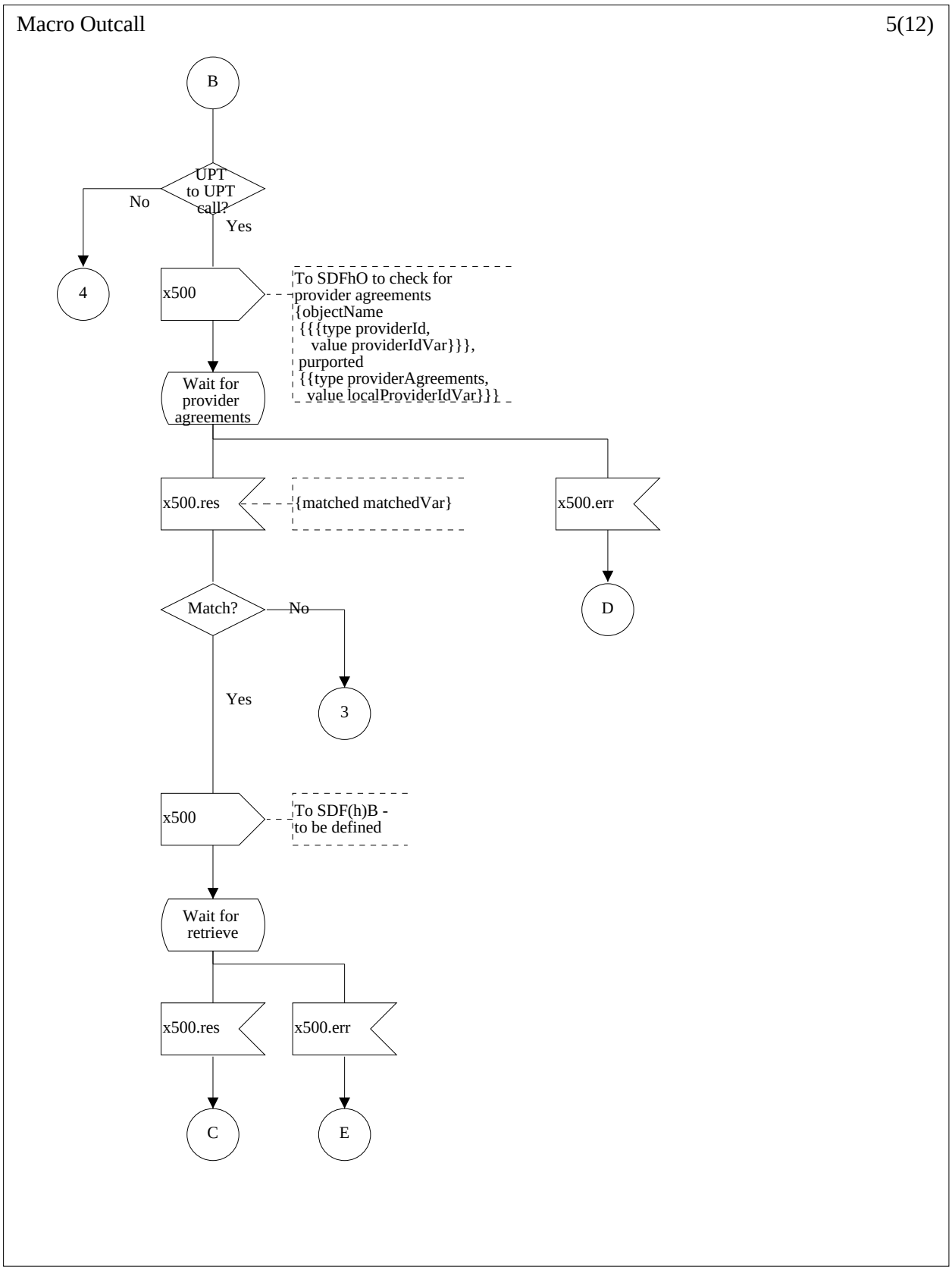


**Figure 15 (sheet 5 of 12): Outgoing UPT Call macro**

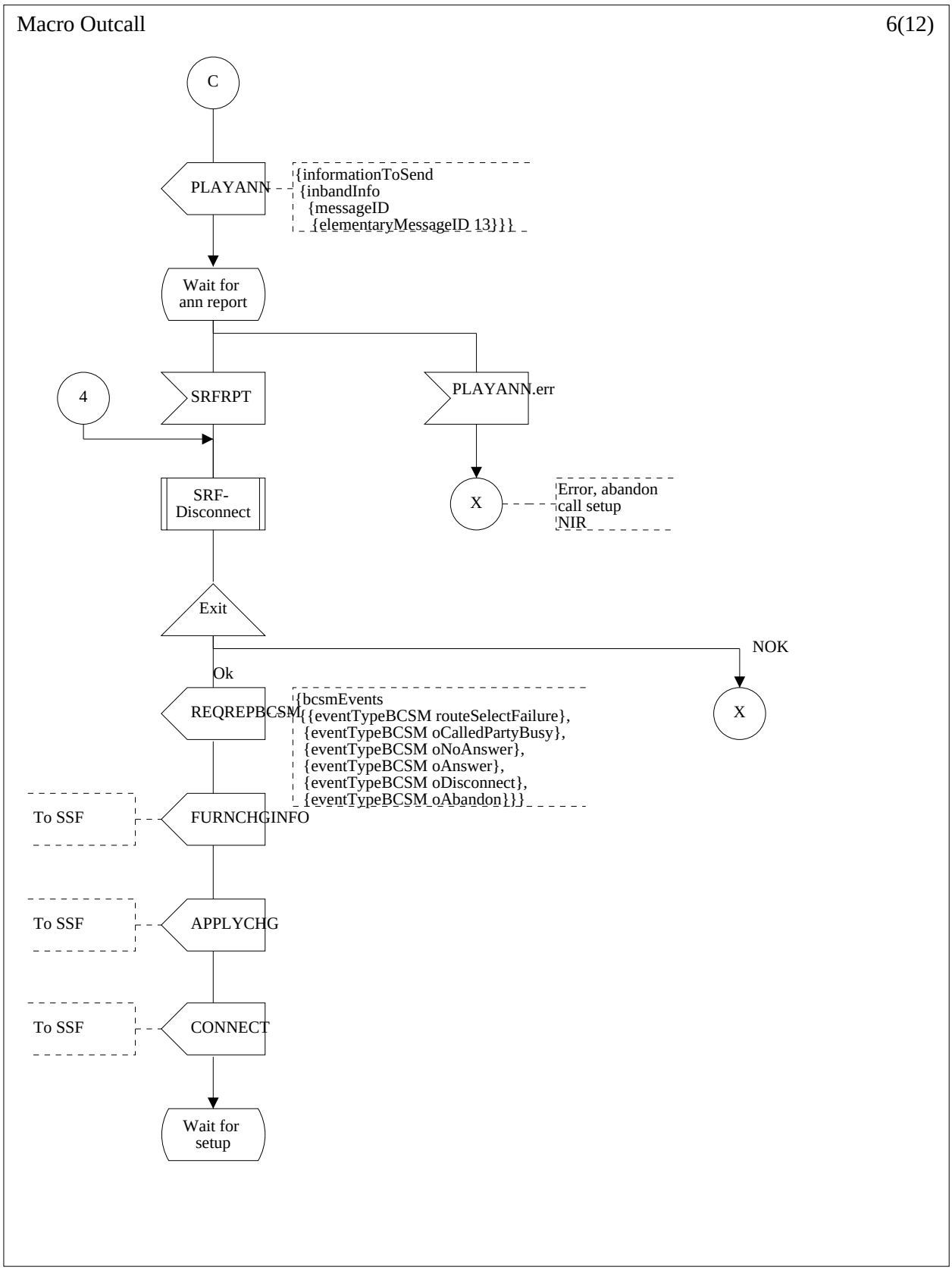


**Figure 15 (sheet 6 of 12): Outgoing UPT Call macro**

Macro Outcall 7(12)

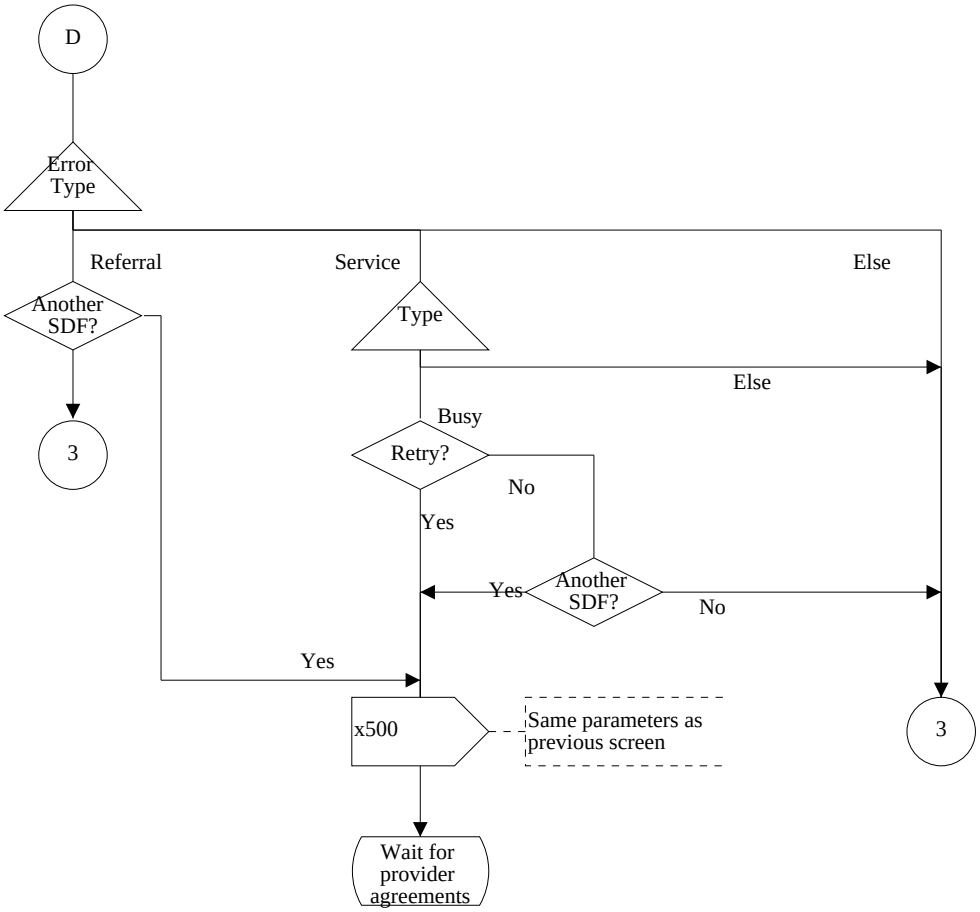


**Figure 15 (sheet 7 of 12): Outgoing UPT Call macro**

Procedure Incall 8(15)

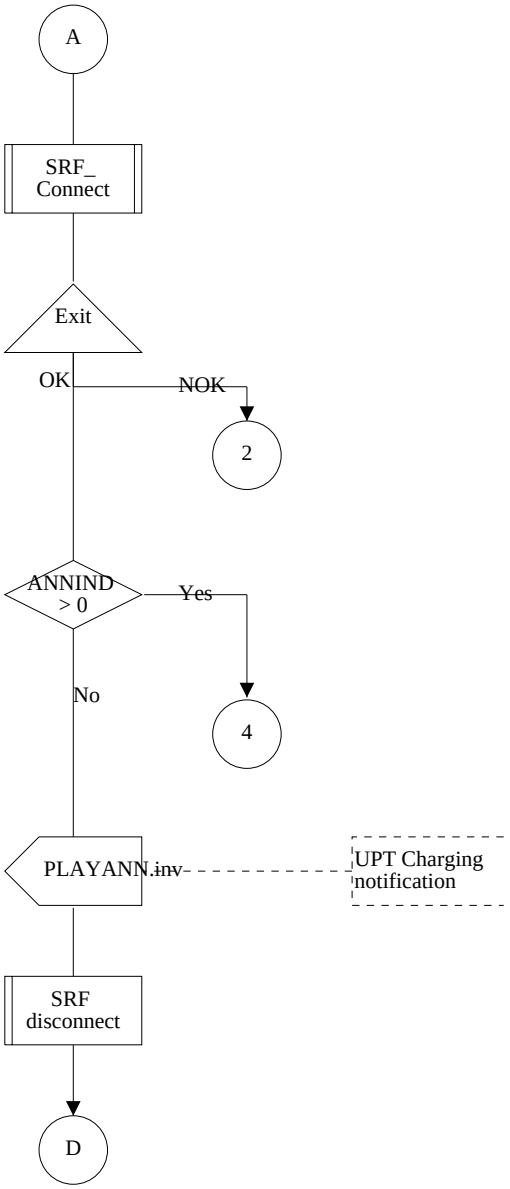


**Figure 15 (sheet 8 of 12): Outgoing UPT Call macro**

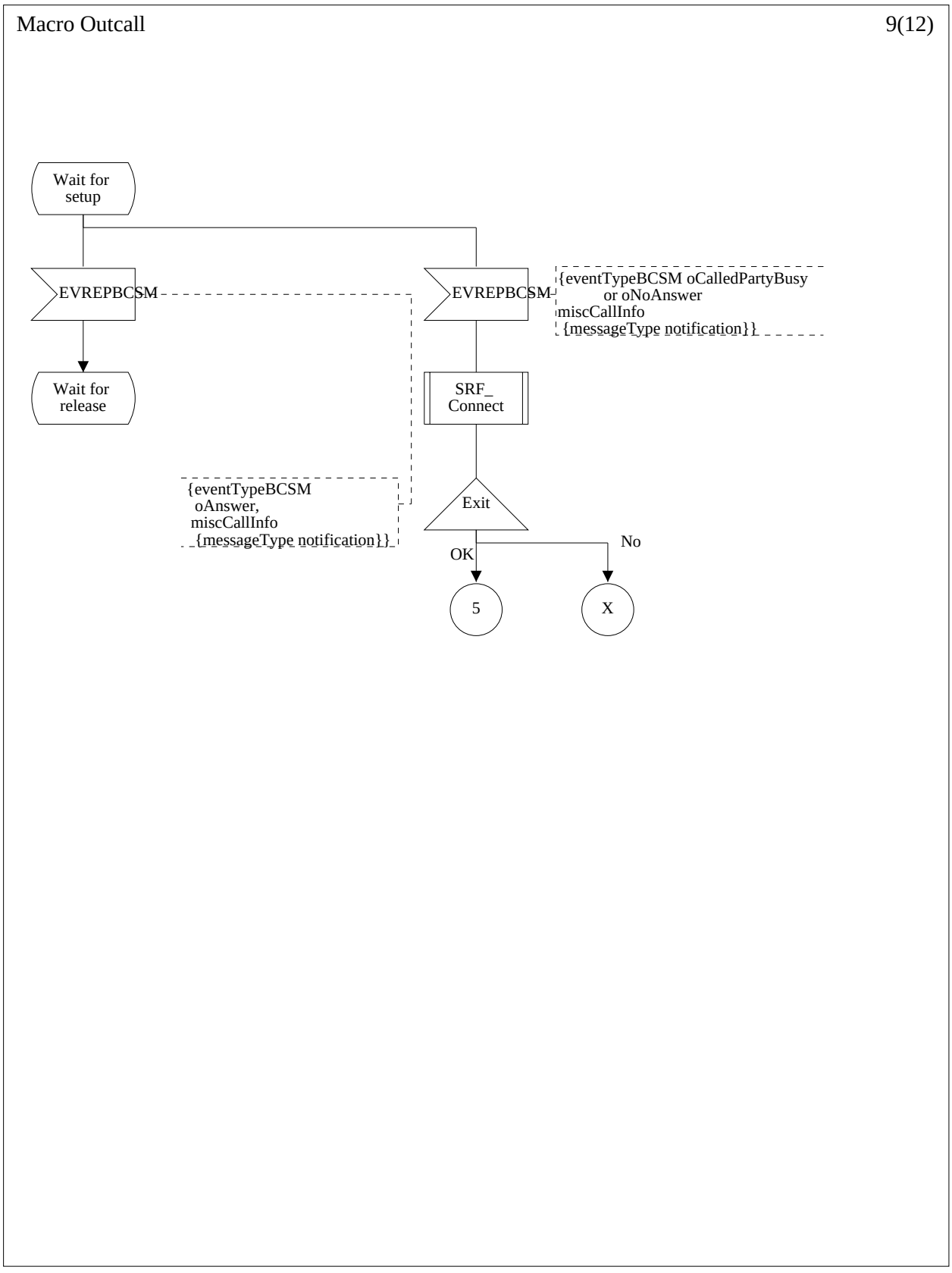


**Figure 15 (sheet 9 of 12): Outgoing UPT Call macro**

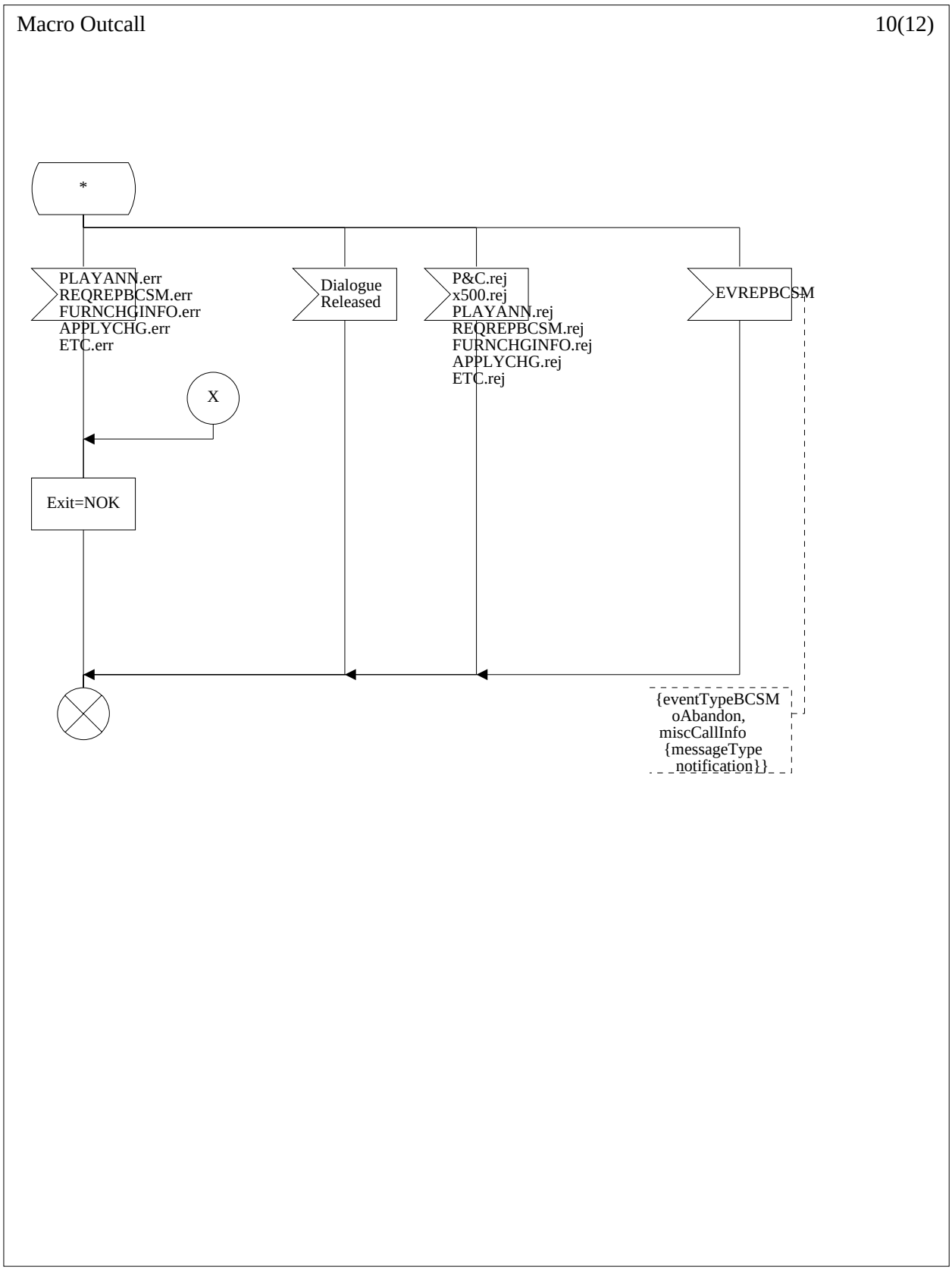


**Figure 15 (sheet 10 of 12): Outgoing UPT Call macro**

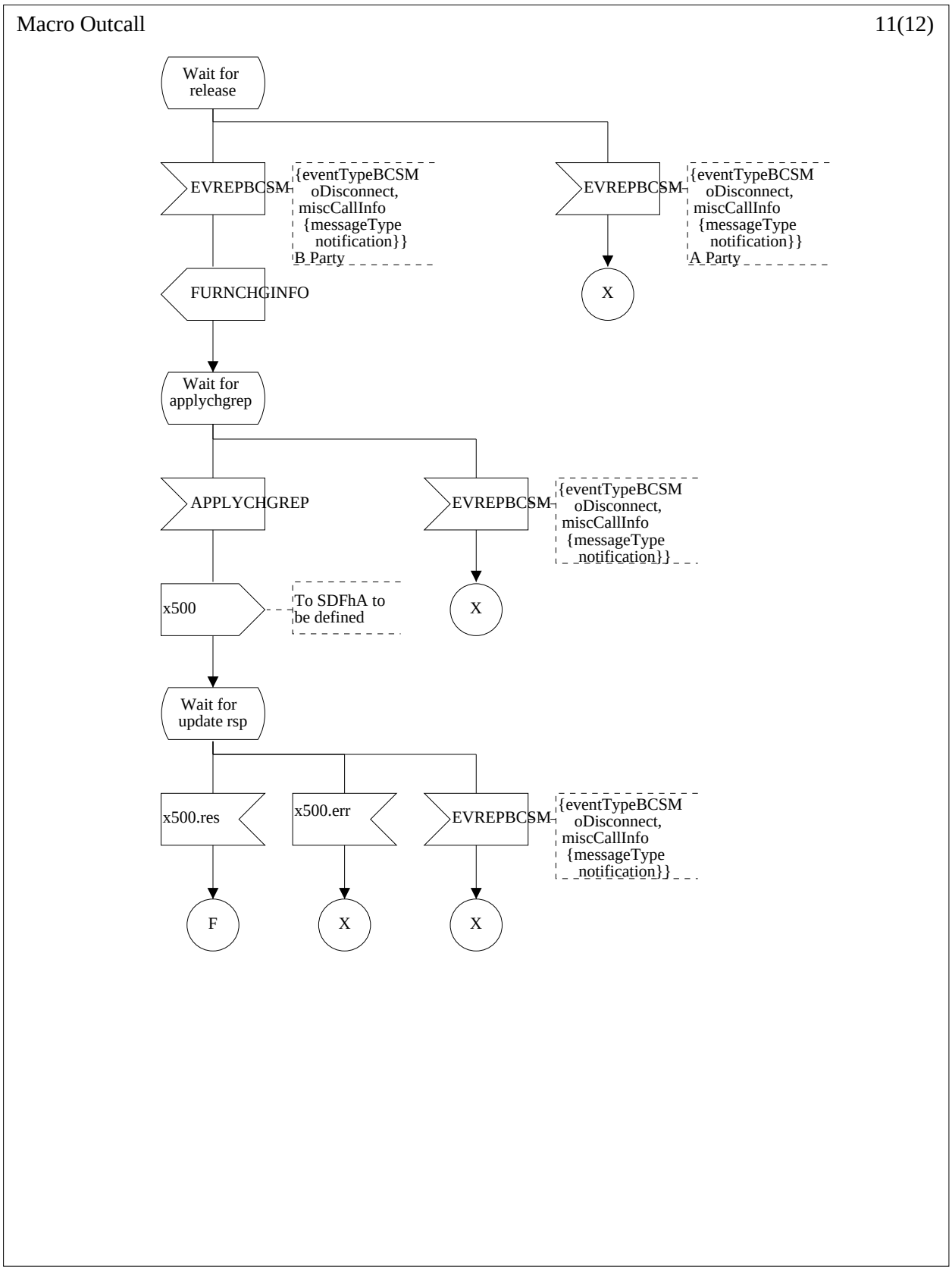


**Figure 15 (sheet 11 of 12): Outgoing UPT Call macro**

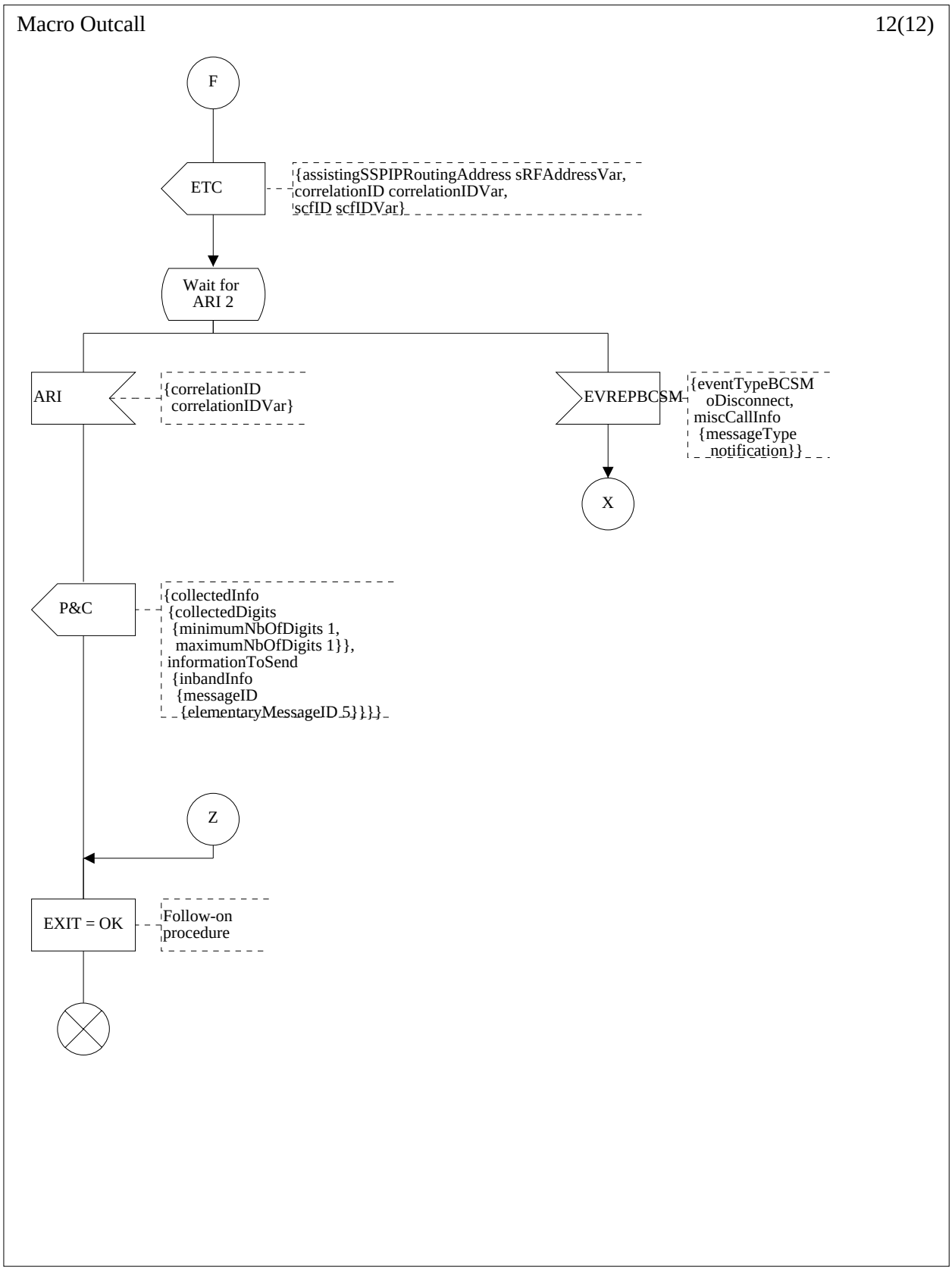


**Figure 15 (sheet 12 of 12): Outgoing UPT Call macro**

### 7.4.2 Incoming UPT Call

#### 7.4.2.1 General

This subclause only describes the case of a non-UPT user calling a UPT user. The case of UPT user calling another UPT user is covered in the previous subclause 7.4.1. It is assumed that a call to a UPT user is always controlled from the "first" network with UPT capabilities. This network may be either the originating network or the called user's home network or another network.

#### 7.4.2.2 Detailed Procedure

The procedure for incoming call handling for UPT calls is described in figure 16. The related macro is shown in figure 17. The incoming call procedure is called by the process UPT_SLP as described in subclause 7.1.3. of this specification.

If Redirection Information is provided in the INITIALDP operation, the Call Forwarding Counter (CFCOUNTER) is set to the same value as the Redirection Counter (see ETS 300 356-15 [1] for a description of this ISUP information element), otherwise the Call Forwarding Counter (CFCOUNTER) is set to zero.

**Provider agreements**

The purpose of the first SEARCH operation is to check the SDFo to determine whether agreement exists between the local service provider and the called user's home provider for establishing incoming calls. The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the SSF is instructed to release the call.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation (.err) are described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt to retry the same SDF, after a time delay. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

   2) For all other error types returned by the SDF, the call handling procedure is terminated and the call released.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDF. If the result is a match (i.e. agreement exists between the local service provider and the called UPT users home provider for establishing incoming calls) then the procedure will continue as described below (Screening of the home database). If there is no agreement to establish incoming calls, the call will be treated as a normal call and the procedure continues with call set-up as described in the subclause below.

**Screening of the home database**

Having established that agreements exist between the local and home service providers, the SCF can now proceed to check the called UPT user's home database to retrieve the current location of the called user. The dialogue is opened with an "empty" Directory BIND operation (i.e. the Credentials parameter will not be present), the outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or BIND.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the SSF is instructed to release the call.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure.

c) Operation error returned (BIND): The possible error causes returned in the BIND operation (.err) are described in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the SSF is instructed to release the call.

d) Successful result (SEARCH): This means that the operation has been successfully executed by the SDF. The dialogue with the SDFhB can now proceed.

To retrieve the location of the called user the SCF will invoke a SEARCH operation (as defined in subclause 8.11). The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the SSF is instructed to release the call.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure.

c) Operation error returned (SEARCH): The possible error causes returned in the SEARCH operation (.err) are described in ETS 300 374-5[3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt to retry the same SDF, after a time delay. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

   2) For all other error types returned by the SDF, the call handling procedure is terminated and the call released.

d) Successful result (SEARCH): This means that the operation has been successfully executed by the SDF. The result is one or more valid routeing addresses. The SCF will select the routeing address based on the following priority:

   1) The routeing address for the Call Forwarding Unconditional service, if this service is active.

   2) If the Call Forwarding Unconditional service is not active but the registration is still valid, the routeing address used will be the registration address.

   3) If the Call Forwarding Unconditional service is not active but the Variable Routeing service is active, the address used will depend on the time or on the calling user.

   4) Default registration address if none of the above criteria apply.

**Default Charging Reference Point**

If the call is being forwarded the retrieval of the default charging reference point is not required, the charging for the forwarded leg of the call is a matter for the original called user and is not described here.

To retrieve the default charging reference the SDFhB is interrogated by the SCF using a SEARCH operation (as defined in subclause 8.16) the outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the Release procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation result (.err) are described in ETS 300 374-5 [3]:

1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

2) For all other error types returned by the SDF, the procedure is terminated with Exit = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDFhB and the default charging reference point retrieved.

If split charging (see ETR 055-3 [7]) is not to be applied the procedure continues with the UPT charging notification subclause below. If split charging is to be applied then the procedure continues with the Credit Limit Check.

**Credit Limit Check**

The SCF will also check the called users Credit Limit to determine that there is sufficient credit available to receive the call. This check will not be repeated for subsequent call set-up attempts (i.e. UPT to UPT Call Forwarding has occurred). The SCF sends a SEARCH operation, the outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the SSF is instructed to release the call.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation (.err) are described in ETS 300 374-5 [3]:

1) In the case of "Service Error" type Busy the SCF can make a further attempt to retry the same SDF, after a time delay. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

2) For all other error types returned by the SDF, the call handling procedure is terminated and the call released.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDF. If credit is available to receive the call the procedure will continue as described below. If there is no credit available the call handling procedures are terminated and the SSF is instructed to release the call.

**UPT Charging Notification**

The SCF instructs the SSF to establish a connection with an SRF. This is done with the SRF_Connect procedure described in subclause 7.2.4. If the above procedure was completed successfully, the SCF then instructs the SRF to play an announcement to the calling user to inform that UPT charging is applicable by invoking the PLAYANN operation. This notification will not be repeated for subsequent call set-up attempts for this call (i.e. UPT to UPT call forwarding has occurred and the calling user was notified that UPT charging was applicable from a previous attempt to set-up this call). Following the instruction for the PLAYANN operation the macro SRF_Disconnect is called, this macro will handle the operation errors and the disconnection of the SRF.

**Retrieve supplementary service information**

The purpose of this part of the procedure is to query the called UPT users home database (SDFhB) for the status of supplementary services and for those call forwarding services which are active, retrieve the relevant conditional forwarding parameters (e.g. No Reply Condition Timer). The SCF will invoke a SEARCH operation (see subclause 8.13) for this. The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or SEARCH.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3.) is invoked.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure.

c) Operation error returned (SEARCH.err): The possible error causes returned in the SEARCH operation result (.err) are described in ETS 300 374-5 [3]:

   1) In the case of "Service Error" type Busy the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries (COUNTER2) is exceeded, the RELEASE procedure is performed.

   2) For all other error types returned by the SDF, the procedure is terminated with Exit = NOK.

d) Successful result (SEARCH.res): This means that the operation has been successfully executed by the SDFhB. The call forwarding parameters retrieved (if any) are stored for future use. The procedure continues as described below.

**Call set-up**

On successful completion of the above procedures the SCF can now instruct the SSF to set-up the call, several operations are invoked for this purpose:

a) FURNCHGINFO: This requests the SSF to generate call record information for the following event.

b) APPLYCHG: The purpose of the operation is to request the SSF to report back to the SCF when a charging related event has been detected (i.e. if UPT user is charged).

c) REQREPBCSM: This requests the SSF to monitor for a call-related event (e.g. busy, no answer, release etc.) and report back to the SCF when the event has been detected. If a No Reply Condition Timer value was retrieved from the called UPT users home database (SDFhB) when searched for supplementary service information, the value will be provided with this operation.

d) CONNECT: This instructs the SSF to set-up the call (i.e. generate the IAM). If any redirection information was provided in the INITIALDP operation the information will be returned in this operation, the redirection counter may have been updated as a result of further call forwarding. If no redirection information was provided in the INITIALDP operation but call forwarding has occurred, then the redirection information will be constructed by the SCF. Redirection information will not be provided if no call forwarding has occurred. The SSF will determine how to handle this information.

The outcome of this procedure will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or <operation_name>.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-1 [2]. Regardless of the reason for the release, the call handling procedure is terminated and the SSF is instructed to release the call.

b) Calling party released: The SCF is notified of the release by the EVREPBCSM operation from the SSF. The SCF will terminate the call handling procedure.

c) Called party busy: This state will be reported to the SCF by the SSF returning the operation EVREPBCSM. There are three possible outcomes:

   1) If call forwarding on busy service is not active, the call handling procedure is terminated and the release procedure (as described in subclause 7.2.3) is invoked.

   2) If call forwarding on busy service is active but the Call Forwarding Counter (CFCOUNTER) has exceeded the network redirection limit (note that the upper limit for this counter is 5), the call handling procedure is terminated and the release procedure (as described in subclause 7.2.3) is invoked.

3) If call forwarding on busy service is active and the Call Forwarding Counter (CFCOUNTER) has not exceeded the network redirection limit, the call can be forwarded. The Call Forwarding Counter (CFCOUNTER) is incremented and the outgoing call procedure restarted.

d) Called party no answer: This state will be reported to the SCF by the SSF returning the operation EVREPBCSM. There are three possible outcomes:

1) If call forwarding on no reply service is not active, the call handling procedure is terminated and the release procedure (as described in subclause 7.2.3) is invoked.

2) If call forwarding on no reply service is active but the Call Forwarding Counter (CFCOUNTER) has exceeded the network redirection limit (note that the upper limit for this counter is 5), the call handling procedure is terminated and the release procedure (as described in subclause 7.2.3) is invoked.

3) If call forwarding on no reply service is active and the Call Forwarding Counter (CFCOUNTER) has not exceeded the network redirection limit, the call can be forwarded. The Call Forwarding Counter (CFCOUNTER) is incremented and the outgoing call procedure restarted.

e) Route select failure: This state will be reported to the SCF by the SSF returning the operation EVREPBCSM. This indicates that it was not possible to complete set up of the call due to either congestion, unsubscribed number or number blocked. The call handling procedure is terminated and the release procedure (as described in subclause 7.2.3) is invoked.

f) Operation error returned (FURNCHGINFO.err, APPLYCHG.err or CONNECT.err): The possible error causes returned in these operations are described in ETS 300 374-1 [2]. Regardless of the reason for the error, the call handling procedure is terminated and the SSF is instructed to release the call.

g) Calling Party Answers: The SCF is notified of this event by an EVREPBCSM operation.

**Call Release**

The SCF is informed of the release of the call by the EVREPBCSM operation. The SCF will then send a FURNCHGINFO operation to the SSF to update the call record. Once the APPLYCHGRPT containing the call record, is received the SCF sends a MODIFY operation (as defined in subclause 8.12) to the called parties SDFhB to store the call record. It should be noted that it is not possible to directly use the call record to modify the user credit as the SDF is not able to calculate the charge that corresponds to the call record. The outcome of this operation will be one of the following events:

a) Dialogue released by IN node (Dialogue_released or MODIFY.rej): The reasons for releasing the dialogue prematurely are described in subclause 7.1 and are further elaborated in ETS 300 374-5 [3]. Regardless of the reason for the release, the call handling procedure is terminated and the RELEASE procedure (as described in subclause 7.2.3) is invoked.

b) Operation error returned (MODIFY.err): The possible error causes returned in the MODIFY operation (.err) are described in ETS 300 374-5 [3]:

1) If the error is "Busy", the SCF can make a further attempt, after a time delay, to retry the same SDF. If the number of retries is exceeded (COUNTER2) the procedure is terminated. To avoid loss of the call record an implementation dependant operation could be performed at this stage.

2) For any other error the procedure is terminated. To avoid loss of the call record an implementation dependant operation could be performed at this stage.

c) Successful result (MODIFY.res): This confirms that the SDF has been successfully updated.

NOTE: If this incoming call was a result of UPT to UPT call forwarding from another IN network but the forwarded-to leg of the call could not be recognized as another UPT call by the original SCF (i.e. no provider agreements exist). Then the original called parties SDFhB may not be the same SDFhB accessed during this incoming call procedure, as this SDFhB will be the forwarded-to UPT user's SDFh. This may result in discrepancies in the charging information. The problem cannot be solved at present.

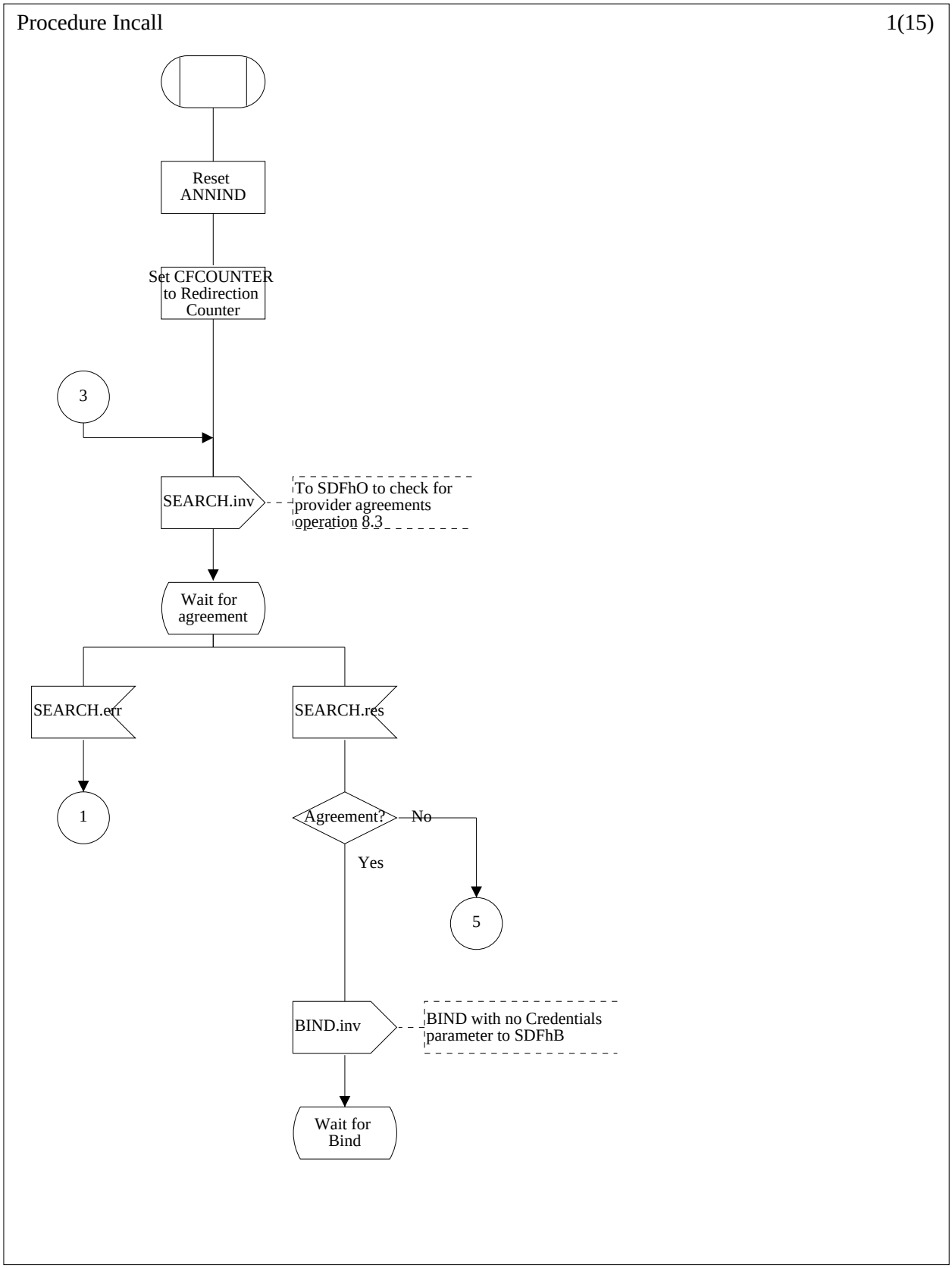


**Figure 16 (sheet 1 of 15): Incoming UPT Call procedures**

Procedure Incall 2(15)

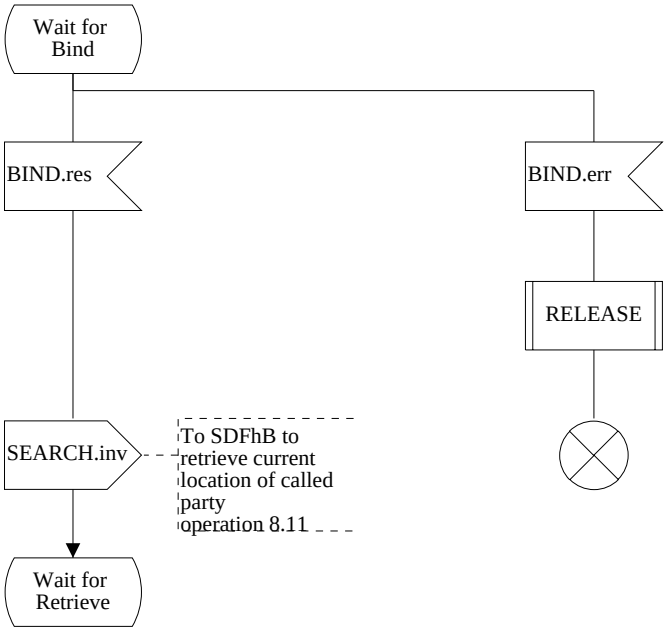


**Figure 16 (sheet 2 of 15): Incoming UPT Call procedures**

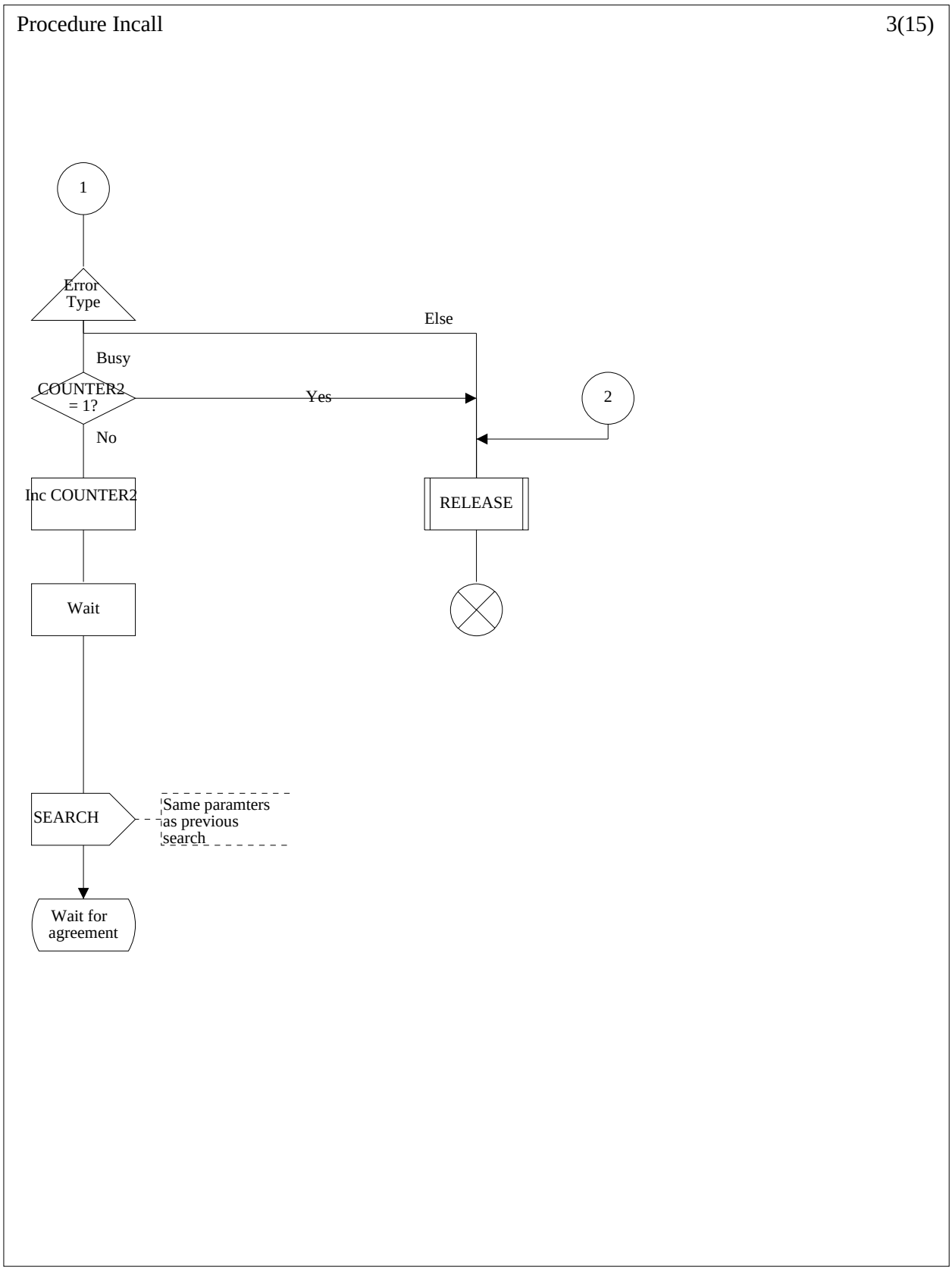


**Figure 16 (sheet 3 of 15): Incoming UPT Call procedures**

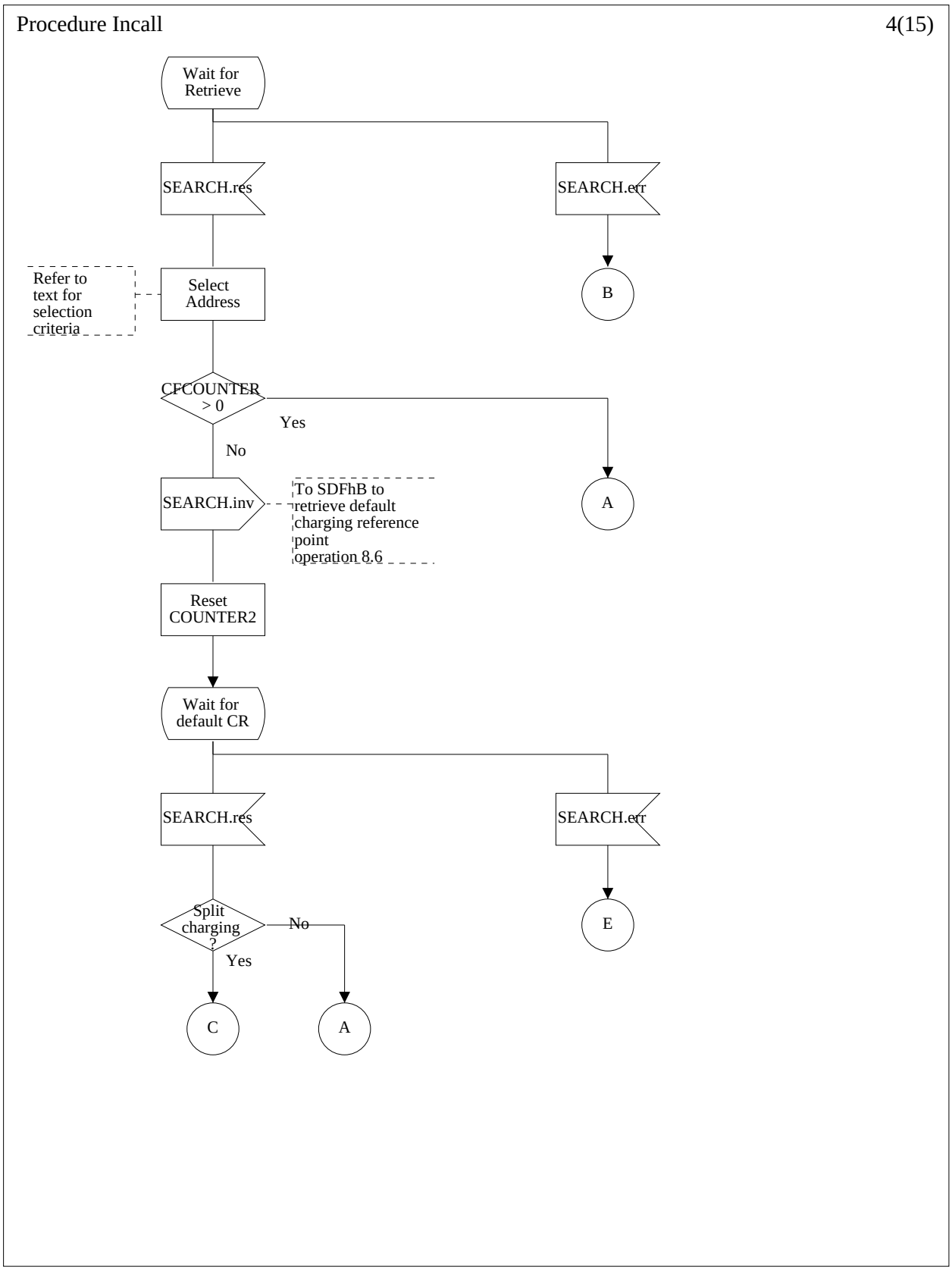


**Figure 16 (sheet 4 of 15): Incoming UPT Call procedures**

Procedure Incall 5(15)

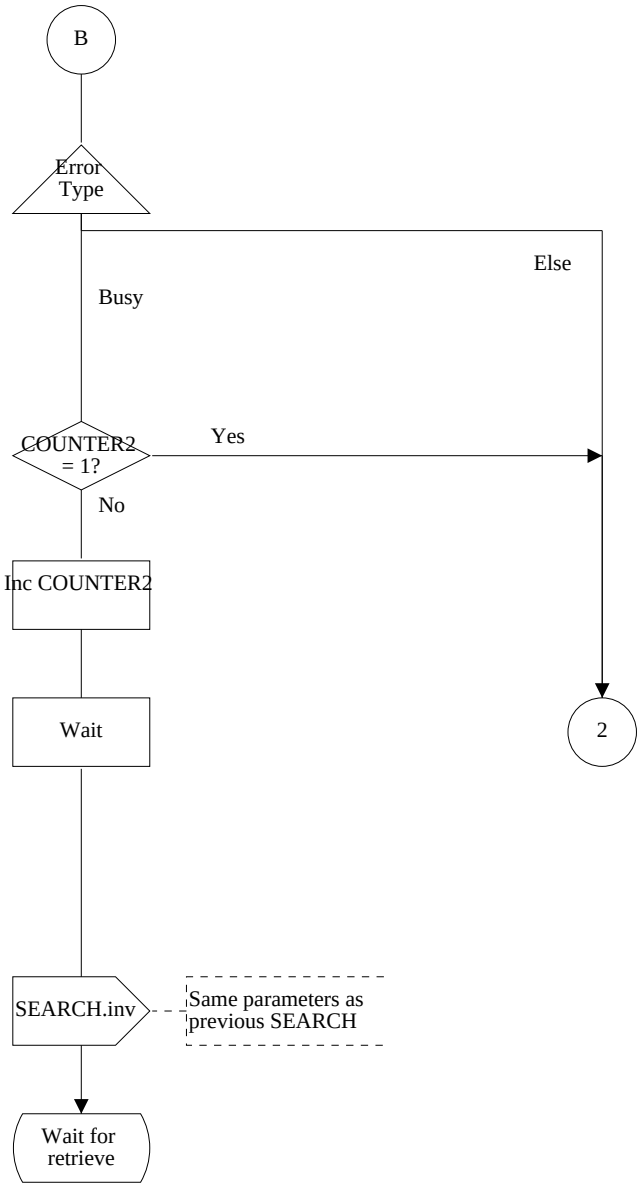


**Figure 16 (sheet 5 of 15): Incoming UPT Call procedures**

Procedure Incall 6(15)

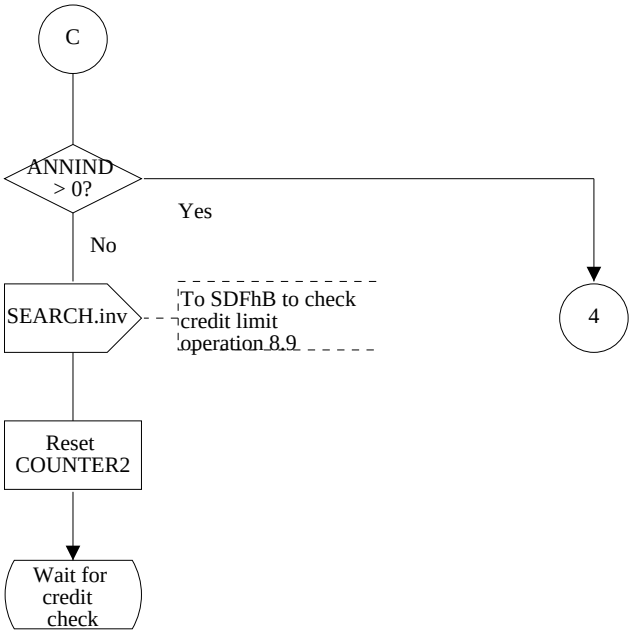


**Figure 16 (sheet 6 of 15): Incoming UPT Call procedures**

Procedure Incall 7(15)

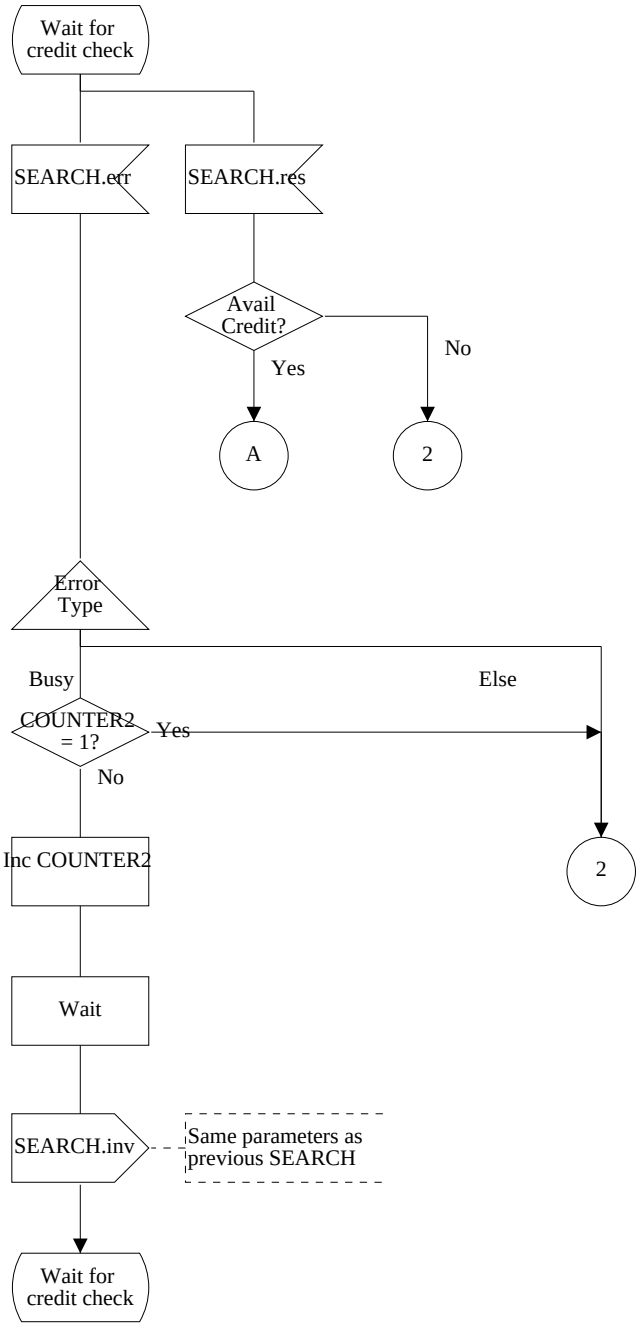


**Figure 16 (sheet 7 of 15): Incoming UPT Call procedures**

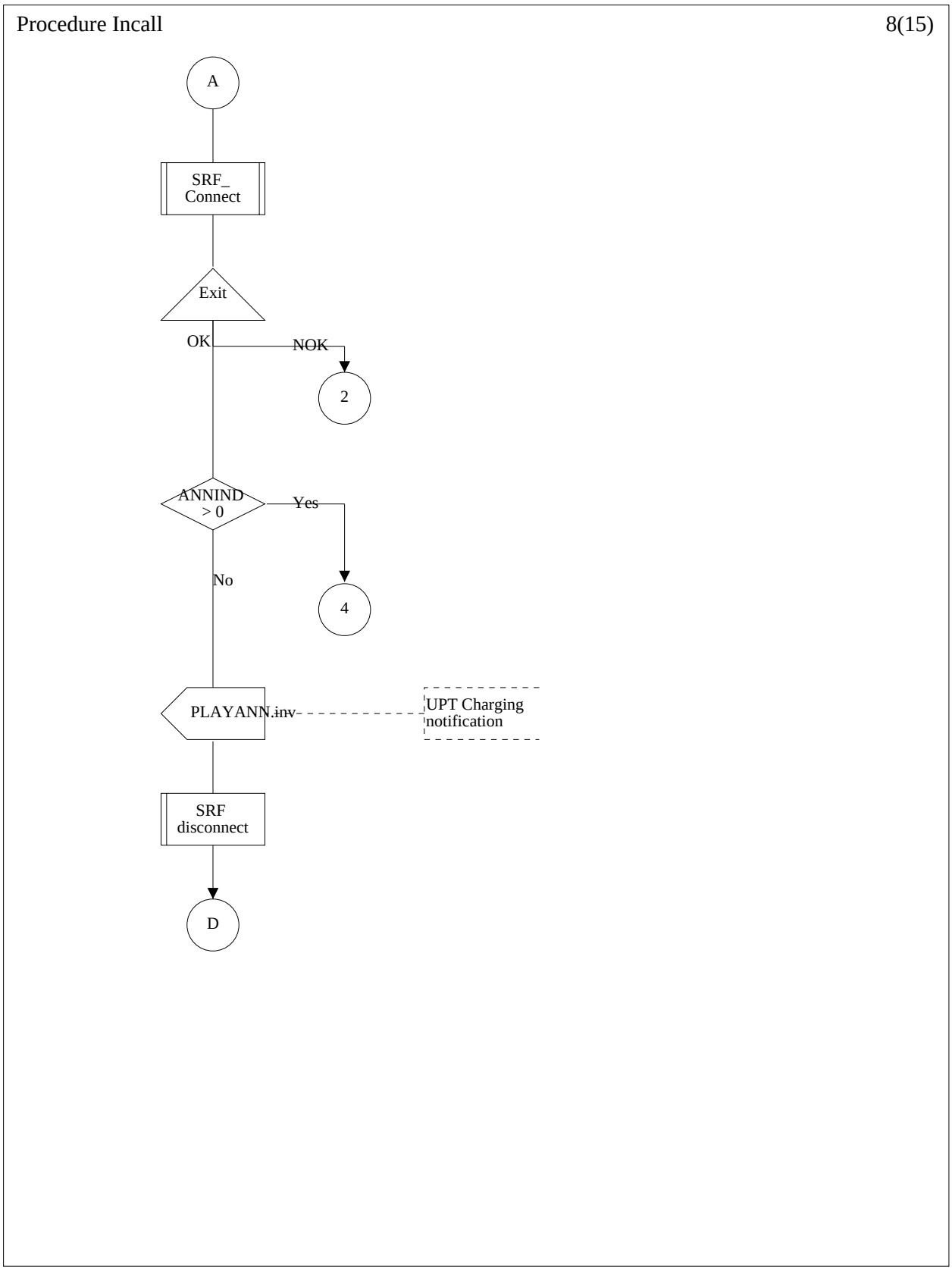


**Figure 16 (sheet 8 of 15): Incoming UPT Call procedures**

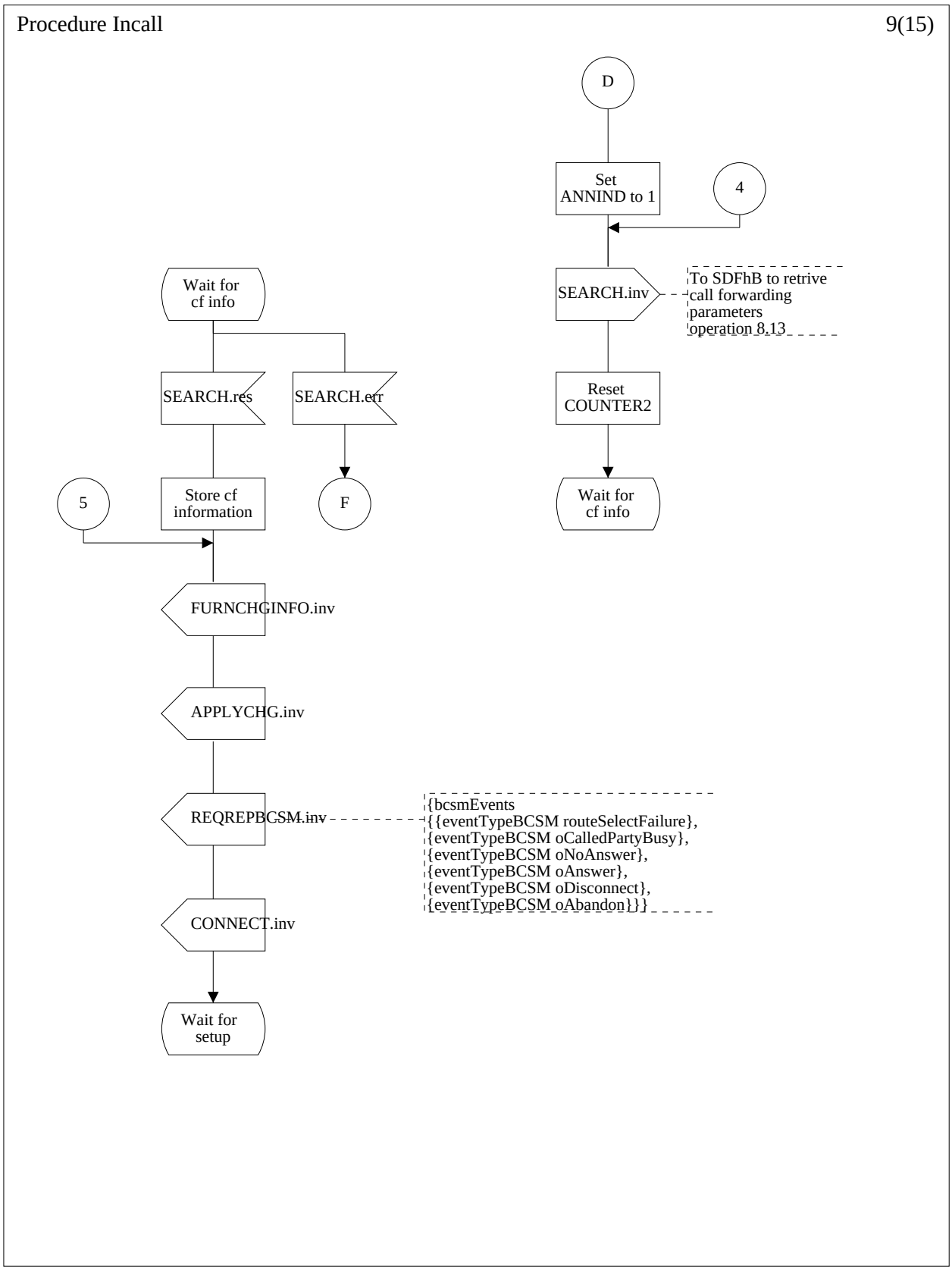


**Figure 16 (sheet 9 of 15): Incoming UPT Call procedures**

Procedure Incall 10(15)

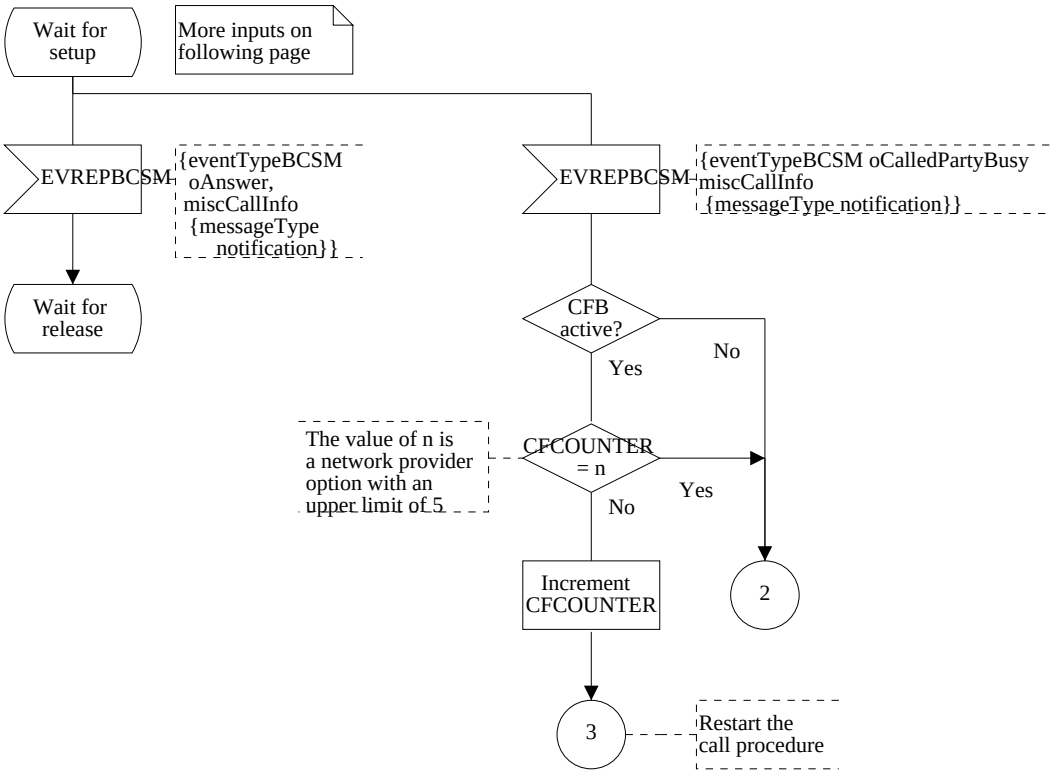


**Figure 16 (sheet 10 of 15): Incoming UPT Call procedures**

Procedure Incall 11(15)

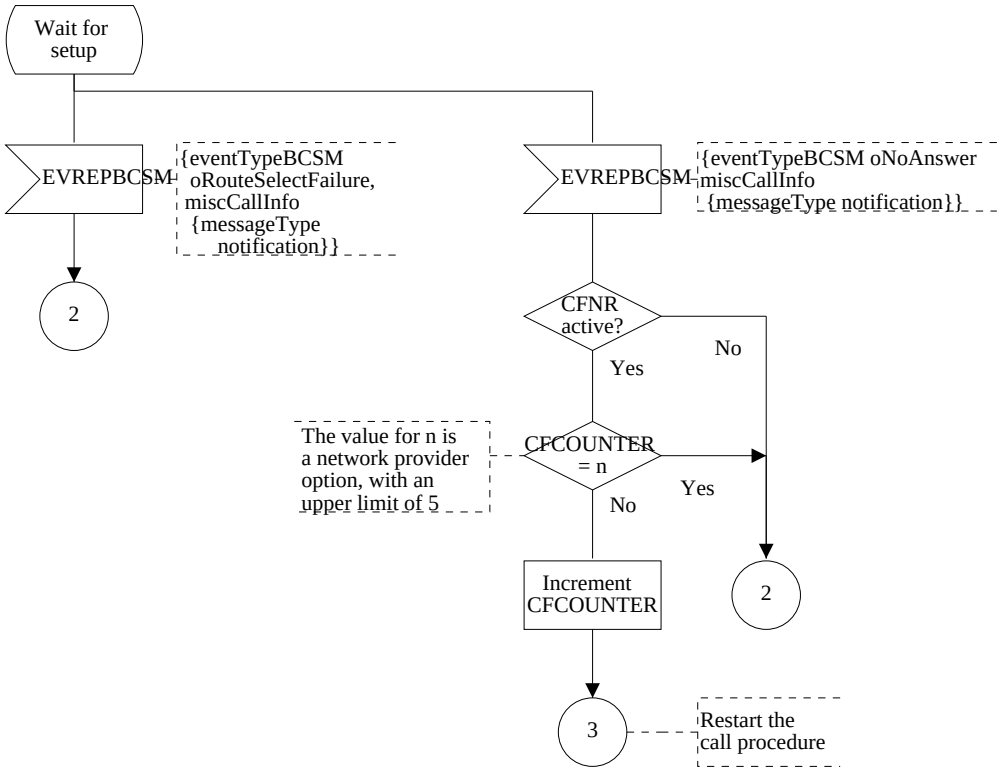


**Figure 16 (sheet 11 of 15): Incoming UPT Call procedures**

Procedure Incall 12(15)

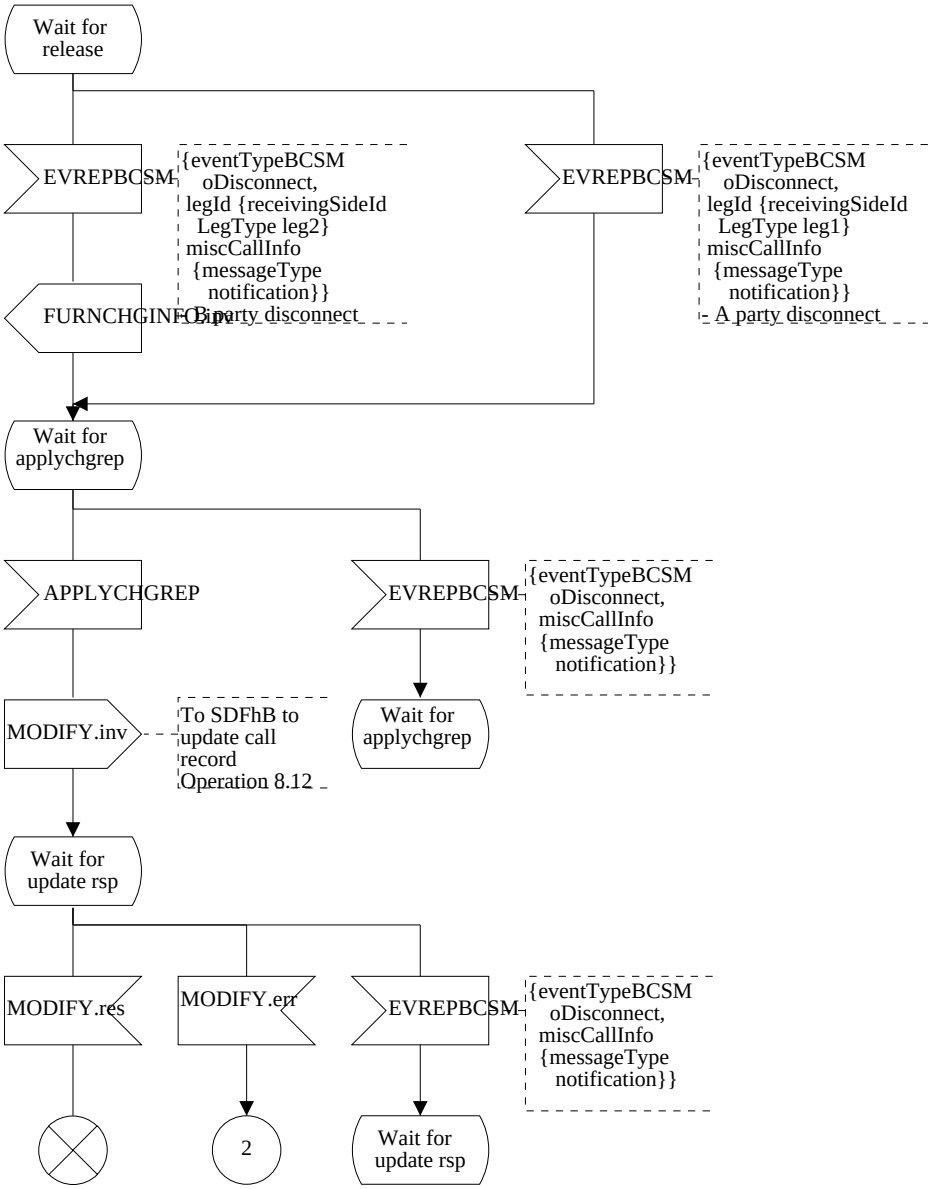


**Figure 16 (sheet 12 of 15): Incoming UPT Call procedures**

Procedure Incall 13(15)

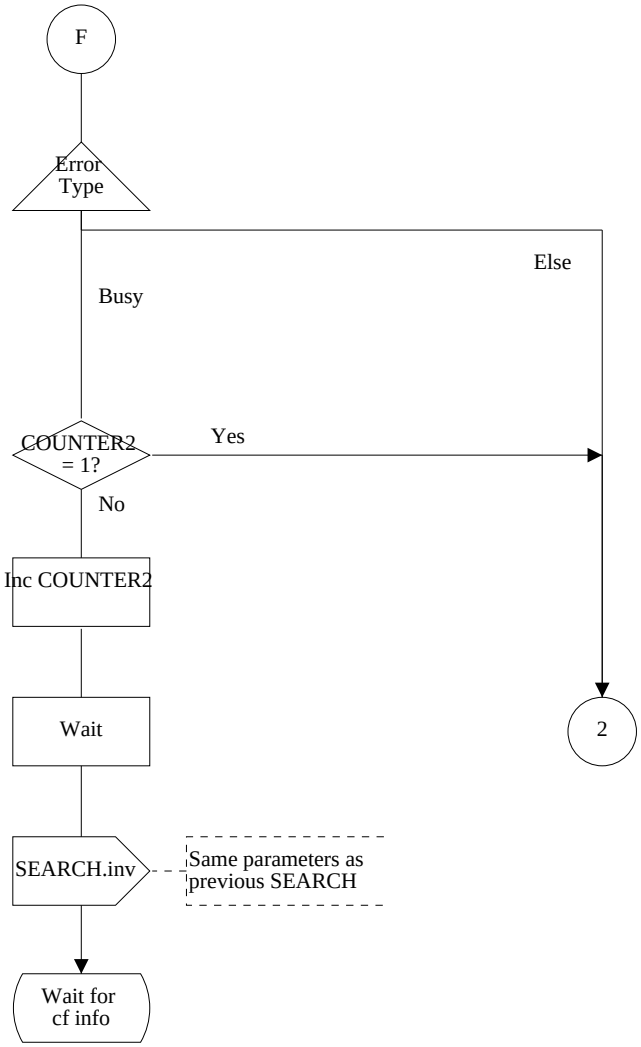


**Figure 16 (sheet 13 of 15): Incoming UPT Call procedures**

Procedure Incall 14(15)

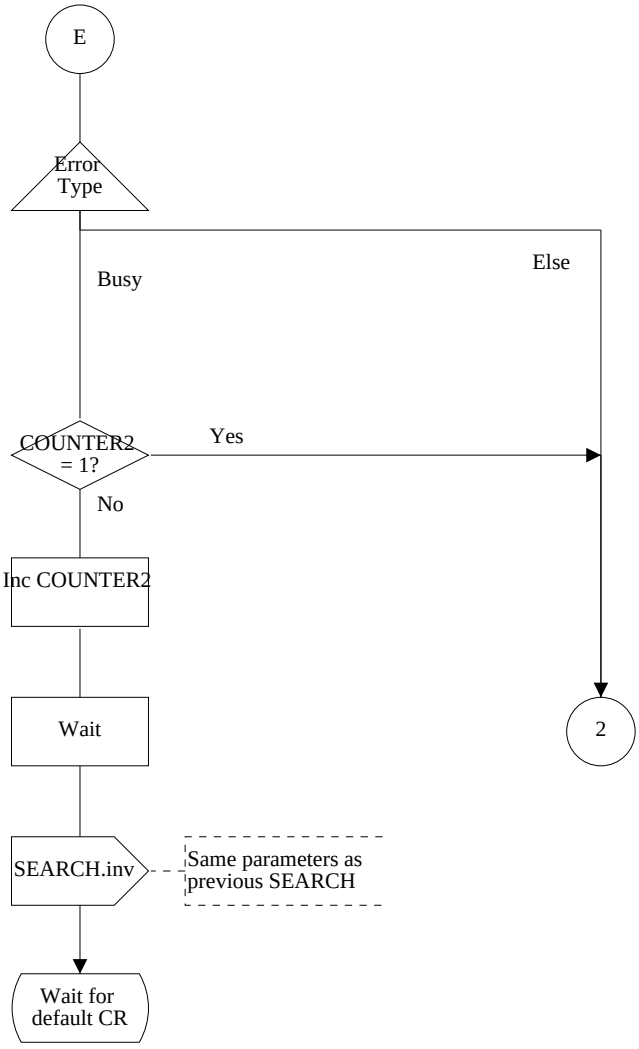


**Figure 16 (sheet 14 of 15): Incoming UPT Call procedures**

Procedure Incall 15(15)

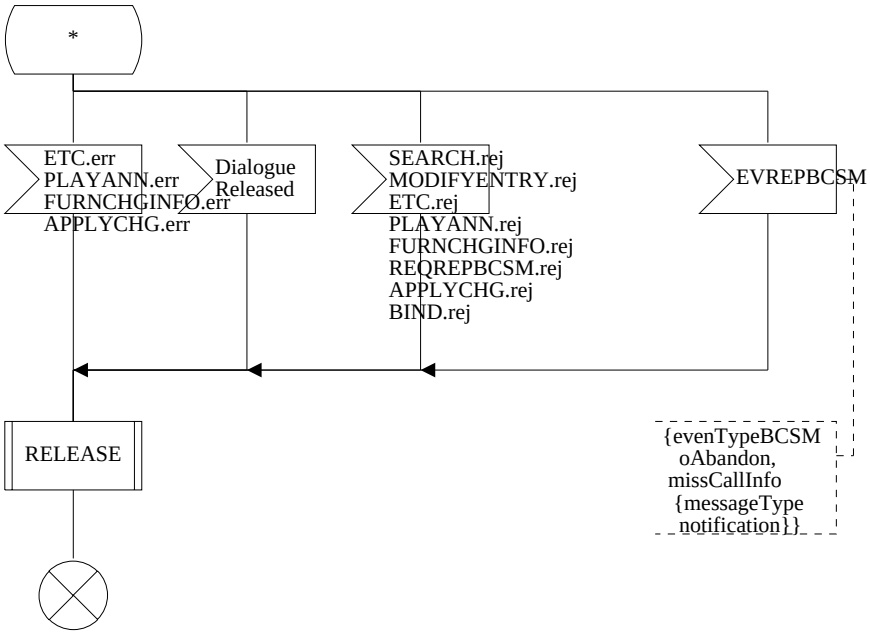


**Figure 16 (sheet 15 of 15): Incoming UPT Call procedures**

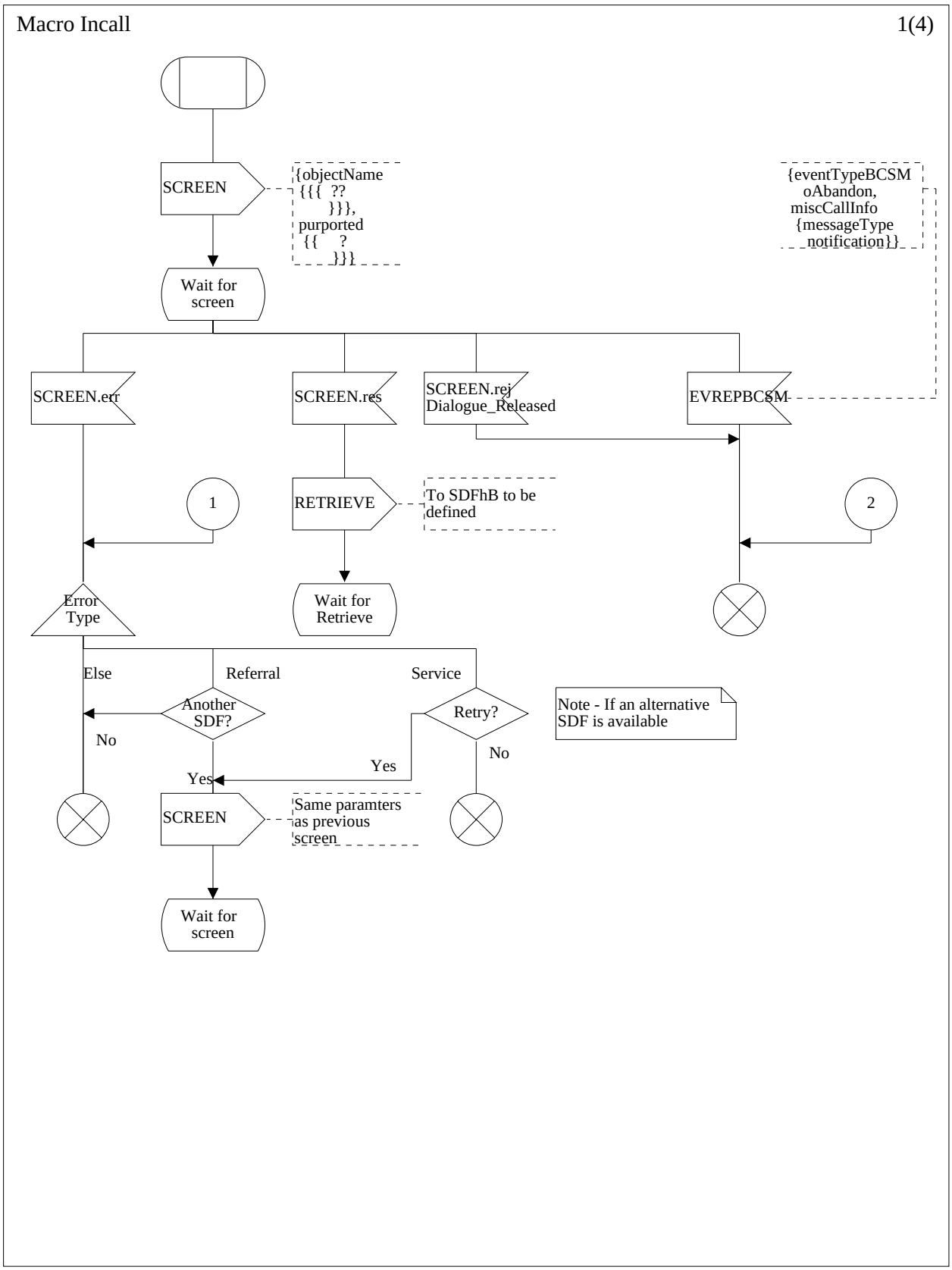


**Figure 17 (sheet 1 of 4): Incoming UPT Call macro**

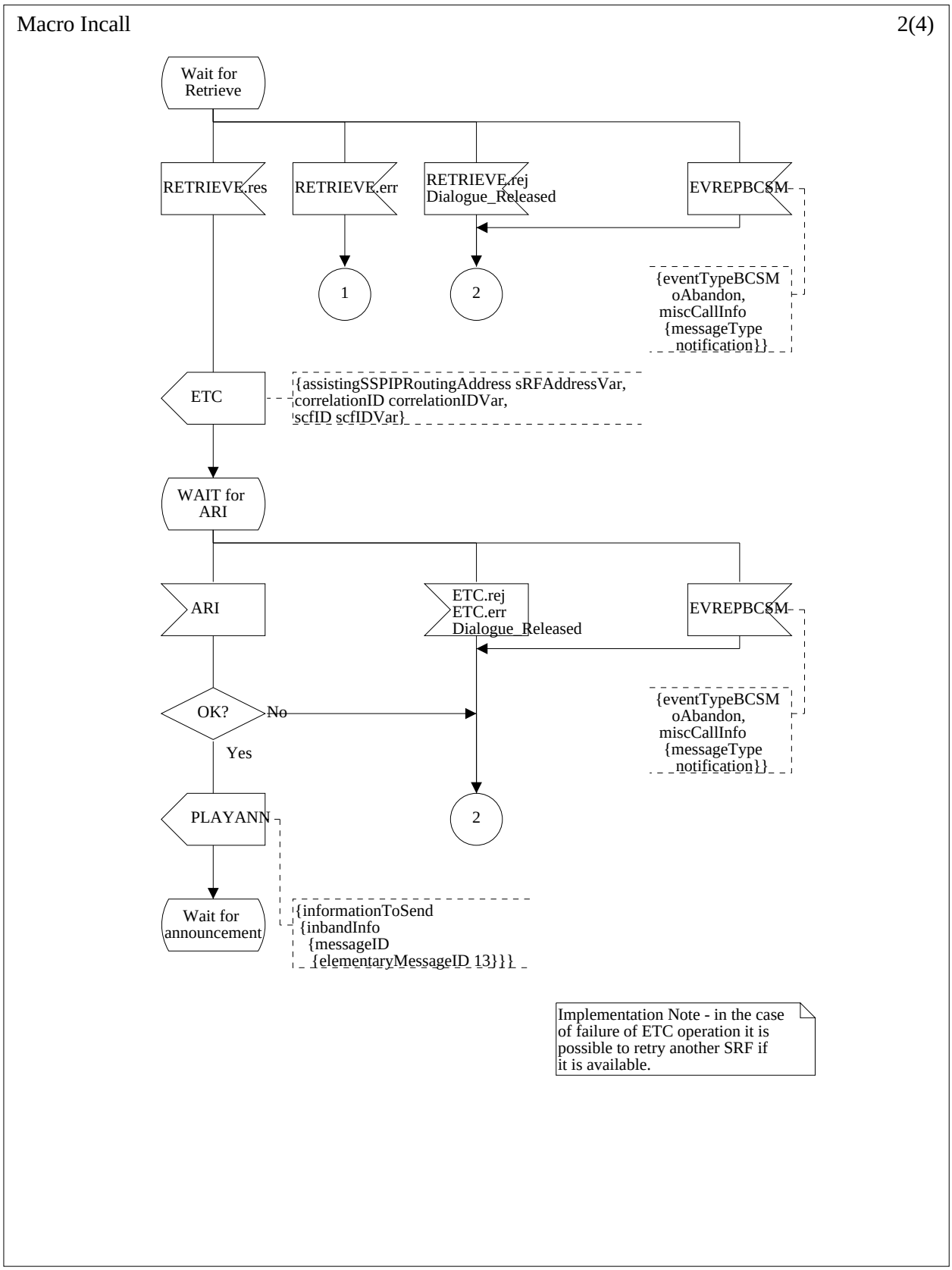


**Figure 17 (sheet 2 of 4): Incoming UPT Call macro**

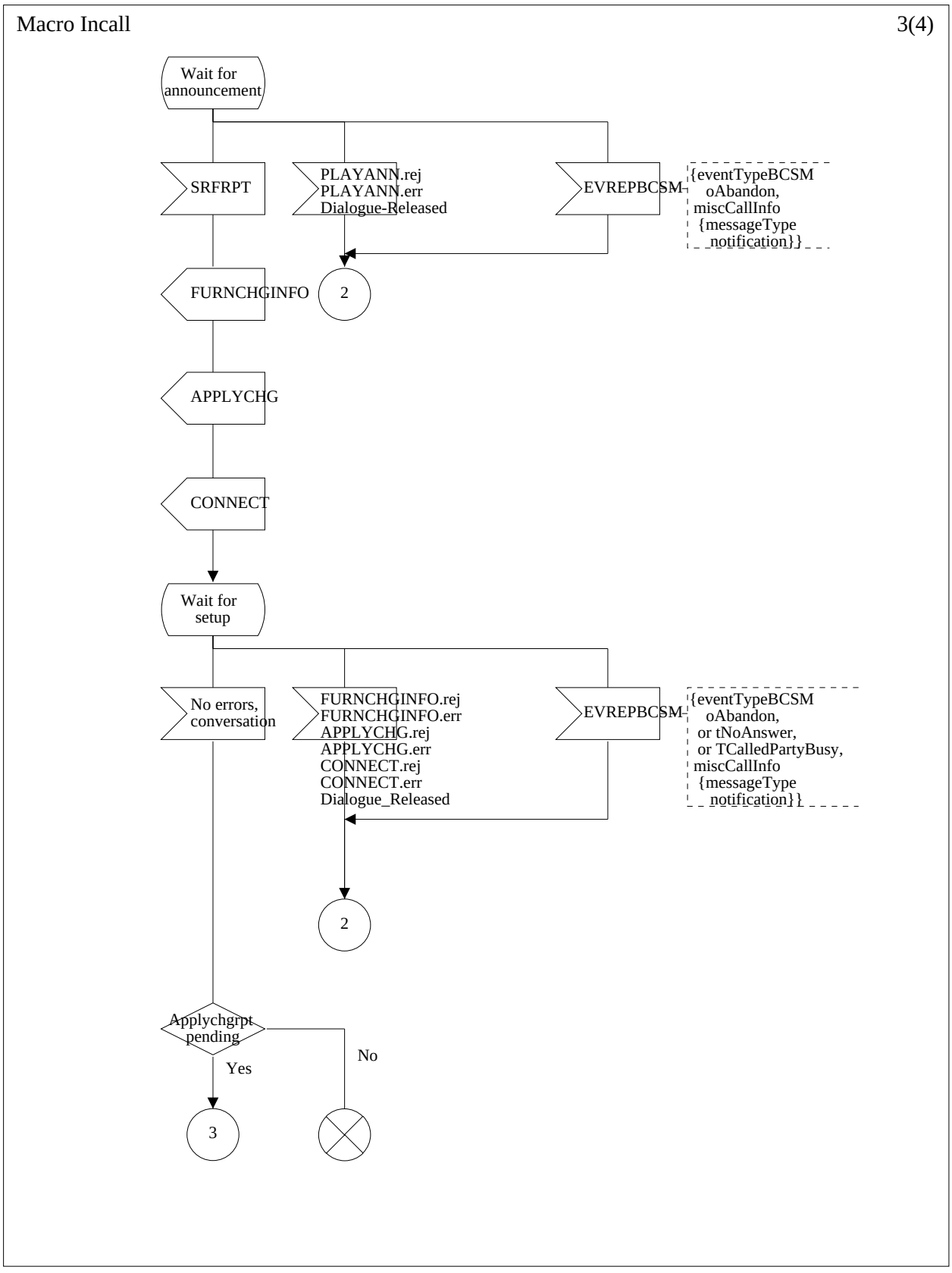


**Figure 17 (sheet 3 of 4): Incoming UPT Call macro**

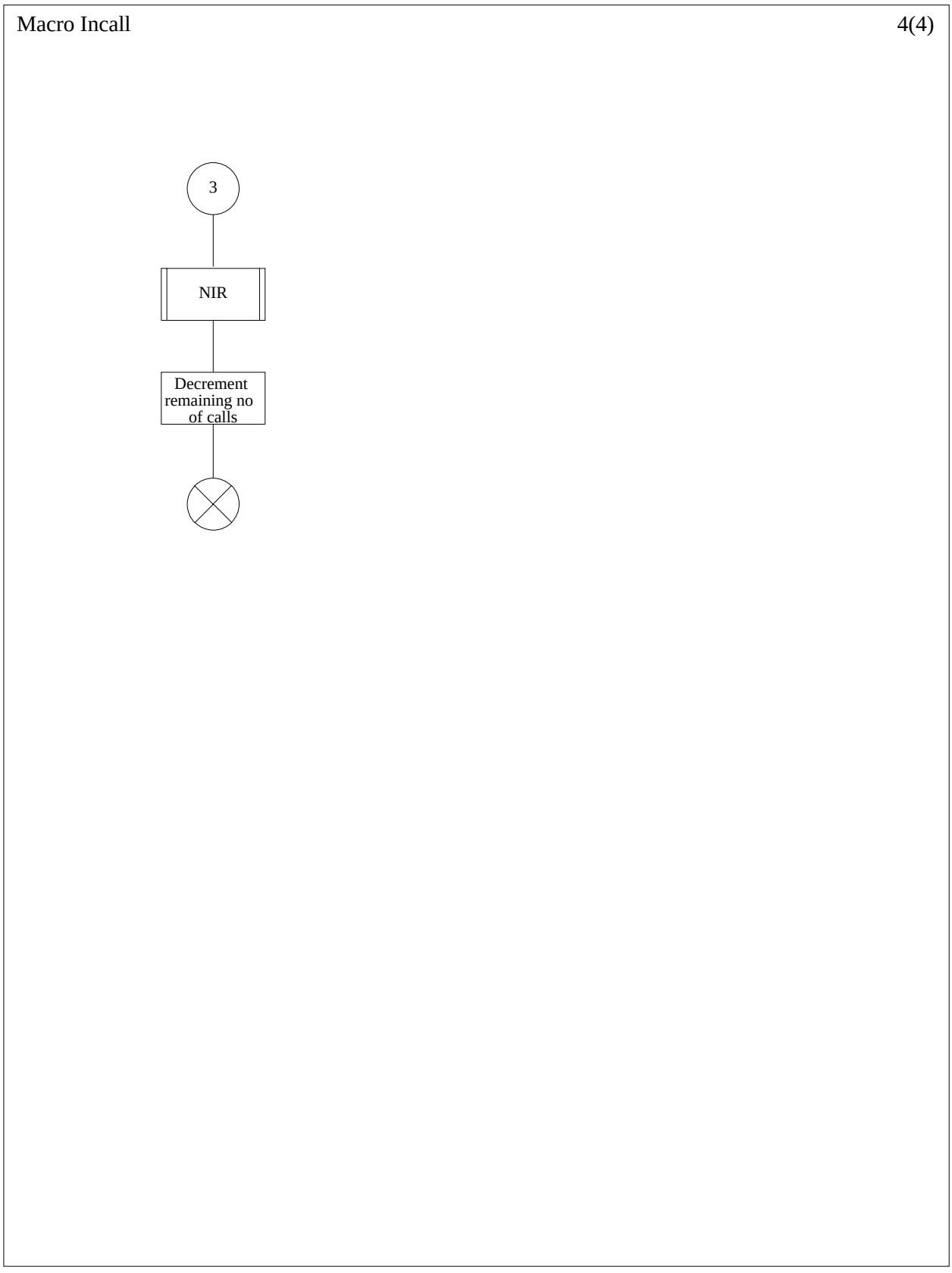


**Figure 17 (sheet 4 of 4): Incoming UPT Call macro**

## 7.5 Service Profile Modification

### 7.5.1 General

The Service Profile Modification takes place when a user requests to do so in the FRI procedure. It is used in order for the user to make changes to his service profile from a network access.

The Service Profile Modification Procedure (SPM) has two logical outputs. They are identical to the ones for the FRI procedure. Like in the IA and the FRI procedures the user is able to abandon at any stage in the procedure.

### 7.5.2 Detailed procedure

The SDL for this procedure can be found in figure 18.

The user is invited to enter the modification request, after which the state "Wait for User Entry" is entered. There are three exits from this state.

a) An error occurs in the P&C operation. If this error is an ImproperCallerResponse, the number of retries counter is incremented. The user is invited to retry the SPM procedure from the beginning unless the maximum number of retries is exceeded, in which case the user is released. Other errors lead to the Exit=NOK, and the user is released.

b) The user abandon leads to Exit=NOK and the user is released.

c) The user enters valid data, the procedure can continue.

The user is requested to enter confirm, cancel or escape. The state "Wait for User Confirmation" is entered. There are four exits from this state.

a) An error occurs in the P&C operation. If this error is an ImproperCallerResponse, the number of retries counter is incremented. The user is invited to retry the SPM procedure from the beginning unless the maximum number of retries is exceeded, in which case the user is released. Other errors lead to the Exit=NOK, and the user is released.

b) The user refuses to perform the SPM by means of the ESC. In this case the counter of retries is incremented. If it does not reach its maximum, the Exit=OK is used, otherwise the user is released.

c) The user abandon leads to Exit=NOK and release of the user.

d) The user enters the response Y, and the procedure can continue.

The user is then informed by means of a PLAYANN operation that he is waiting for an acknowledgement from the Home Network to his request. At the same time the MODIFY operation is performed against the SDFh. This operation contains parameters described in the table 5 in subclause 8.14. This table describes how the string to be entered by the user conditions the contents of the operation. The sending of the MODIFY operation leads to the state "Wait For Confirmation". This takes place for all the cases described by the table. There are three exits from this state.

a) An error occurs in the MODIFY operation. If this error is a busy indication, one retry is attempted. Otherwise and other errors lead to a PLAYANN to the user and the Exit=NOK, and the user is released.

b) The user abandon leads to Exit=NOK and the user is released.

c) The ModifyEntry is successfully completed, and the procedure can continue.

The continuation of the procedure involves a P&C operation to the user. This indicates that the modification was either performed or rejected, and invites the user to make another request or to terminate. A FURNCHGINFO is then sent to the SSF, Exit=OK. The follow-on procedure then continues with the FRI procedure as described in subclause 7.2.2.

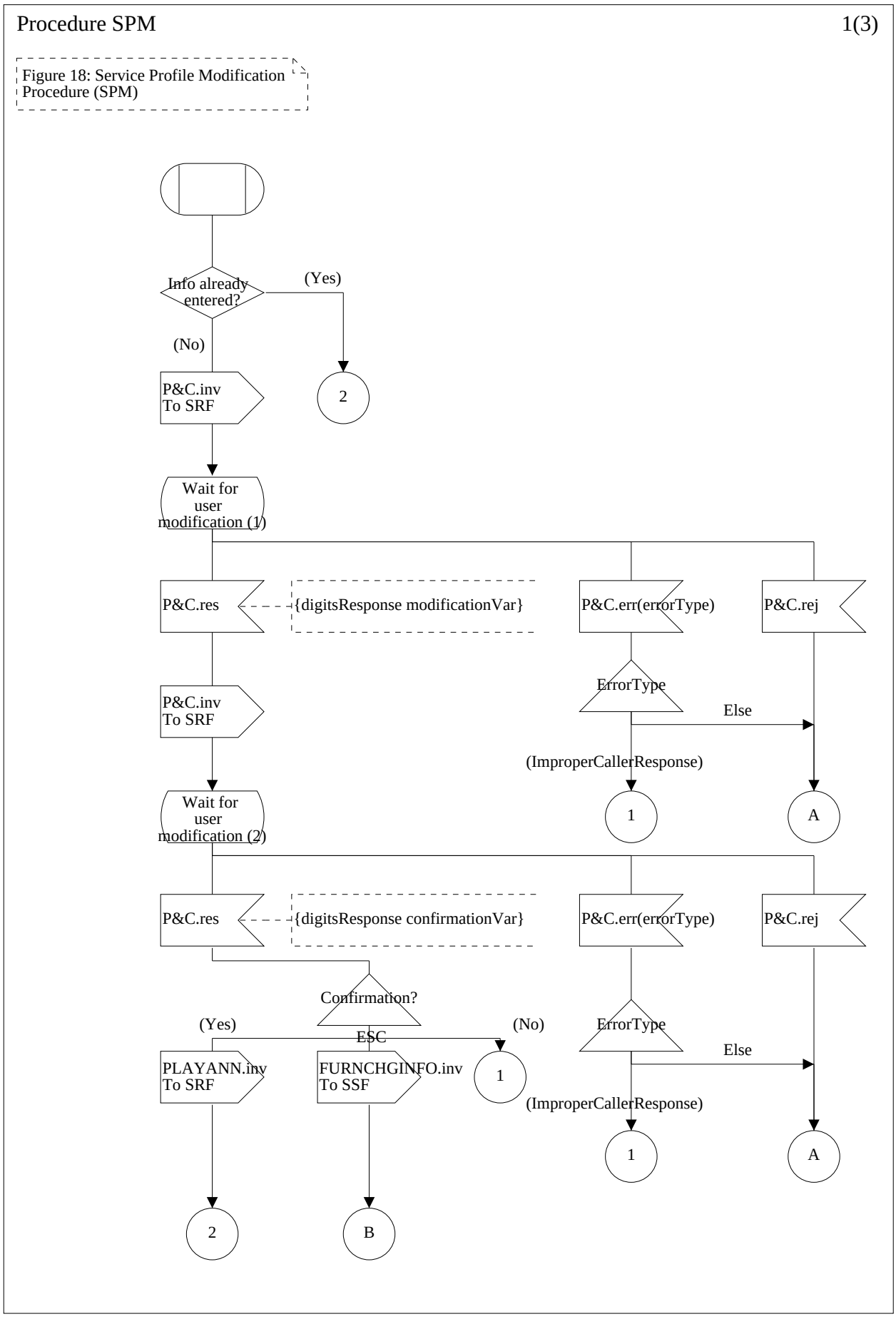


**Figure 18 (sheet 1 of 3): Service Profile Modification procedure**

Procedure SPM 2(3)

Figure 18: Service Profile Modification Procedure (SPM)

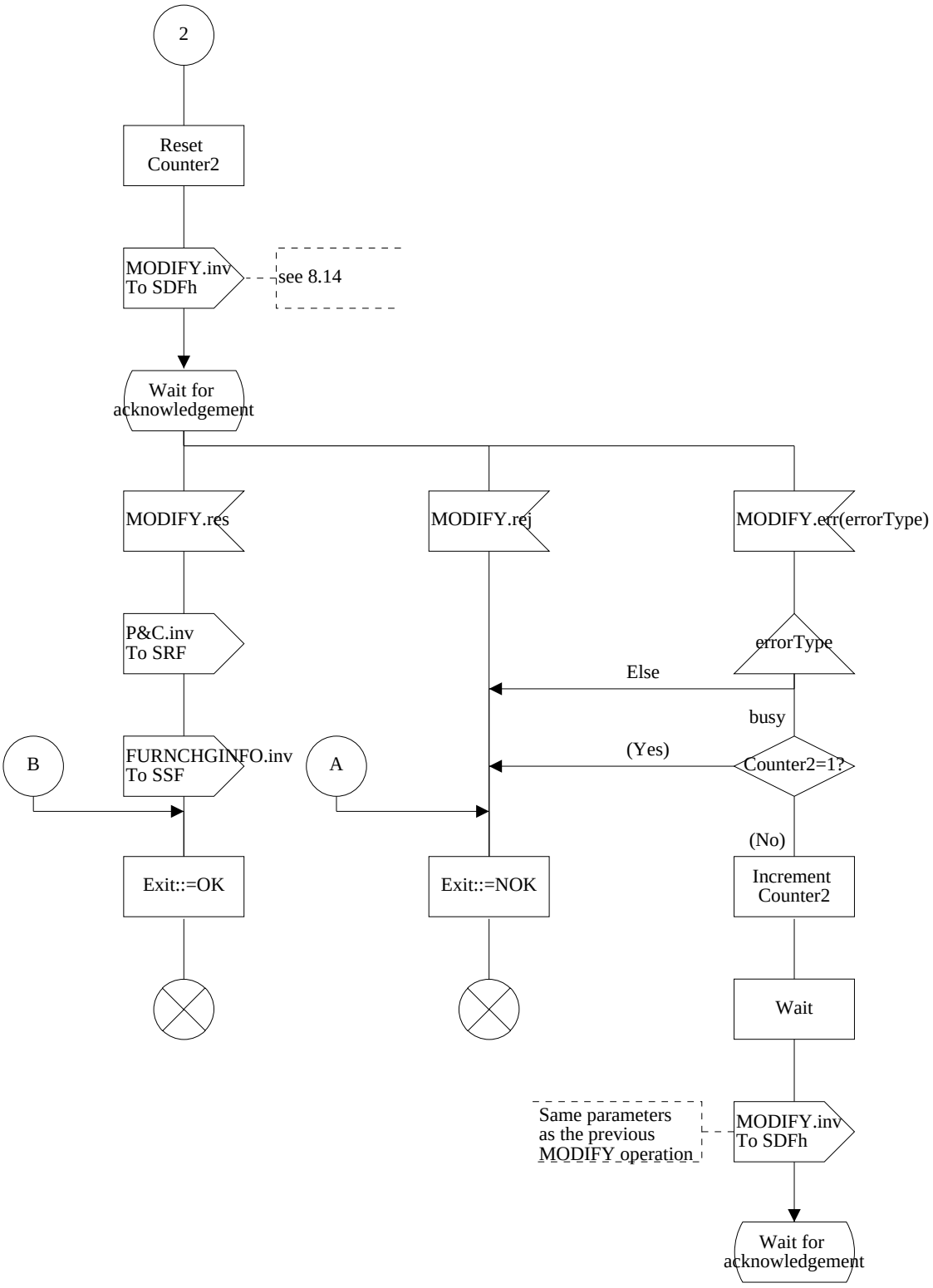


**Figure 18 (sheet 2 of 3): Service Profile Modification procedure**

Procedure SPM 3(3)

Figure 18: Service Profile Modification Procedure (SPM)

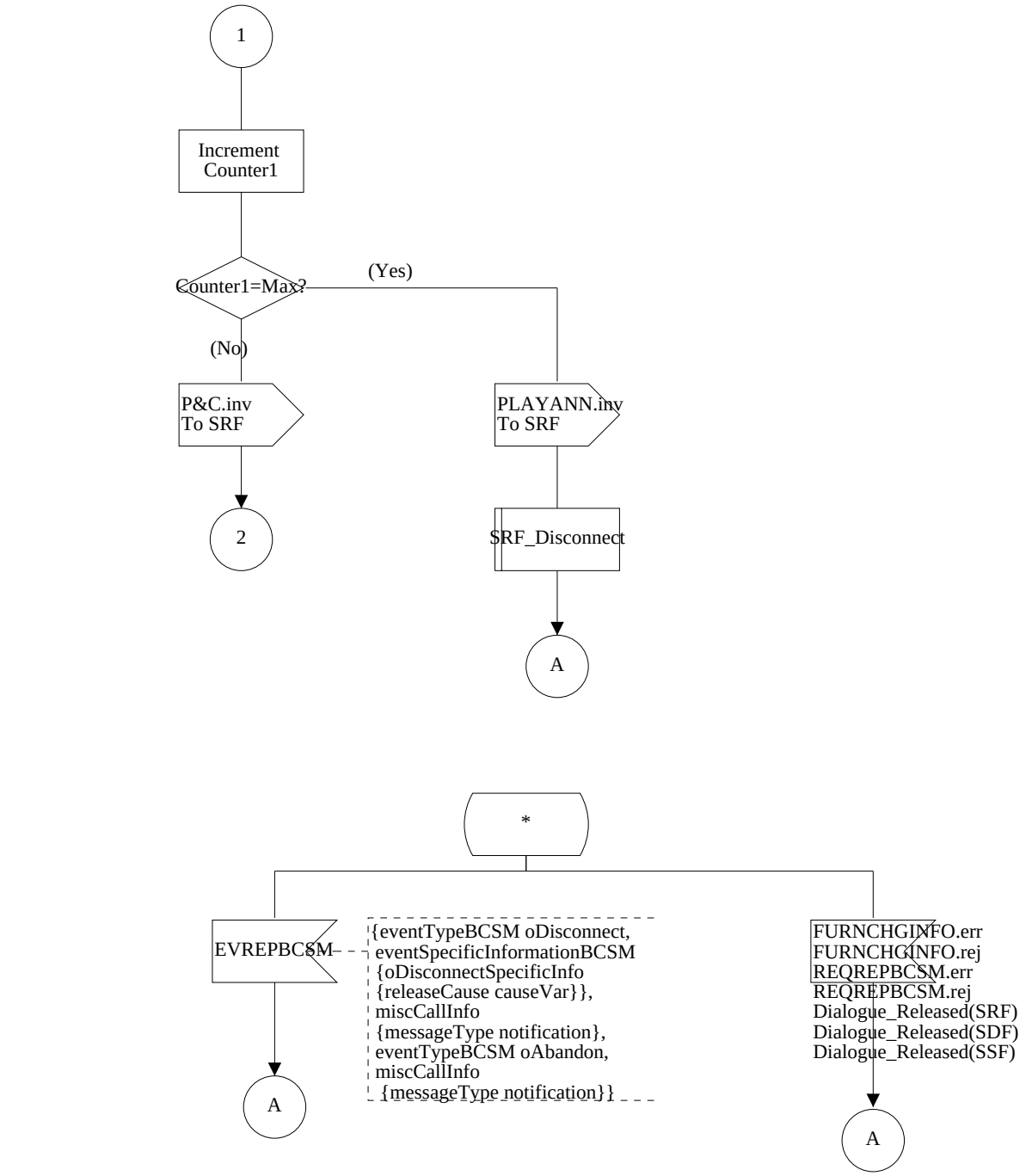


**Figure 18 (sheet 3 of 3): Service Profile Modification procedure**

## 7.6 Change of PIN Code

### 7.6.1 General

The change of PIN Code takes place either when a user has requested this procedure in the FIR procedure or when he has successfully identified himself with the SPIN. It is used to change the PIN code attached to the user's PUI. This can be done to personalize the PIN code or to prevent a third party from guessing it.

The associated procedure is named PIN Change (PIN_CHANGE). It has two logical outputs. They are identical to the ones for the FRI procedure. Like in the IA and in the FRI procedures, at any stage of the procedure, the user can abandon.

### 7.6.2 Detailed procedure

The SDL diagram for this procedure is figure 19.

If the PIN_CHANGE procedure follows a SPIN authentication, the user is asked through a P&C operation whether he wants to modify his PIN code or not. Then the SCF moves to the state "Wait for user confirmation". Three solutions are possible to exit this state (besides the user abandon included in the state *):

a) An error has occurred for the P&C operation. If the error is an ImproperCallerResponse, the user is allowed to retry the PIN_CHANGE procedure from the beginning provided that the maximum number of retries is not exceeded. When that number is reached, the user is automatically released. For the other errors, the PIN_CHANGE procedure is terminated with Exit = NOK, the user is released.

b) The user refuses to perform the PIN_CHANGE procedure. The procedure is normally stopped (Exit = OK) and the user is proposed a new feature identification request (see subclause 7.2.2).

c) The user accepts the PIN_CHANGE procedure. The procedure can continue as described below.

The user is requested to enter his new PIN code sequence (twice the new PIN code separated by the * digit) with a P&C operation. It is the first announcement made to a user when he has initiated this procedure though the FRI procedure (i.e. the user has not used the SPIN procedure that automatically starts the PIN_CHANGE procedure). The SCF moves to the state "Wait for user info" while waiting for the user to input his new PIN. The four possible outcomes are:

a) The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user's release. The PIN_CHANGE procedure is terminated and followed by the RELEASE procedure.

b) The user has mistyped his input. This can be recognized either through an error of type ImproperCallerResponse in response to the P&C operation or through an analysis of the user's input (e.g. the two new PIN codes dialled by the user are different). In that case, the user is allowed to retry the PIN_CHANGE procedure from the beginning provided that the maximum number of retries is not exceeded. When that number is reached, the user is automatically released.

c) An error (other than ImproperCallerResponse) for the P&C operation has occurred. The PIN_CHANGE procedure is terminated with Exit = NOK and the user is released.

d) The user has correctly provided the new PIN code sequence. The procedure can continue as described below.

The user is informed with a PLAYANN operation that he has to wait for the acknowledgement of his request and that his request is being processed. At the same time the ModifyEntry operation to change the value of the PIN code is sent to the SDF. The three outcomes to this query are:

a) The user has abandoned the procedure: The SCF receives an EVREPBCSM indicating the user's release. The PIN_CHANGE procedure is terminated and followed by the RELEASE procedure.

b) An error has occurred and the SCF receives an error indication. The only error to be treated is the Busy error. In that case the ModifyEntry operation is sent back to the SDF after a small delay set up by a timer (network operator dependent). Another Busy error leads to the rejection of the operation like in the other error cases and the termination of the PIN_CHANGE procedure. The user is then released.

c) The ModifyEntry operation has succeeded: The user is informed with a P&C operation of that success and can move on to the next procedure (feature request identification, see subclause 7.2.2).

Procedure Pin_Change 1(3)

Figure 19: Pin change procedure (PIN_CHANGE)

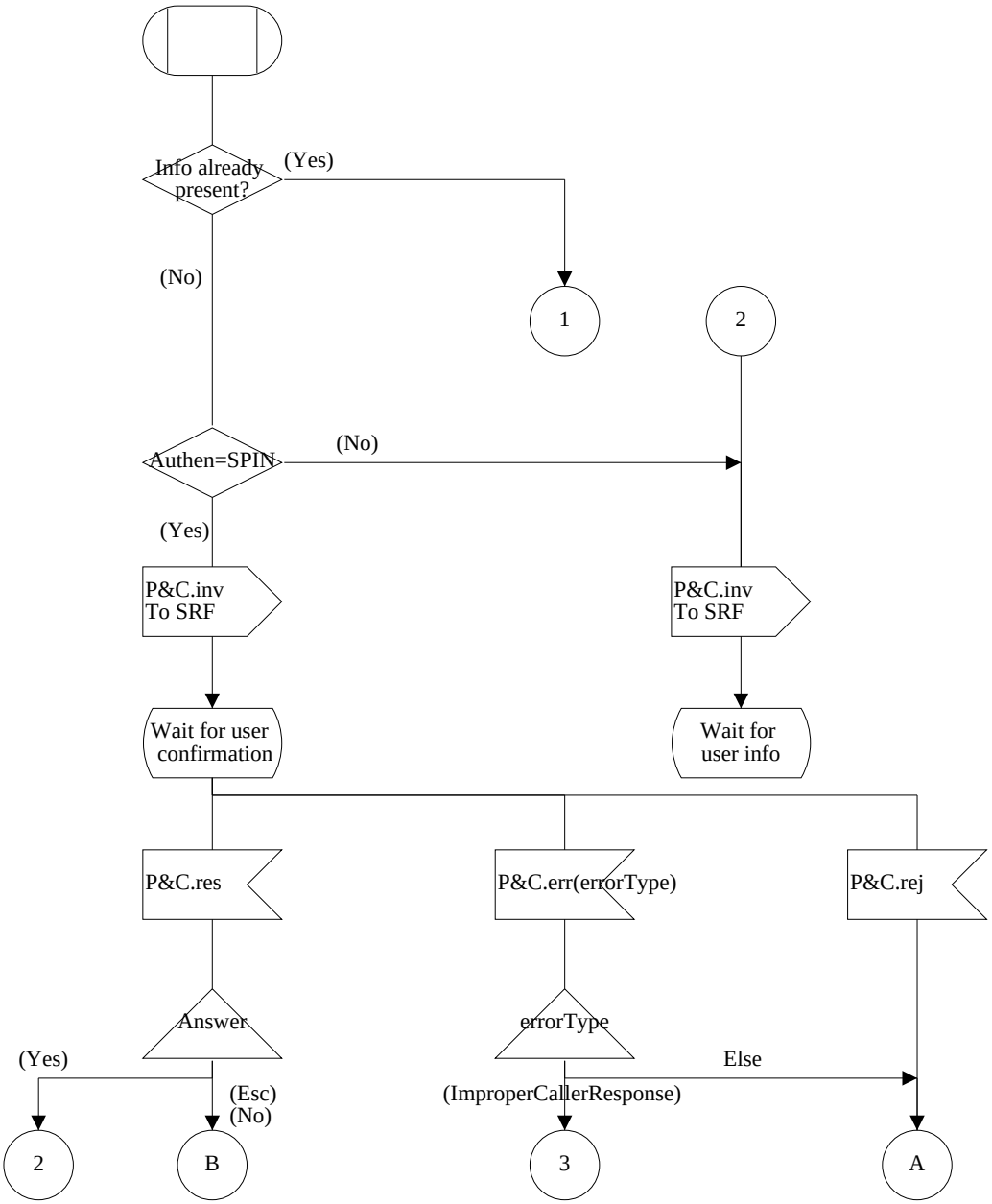


**Figure 19 (sheet 1 of 3): PIN Change procedure**

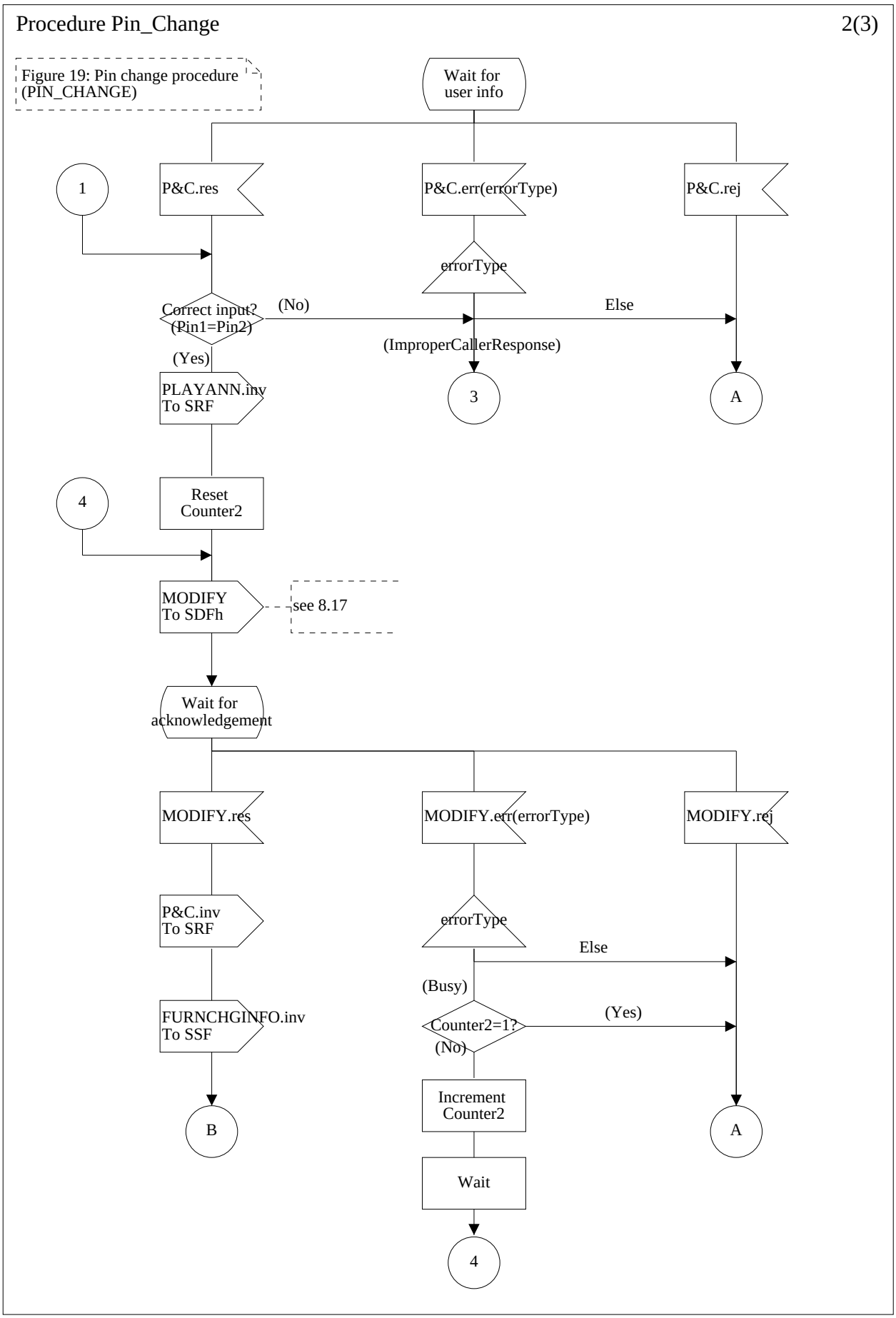


**Figure 19 (sheet 2 of 3): PIN Change procedure**

Procedure Pin_Change 3(3)

Figure 19: Pin change procedure (PIN_CHANGE)

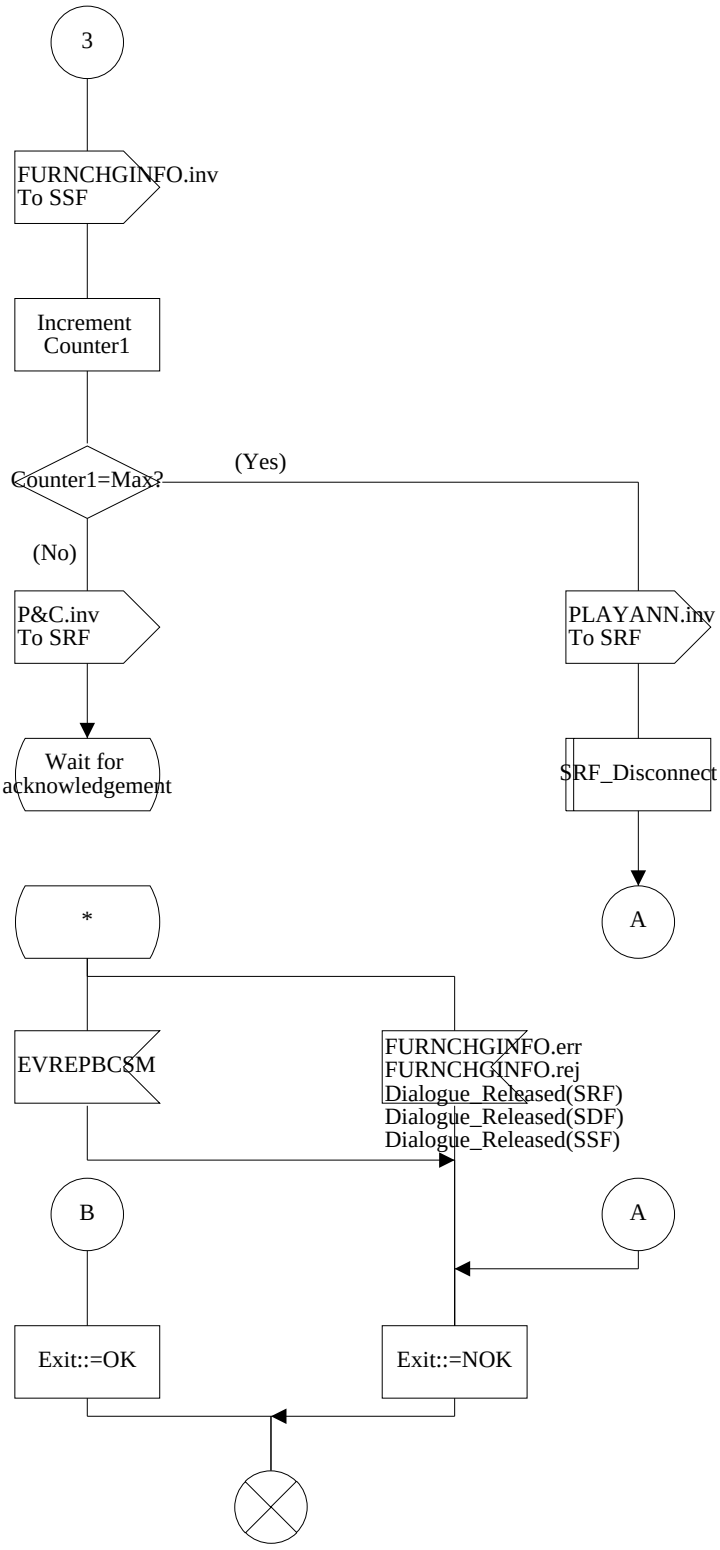


**Figure 19 (sheet 3 of 3): PIN Change procedure**

# 8 SDF Procedures

This clause describes the operations that sent from an SCF to an SDF. Most of these operations are internetwork operations and fully describe the internetwork interface used in UPT. The operations are built upon the data model presented in clause 6. The subclauses below serve as references in clause 7.

## 8.1 Agreement check at the service provider level

This operation taking place within the visited network allows the visited service provider to verify if there exists a general agreement between the visited and the home service providers. It is used before giving to the user access to the service. The operation used should be the SEARCH operation with the following argument:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   visitedProviderId}}}},
    subset      oneLevel,
    extendedFilter{ item{   equality{   type    providerId,
                            assertion   homeProviderId}}}}}
```

The value **homeProviderId** is deduced from the information provided by the user, i.e. his PUI.

## 8.2 User's authentication

This operation is used to identify and authenticate a user. The argument of this operation depends on the type of authentication used. It initiates the dialogue between the SDFh and the visited SCF. The operation used should be the BIND operation with the following argument:

```
{   credentials pinCredentials} -- for the PIN authentication

{   credentials spinCredentials} -- for the SPIN authentication

{   credentials strongCredentials} -- for the strong authentication
```

The **pinCredentials, spinCredentials** and **strongCredentials** values are defined in subclause 6.3.2.

## 8.3 Provider agreement at the service feature level

This operation allows the visited service provider to locally check if there exists an agreement between the visited and the home service providers for a specific service feature, i.e. if any visiting user have normally (if his subscription allows it) access to that specific service feature. At the same time it verifies if the current location of the calling UPT user can be used given the service agreements. The operation used should be the SEARCH operation with the following arguments:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   visitedProviderId}},
                    {{  type    providerId,
                        value   homeProviderId}}}},
    subset      oneLevel,
    filter{ and{    item{   equality{   type        serviceId,
                        assertion   requestedServiceId}},
            item{   extensibleMatch{    matchingRule
                                {numericStringSubstringsMatch},
                            type            providedLocations,
                            matchValue
                                {initial        currentLocation},
                            reverseMatch    TRUE}}}}}
```

The same operation is used to verify if a number provided by the user as a registration address is valid as a registration address. In that case the value **currentLocation** is to be replaced by the registration address provided by the user.

## 8.4 Check on the subscription to the service

This operation is used to verify in the home database that the user has subscribed to the service feature he is requesting and if any particular restrictions apply. The type of restriction that is used in this example is the check on the permission to perform the service feature at the current location of the user. The operation used should be the SEARCH operation with the following argument:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   homeProvider}},
                    {{  type    pui,
                        value   userPUI}}}},
    subset      baseObject,
    selection{  attributes          {select {allowedServices}},
            contextAssertions   {{type      allowedServices,
                            contextAssertion
        {contextType    numericContext,
        contextValues   {currentLocation}}}}}}}
```

If there is no location restrictions due to the agreements between the service providers, the operation should be modified and the contextAssertions component removed.

### 8.5 Check on the registration address

This operation is used for two purposes: Firstly to verify that the terminal number where the user wants to register is not forbidden as a registration address (free phone, emergency service...), secondly to check if any restrictions of the service offered to the visiting users apply to the registration address chosen by the user. It is important to notice that the first case does not apply to the remote registration since the forbidden registration addresses for a network are not necessarily known by another network. Also the forbidden registration addresses of the visited network do not take into account the terminals that are blocked for registration, because it is not possible to maintain such a list, however such a check is desirable in the future. Concerning the possible restrictions of the service for the visiting users, it is a service provider matter and might not exist. This type of check should be carried out by a SEARCH operation with the following argument:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   visitedProviderId}},
                    {{  type    pui,
                        value   userPUI}},
                    {{  type    basicServiceId,
                        value   isdnTelephony}}}},
    subset      baseObject,
    filter{ item{   extensibleMatch{    matchingRule    {numericStringSubstringsMatch},
                        type        allowedRegistrationAddress,
                        matchValue
                            {initial    givenRegistrationAddress},
                        reverseMatch    TRUE}}}}}
```

The **givenRegistrationAddress** attribute contains the address where the user wants to register and provided before this operation being sent. If the user has entered the address, the address is translated into an international address. Dialled local numbers are considered as local to the visited network. This rule stands for all the numbers entered by the user.

### 8.6 Update of the registration address

This operation is used at the end of the registration procedure to change the registration of the address of the user in the home database. The operation should not modify the default registration address. The operation used should be the MODIFY operation with the following argument:

```
{{object{   rdnSequence{{{  type    providerId,
                    value   homeProviderId}},
                {{  type    pui,
                    value   userPUI}},
                {{  type    basicServiceId,
                    value   isdnTelephony}}}},
    changes{{   resetValue  icRegistrationAddress,
            addValues   {type   icRegistrationAddress,
    --see Note 1        valuesWithContext{
                        value       newRegistrationAddress,
                        contextList {{contextType   temporal,
                                contextValues
                            {{  startTime   currentTime,
                                endTime
                                    endOfRegistration}}}}}}}}}
```

The time entered by the user is a local time (local to the visited network). This time is translated into UTC time. This is the case for the parameters **currentTime** and **endOfRegistration**.

NOTE: Depending whether the user has specified a limitation to his registration or not, a context is used or not (i.e. the **valuesWithContext** component is replaced by the **values** component). That context will contain the specification of the time validity of the registration i.e when it starts and when it ends.

### 8.7 Reading the registration address

This operation is used to verify that a user is actually registered when he requests a deregistration. The operation used should be the SEARCH operation with the following arguments:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   homeProviderId}},
                    {{  type    pui,
                        value   userPUI}},
                    {{  type    basicServiceId,
                        value   isdnTelephony}}}},
    subset  baseObject,
    selection{  attributes{ select{ icRegistrationAddress}},
            contextAssertions{{
                type    icRegistrationAddress,
                contextAssertion{
                    contextType temporalContext,
                    contextValues   {currentTime}}}}}}
```

The registration addresses whose context assertion is true (i.e. applicable at the time of the request) are retrieved. If no registration address is applicable, the default registration address is retrieved

### 8.8 Deregistration

This operation is used to remove the current registration address from the registrationAddress attribute in the home database. The operation used should be the MODIFY operation with the following parameters:

```
{{object{   rdnSequence{{{  type    providerId,
                    value   homeProviderId}},
                {{  type    pui,
                    value   userPUI}},
                {{  type    basicServiceId,
                    value   isdnTelephony}}}},
    changes{{   resetValues{    type        icRegistrationAddress}}}}
```

### 8.9 Check on the user credit

This operation verifies if the user's account has some available credit. It is important to notice that it is not possible to do on-line charging since the home SDF is not able to calculate the charge. This credit check is therefore not a real-time check. The operation used should be the SEARCH operation with the following arguments:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   homeProviderId}},
                    {{  type    pui,
                        value   userPUI}}}},
    subset  baseObject,
    filter{ item{   greaterOrEqual{ type        userCredit,
                        assertion   NULL}}}}}
```

When this operation is used to check the credit of the called used, the **rdnSequence** should be changed. The **pui** and **userPUI** parameter should be respectively replaced by the **uptNumber** and **calledUserUPTN**.

### 8.10 Check on the destination address

The operation checks if the called address is an address that can be called according to the user's rights. A similar check could take place in the visited network with the SDFo. The operation used should be the SEARCH operation with the following argument:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   homeProviderId}},
                    {{  type    pui,
                        value   userPUI}},
                    {{  type    basicServiceId,
                        value   isdnTelephony}}}},
    subset  baseObject,
    filter{item{    extensibleMatch{    matchingRule    {numericStringSubstringsMatch},
                        type                allowedDestinations,
                        matchValue  {initial    dialledNumber},
                        reverseMatch    TRUE}}}}}
```

### 8.11 Reading of the routeing address

This operation is used to retrieve the routeing address corresponding to a UPT user. The address returned takes into account the call forwarding unconditional service, the variable routeing service and the registration service. If several values are returned in the response to the request, it is the service logic that decides which one is valid at the time of the request. The operation used should be the SEARCH operation with the following argument:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   calledHomeProviderId}},
                    {{  type    uptNumber,
                        value   userUPTN}},
                    {{  type    basicServiceId,
                        value   isdnTelephony}}}},
    subset  wholeSubtree,
    selection{  attributes{ select{ routingAddress,
                            icRegistrationAddress}},
            contextAssertions{
                type    icRegistrationAddress,
                contextAssertion{
                    contextType temporalContext,
                    contextValues   {currentTime}}},
                type    routingAddress,
                    contextType temporalContext,
                    contextValues   {currentTime}}},
                type    routingAddress,
                    contextType numericContext,
                    contextValues   {calledPartyAddress}}},
            returnContexts  TRUE},
    filter{ or{{    item{   present icRegistration}},
        {   and{{   item{   equality{   type    cfuServiceId,
                            assertion
                                callForwardingUnconditional}}},
            {   item{   equality{   type        activationStatus,
                            assertion   activated}}}}},
        {   and{{   item{   equality{   type    vrclServiceId,
                            assertion   variableRoutingOnTime}}},
            {   item{   equality{   type        activationStatus,
                            assertion   activated}}}}},
        {   and{{   item{   equality{   type        vrtServiceId,
                            assertion
                                variableRoutingOnCallingLine}}},
            {   item{   equality{   type        activationStatus,
                            assertion   activated}}}}}}}}}
```

The **calledHomeProviderId** value is deduced from the UPT number provided by the user.

NOTE: The contexts attached to the **routingAddress** attribute define for the variable routeing service which attribute values are valid at the time of the request. When the user has activated the variable routeing on calling party, the calling party address need to be provided and when the time-dependent variable routeing is activated, it is the data and time that need to be provided. The contexts attached to the **registrationAddress** attribute define for the registration service which attribute values are applicable at the time of the request. When the registration address has a limited lifetime, the time of the request should be provided as a context value. When the registration address is not limited in time, the value of the **registrationAddress** attribute is given without context.

### 8.12 Transfer of call records

This operation allows the storage of call records in the database. That database could be the one of the calling or called party. However, it is not possible to directly use the call records in order to modify the user's credit since the SDF is not able to calculate the charge that corresponds to a call record. This operation is used to pass the record of the user's calls over to the SDFh. The operation used should be the MODIFY operation with the following argument:

```
{{object{   rdnSequence{{{  type    providerId,
                    value   homeProviderId}},
                {{  type    pui,
                    value   userPUI}}}},
    changes{{   addValues{  type        callInfoRecords,
                    value       newCallInfoRecord}}}}}
```

When this operation is used for the called user, the **rdnSequence** should be changed. The **pui** and **userPUI** parameters should be respectively replaced by the **uptNumber** and **calledUserUPTN**.

### 8.13 Retrieving call forwarding parameters

This operation is used to retrieve the parameters allowed to the user and attached to his supplementary services. This operation retrieves all the parameters attached to one service. It is possible to have operations retrieving only one parameter and operations dealing with several services. The parameters are retrieved only if the service is activated. By that means, this operation is used to check if a supplementary service is activated when a call attempt reports that this supplementary service should be invoked. It is important to notice that this checking is only applicable to a called UPT user i.e. when the supplementary service is not attached to the line but to the user. The information flows related to this handling of supplementary services are not described in ETR 066 [9]. The operation used should be the SEARCH operation with the following argument:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   calledHomeProviderId}},
                    {{  type    uptNumber,
                        value   userUPTN}},
                    {{  type    basicServiceId,
                        value   isdnTelephony}},
                    {{  type    routingServiceId,
                        value   neededRoutingServiceId}}}},
    subset  baseObject,
    filter{ item{   equality{   type        activationStatus,
                    assertion   activated}}},
    selection   attributes  {select {activatedCFparameters,
                            noReplyConditionTimer}}}}
```

### 8.14 Modifying the service profile

This operation is used to modify the user's service profile. It is used to register and activate supplementary services and for several other type of modifications. Hence it is very difficult to specify all the various types of modifications with only one operation argument since the operation depending of the type of modification can address different objects. However the operation should be in all cases the MODIFY operation. Three call forwarding services are considered for UPT: call forwarding unconditional, call forwarding on no reply and call forwarding on busy. The table 5 specifies the possible arguments according to the type of modification requested by the user:

**Table 5: Arguments of the operations performed to change the user's profile**

| Type of modifications | parameters for the MODIFY operation towards the SDFh |
|---|---|
| Activation of a call forwarding service (unconditional, on busy, on no reply) | {{object{rdnSequence{{{type providerId,<br>value homeProviderId}},<br>{{type pui,<br>value userPUI}},<br>{{type basicServiceId,<br>value isdnTelephony}}<br>{{type cfServiceId,<br>value requestedCallForwardingService}}}},<br>changes{{removeAttribute activationStatus,<br>addAttribute {type activationStatus,<br>values activated}}}}} |
| Deactivation of a call forwarding service | {{object{rdnSequence{{{type providerId,<br>value homeProviderId}},<br>{{type pui,<br>value userPUI}},<br>{{type basicServiceId,<br>value isdnTelephony}}<br>{{type cfServiceId,<br>value requestedCallForwardingService}}}},<br>changes{{removeAttribute activationStatus,<br>addAttribute {type activationStatus,<br>values deactivated}}}}} |
| Registration of a call forwarding service | {{object{rdnSequence{{{type providerId,<br>value homeProviderId}},<br>{{type pui,<br>value userPUI}},<br>{{type basicServiceId,<br>value isdnTelephony}}<br>{{type cfServiceId,<br>value requestCallForwardingService}}}},<br>changes{{removeAttribute activationStatus,<br>addAttribute {type activationStatus,<br>values activated},<br>removeAttribute RoutingAddress<br>addAttribute {type routingAddress,<br>value newForwardedToAddress}}}}} |
| Deregistration of a call forwarding service | {{object{rdnSequence{{{type providerId,<br>value homeProviderId}},<br>{{type pui,<br>value userPUI}},<br>{{type basicServiceId ,<br>value isdnTelephony}}<br>{{type cfServiceId,<br>value callForwardingUnconditional}}}},<br>changes{{removeAttribute activationStatus,<br>removeAttribute routingAddress,<br>addAttribute {type routing Address,<br>values NULL},<br>addAttribute {type activationStatus,<br>values deactivated}}}}} |

(continued)

**Table 5 (concluded): Arguments of the operations performed to change the user's profile**

| Type of modifications | parameters for the MODIFY operation towards the SDFh |
|---|---|
| Change of the default registration address | {{object{rdnSequence{{{type providerId,<br>value homeProviderId}},<br>{{type pui,<br>value userPUI}},<br>{{type basicServiceId,<br>value isdnTelephony}}}},<br>changes{{removeValues {type icRegistrationAddress,<br>valueWithContext{<br>value oldRegistrationAddress,<br>contextList{{contextType override}}}}}}<br>addAttribute {type icRegistrationAddress,<br>valueWithContext{<br>value newRegistrationAddress,<br>contextList{{contextType override}}}}}}} |
| Change of the no reply condition timer | {{object{rdnSequence{{{type providerId,<br>value homeProviderId}},<br>{{type pui,<br>value userPUI}},<br>{{type basicServiceId,<br>value isdnTelephony}}<br>{{type cfServiceId,<br>value callForwardingOnNoReply}}}},<br>changes{{removeAttribute noReplyConditionTimer,<br>addAttribute {type noReplyConditionTimer,<br>value newTimer}}}} |

### 8.15 Getting the routeing address for conditional call forwarding services

This operation checks if a specific call forwarding service was activated and at the same time retrieves the routeing address associated with the service. This operation can be used for all the call forwarding services. The operation used should be the SEARCH operation with the following parameter:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   calledHomeProviderId}},
                    {{  type    uptNumber,
                        value   userUPTN}},
                    {{  type    basicServiceId,
                        value   isdnTelephony}},
                    {{  type    routingServiceId,
                        value   neededRoutingServiceId}}}},
    subset  baseObject,
    filter{ item{   equality{   type        activationStatus,
                    assertion   activated}}},
    selection   attributes  {select {routingAddress}}}}
```

### 8.16 Retrieving the default charging reference point

This operation is used to retrieve the default charging reference point. It could be used to check if a split charging occurs and to calculate the cost of a call. The operation used should be the SEARCH operation with the following argument:

```
{{baseObject{   rdnSequence{{{  type    providerId,
                        value   calledHomeProviderId}},
                    {{  type    uptNumber,
                        value   userUPTN}}}},
    subset  baseObject,
    selection   attributes  {select {defaultChargingReference}}}}
```

### 8.17 Changing the PIN code

This operation is used to change the PIN code of a user. The PIN code is the only user password that can be modified. The operation used should be the MODIFY operation with the following argument:

```
{{object{   rdnSequence{{{  type    providerId,
                    value   homeProviderId}},
                {{  type    pui,
                    value   userPUI}}}},
    changes{{   removeAttribute{    type    uptPassword},
            addAttribute{       type    uptPassword,
                        value   newPin}}}}}
```

The **newPin** is the new value of the PIN code provided by the user.

## Annex A (Normative): ASN.1 Information Object Notation

This modules contains the ASN.1 Information Object Notation for defining the contents of an SDF to fulfil the UPT service.

```
UPT-DataModel
--
DEFINITIONS::=

BEGIN


IMPORTS
    OBJECT-CLASS,
    ATTRIBUTE,
    MATCHING-RULE,
    STRUCTURE-RULE,
    NAME-FORM,
    top,
    alias
FROM InformationFramework
            {joint-iso-ccitt ds(5) module(1) informationFramework(1) 2}

FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 2};

    userPassword,
    objectIdentifierMatch,
    distinguishedNameMatch,
    numericStringMatch,
    numericStringSubstringsMatch,
    integerOrderingMatch,
    integerMatch,
    octetStringMatch,
    caseIgnoreMatch,
    caseIgnoreSubstringsMatch,
FROM X.520

-- provider object-class

id-oc OBJECT IDENTIFIER::=  {ccitt (0) identified-organisation (4) etsi (0) inDomain (1)
                    upt (1) object-class (0)}
id-at OBJECT IDENTIFIER::=  {ccitt (0) identified-organisation (4) etsi (0) inDomain (1)
                    upt (1) attribute (1)}
id-nf OBJECT IDENTIFIER::=  {ccitt (0) identified-organisation (4) etsi (0) inDomain (1)
            upt (1) name-form (2)}


uptProvider  OBJECT-CLASS ::= {
    SUBCLASS OF {top}
    MUST CONTAIN {
        providerId}
    ID  id-oc-uptProvider}

providerId  ATTRIBUTE ::= {
    WITH SYNTAX NumericString (SIZE (1..ub-provider-id))
    EQUALITY MATCHING RULE  numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    SINGLE VALUE    TRUE
    ID  id-at-providerId}

-- agreed service object class

agreedService  OBJECT-CLASS ::= {
    SUBCLASS OF {top}
    MUST CONTAIN {
        providedServiceId|
        providedLocations}
    ID  id-oc-agreedService}

providedServiceId  ATTRIBUTE ::= {
    WITH SYNTAX Service
    SINGLE VALUE    TRUE
    ID  id-at-providedServiceId}

Service ::=INTEGER {
    isdnTelephony (0),
    icRegistration (10),
    serviceProfileModification (20),
    standard (30),
    callForwardingUnconditional (40),
    callForwardingOnNoReply (41),
    callForwardingOnBusy (42),
    variableRouting (43)}
```

```
providedLocations  ATTRIBUTE ::= {
    WITH SYNTAX NumericString (SIZE (1..ub-locations))
    EQUALITY MATCHING RULE  numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringMatch
    ID  id-at-providedLocations}

-- user profile object class

userProfile  OBJECT-CLASS ::= {
    SUBCLASS OF {top}
    MUST CONTAIN {
        pui|
        chargingAttributeSet|
        allowedServices|
        allowedCFParameters|
        nbOfFailedAuthentications}
    MAY CONTAIN {
        userPassword|
        callInfoRecords|
        specialPassword|
        variablePassword}
    ID  id-oc-userProfile}

pui  ATTRIBUTE ::= {
    WITH SYNTAX         NumericString (SIZE (1..ub-pui))
    EQUALITY MATCHING RULE  numericStringMatch
    SINGLE VALUE        TRUE
    ID              id-at-pui}

specialPassword  ATTRIBUTE ::={
    WITH SYNTAX         OCTET STRING (SIZE(0..ub-special-password))
    EQUALITY MATCHING RULE  octetStringMatch
    ID              id-at-specialPassword}

variablePassword  ATTRIBUTE ::={
    WITH SYNTAX         OCTET STRING (SIZE(0..ub-variable-password))
    EQUALITY MATCHING RULE  octetStringMatch
    ID              id-at-variablePassword}

nbOfFailedAuthentications  ATTRIBUTE ::= {
    WITH SYNTAX INTEGER (SIZE (1..ub-max-number-of-failed-authentications))
    ORDERING MATCHING RULE  integerOrderingMatch
    SINGLE VALUE        TRUE
    ID              id-at-nbOfFailedAuthentications}

chargingAttributeSet  ATTRIBUTE ::= {
    defaultChargingReference|
    userCredit|
    activeChargingService}

defaultChargingReference  ATTRIBUTE ::= {
    WITH SYNTAX         IsdnAddress
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    SINGLE VALUE            TRUE
    ID              id-at-defaultChargingReference}

IsdnAddress ::=NumericString (SIZE(1.. ub-international-isdn-number))

userCredit  ATTRIBUTE ::= {
    WITH SYNTAX         INTEGER (SIZE (1..ub-max-user-credit))
    ORDERING MATCHING RULE  integerOrderingMatch
    SINGLE VALUE        TRUE
    ID              id-at-userCredit}

callInfoRecords  ATTRIBUTE ::= {
    WITH SYNTAX CallInfoRecord
    ID      id-at-callInfoRecords}

CallInfoRecord ::= SEQUENCE {
    authenticationTime  [2] UTCTime,
    callStopTimeValue   [3] UTCTime,
    callStartTimeValue  [4] UTCTime,
    callingAddressValue [6] IsdnAddress ,
    calledNumber        [7] IsdnAddress,
    duration        [5] INTEGER (0..2147483647) OPTIONAL,
    visitedNetwork      [0] NetworkCode OPTIONAL,
    callCost        [1] Cost OPTIONAL,
    routingAddress      [8] IsdnAddress OPTIONAL,
    reroutingAddress    [9] IsdnAddress OPTIONAL,
    invokedSupplementaryServices    [10] CFServices OPTIONAL,
    surcharges      [11] Cost OPTIONAL,
    releaseCause        [12] Cause OPTIONAL}
```

```
Cost ::= CHOICE {
    pulse   [0] INTEGER (1..ub-pulse),
    cost    [1] CurrencyValue}
CurrencyValue::= SEQUENCE {
    amount  [0] INTEGER (0..ub-amount),
    currency    [1] Currency}
Currency ::= ENUMERATED {
    usDollar (0),
    frenchFranc (1),
    germanMark (2),
    dutchGuilder (3),
    italianLira (4),
    englishPound (5),
    spanishPeseta (6),
    swedishKrone (7)}
CFServices ::= SET OF Service(40..49)
Cause ::= OCTET STRING (SIZE(lb-causeLength..ub-causeLength))

activeChargingService  ATTRIBUTE ::= {
    WITH SYNTAX     Service (30..39)
    SINGLE VALUE    TRUE
    ID          id-at-activeChargingService}

allowedServices  ATTRIBUTE ::= {
    WITH SYNTAX         Service
    EQUALITY MATCHING RULE  integerMatch
    ID              id-at-allowedServices}

allowedCFParameters  ATTRIBUTE ::= {
    WITH SYNTAX         CFParameter
    EQUALITY MATCHING RULE  integerMatch
    ID              id-at-allowedCFParameters}

CFParameter ::= INTEGER {
    notifyActivation (0),
    notifyForwarding (1),
    notifyCallingPartyWithNumber (2),
    notifyCallingPartyWithoutNumber (3),
    notifyForwardedTo (4)}

-- user profile alias

userProfileAlias  OBJECT-CLASS ::= {
SUBCLASS OF alias
MUST CONTAIN {
    uptNumber}
ID  id-oc-userProfile}
uptNumber  ATTRIBUTE ::= {
    WITH SYNTAX             IsdnAddress
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    SINGLE VALUE                TRUE
    ID                  id-at-uptNumber}

-- basic service object class

basicService  OBJECT-CLASS ::= {
    SUBCLASS OF {top}
    MUST CONTAIN {
        basicServiceId|
        icRegistrationAddress|
        allowedDestinations|
        allowedRegistrationAddress}
    ID  id-oc-basicService}

basicServiceId  ATTRIBUTE ::= {
    WITH SYNTAX         Service(0..9)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE            TRUE
    ID              id-at-basicServiceId}

icRegistrationAddresses  ATTRIBUTE ::= {
    WITH SYNTAX             IsdnAddress
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID                  id-at-icRegistrationAddresses}

allowedDestinations  ATTRIBUTE ::= {
    WITH SYNTAX         NumericString (SIZE (1..ub-locations))
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID                  id-at-allowedDestinations}
```

```
allowedRegistrationAddress  ATTRIBUTE ::= {
    WITH SYNTAX         NumericString (SIZE (1..ub-locations))
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID                  id-at-allowedRegistrationAddress}

-- registered routing service

registeredRoutingService  OBJECT-CLASS ::={
    SUBCLASS OF {top}
    MUST CONTAIN {
        routingServiceId|
        routingAddress|
        activationStatus}
    MAY CONTAIN {
        activatedCFParameters|
        noReplyConditionTimer}
    ID  id-oc-registeredRoutingService}

routingServiceId  ATTRIBUTE ::= {
    WITH SYNTAX         Service(40..49)
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE            TRUE
    ID              id-at-routingServiceId}

routingAddress  ATTRIBUTE ::= {
    WITH SYNTAX             IsdnAddress
    EQUALITY MATCHING RULE      numericStringMatch
    SUBSTRINGS MATCHING RULE    numericStringSubstringsMatch
    ID                  id-at-routingAddress}

activationStatus  ATTRIBUTE ::= {
    WITH SYNTAX         ActivationStatus
    EQUALITY MATCHING RULE  integerMatch
    SINGLE VALUE        TRUE
    ID              id-at-activationStatus}

ActivationStatus ::= INTEGER {
    notActivated (0),
    activated (1)}

activatedCFParameters  ATTRIBUTE ::= {
    WITH SYNTAX         CFParameter
    EQUALITY MATCHING RULE  integerMatch
    ID              id-at-allowedCFParameters}

noReplyConditionTimer  ATTRIBUTE ::= {
    WITH SYNTAX         INTEGER
    ORDERING MATCHING RULE  integerOrderingMatch
    ID              id-at-noReplyConditionTimer}

-- name forms

uptProviderNameForm  NAME-FORM ::= {
    NAMES       uptProvider
    WITH ATTRIBUTES {providerId}
    ID          id-nf-uptProviderNameForm}

agreedServiceNameForm  NAME-FORM ::= {
    NAMES       agreedService
    WITH ATTRIBUTES {providedServiceId}
    ID          id-nf-agreedServiceNameForm}

userProfileNameForm  NAME-FORM ::= {
    NAMES       userProfile
    WITH ATTRIBUTES {pui}
    ID          id-nf-userProfileNameForm}

userProfileAliasNameForm  NAME-FORM ::= {
    NAMES       userProfileAlias
    WITH ATTRIBUTES {uptNumber}
    ID          id-nf-userProfileAliasNameForm}

basicServiceNameForm  NAME-FORM ::= {
    NAMES       basicService
    WITH ATTRIBUTES {basicServiceId}
    ID          id-nf-basicServiceNameForm}

registeredRoutingServiceNameForm  NAME-FORM ::= {
    NAMES       registeredRoutingService
    WITH ATTRIBUTES {routingServiceId}
    ID          id-nf-registeredRoutingServiceNameForm}

-- structure rules
```

```
sr1 STRUCTURE-RULE ::= {
    NAME FORM   uptProviderNameForm
    ID      1}

sr2 STRUCTURE-RULE::= {
    NAME FORM       uptProviderNameForm
    SUPERIOR RULES  {sr1}
    ID          2}

sr3 STRUCTURE-RULE::= {
    NAME-FORM       userProfileNameForm
    SUPERIOR RULES  {sr1}
    ID          3}

sr4 STRUCTURE-RULE::= {
    NAME FORM       userProfileAliasNameForm
    SUPERIOR RULES  {sr1}
    ID          4}

sr5 STRUCTURE-RULE::= {
    NAME FORM       agreedServiceNameForm
    SUPERIOR RULES  {sr2}
    ID          5}

sr6 STRUCTURE-RULE::= {
    NAME FORM       basicServiceNameForm
    SUPERIOR RULES  {sr3}
    ID          6}

sr7 STRUCTURE-RULE::= {
    NAME FORM       registeredRoutingServiceNameForm
    SUPERIOR RULES  {sr6}
    ID          7}

-- attributes
id-at-providerId OBJECT IDENTIFIER ::= {id-at 2}
id-at-providedServiceId OBJECT IDENTIFIER ::= {id-at 3}
id-at-providedLocations OBJECT IDENTIFIER ::= {id-at 4}
id-at-pui OBJECT IDENTIFIER ::= {id-at 5}
id-at-specialPassword OBJECT IDENTIFIER ::= {id-at 7}
id-at-variablePassword OBJECT IDENTIFIER ::= {id-at 8}
id-at-nbOfFailedAuthentications OBJECT IDENTIFIER ::= {id-at 9}
id-at-userCredit OBJECT IDENTIFIER ::= {id-at 10}
id-at-defaultChargingReference OBJECT IDENTIFIER ::= {id-at 11}
id-at-callInfoRecords OBJECT IDENTIFIER ::= {id-at 12}
id-at-allowedServices OBJECT IDENTIFIER ::= {id-at 13}
id-at-activeChargingService OBJECT IDENTIFIER ::= {id-at 14}
id-at-allowedCFParameters OBJECT IDENTIFIER ::= {id-at 15}
id-at-basicServiceId OBJECT IDENTIFIER ::= {id-at 16}
id-at-icRegistrationAddresses OBJECT IDENTIFIER ::= {id-at 17}
id-at-routingServiceId OBJECT IDENTIFIER ::= {id-at 18}
id-at-activationStatus OBJECT IDENTIFIER ::= {id-at 19}
id-at-routingAddress OBJECT IDENTIFIER ::= {id-at 20}
id-at-activatedCFParameters OBJECT IDENTIFIER ::= {id-at 21}
id-at-noReplyConditionTimer OBJECT IDENTIFIER ::= {id-at 22}
id-at-uptNumber OBJECT IDENTIFIER ::= {id-at 23}

SupportedAttributes ATTRIBUTE ::={
    objectClass|
    aliasedEntryName|
    providerId|
    providedServiceId|
    providedLocations|
    pui |
    userPassword|
    specialPassword|
    variablePassword|
    nbOfFailedAuthentications|
    userCredit|
    defaultChargingReference|
    callInfoRecords|
    allowedServices|
    activeChargingService|
    allowedCFParameters|
    basicServiceId|
    icRegistrationAddresses|
    routingServiceId|
    activationStatus|
    routingAddress|
    activatedCFParameters|
    noReplyConditionTimer|
    uptNumber }
```

```
-- object classes
id-oc-uptProvider  OBJECT IDENTIFIER ::= {id-oc 2}
id-oc-partner  OBJECT IDENTIFIER ::= {id-oc 3}
id-oc-agreedService  OBJECT IDENTIFIER ::= {id-oc 4}
id-oc-userProfile  OBJECT IDENTIFIER ::= {id-oc 5}
id-oc-userProfileAlias  OBJECT IDENTIFIER ::= {id-oc 6}
id-oc-basicService  OBJECT IDENTIFIER ::= {id-oc 7
id-oc-registeredRoutingService  OBJECT IDENTIFIER ::= {id-oc 8}

--name forms
id-nf-uptProviderNameForm  OBJECT IDENTIFIER ::= {id-nf 0}
id-nf-partnerNameForm  OBJECT IDENTIFIER ::= {id-nf 1}
id-nf-agreedServiceNameForm  OBJECT IDENTIFIER ::= {id-nf 2}
id-nf-userProfileNameForm  OBJECT IDENTIFIER ::= {id-nf 3}
id-nf-userProfileAliasNameForm  OBJECT IDENTIFIER ::= {id-nf 4}
id-nf-basicServiceNameForm  OBJECT IDENTIFIER ::= {id-nf 5}
id-nf-registeredRoutingServiceNameForm  OBJECT IDENTIFIER ::= {id-nf 6}

--upper bounds
ub-provider-id  INTEGER ::= 6
ub-locations  INTEGER ::= 15
ub-pui  INTEGER ::= 15
ub-user-password  INTEGER ::= 128
ub-special-password  INTEGER ::= 128
ub-variable-password  INTEGER ::= 128
ub-max-number-of-failed-authentications INTEGER ::= 6
ub-international-isdn-number  INTEGER ::= 15
ub-max-user-credit  INTEGER ::= 4096
ub-pulse  INTEGER ::= 32768
ub-amount  INTEGER ::= 4096
lb-causeLength  INTEGER ::= 2
ub-causeLength  INTEGER ::= 128

END
```

## History

| Document history | | | |
|---|---|---|---|
| August 1996 | Public Enquiry | PE 111: | 1996-08-05 to 1996-11-29 |
| | | | |
| | | | |
| | | | |
| | | | |