

TECHNICAL REPORT

ISO/IEC TR 11590

First edition
1995-10-01

Information technology — Open Systems Interconnection — LOTOS description of the CCR protocol

*Technologies de l'information — Interconnexion de systèmes ouverts
(OSI) — Description LOTOS du protocole CCR*



Reference number
ISO/IEC TR 11590:1995(E)
AVA Documents
Document Sharing Hub

Contents

1	Scope	1
2	Normative references	1
3	Definitions	1
4	Symbols and abbreviations	1
5	Conventions	2
6	Introduction to the formal description	2
6.1	Model	2
6.2	Structure	3
7	Interface of the description of CCR protocol	12
8	Type definitions	12
8.1	General Type	12
8.1.1	Key Type	13
8.1.2	Formal Identifier Type	13
8.1.3	Atomic Action Identifier	14
8.1.4	Branch Identifier	14
8.1.5	Masters Name	15
8.1.6	Superiors Name	15
8.1.7	Name Type	15
8.1.8	AE Title Type	16
8.1.9	AP Title Type	16
8.1.10	AE Qualifier Type	17

© ISO/IEC 1995

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

8.2	Side Type	17
8.2.1	Atomic Action Suffix Type	17
8.2.2	Branch Suffix Type	17
8.2.3	Recovery State Type	18
8.2.4	Result	18
8.2.5	Result Source	19
8.2.6	Diagnostic	19
8.2.7	Other Parameters	19
8.2.8	Synchronization Type	20
8.2.9	Synchronization Point Serial Number	20
8.2.10	Re-Synchronization Type	20
8.2.11	Token Type	21
8.2.12	Presentation Context Identifier List	21
8.2.13	Data Separation Type	21
8.2.14	CCR Protocol Data Unit User Data	22
8.2.15	Transformation Between User_Data and CCR_User_Data	22
9	Service Primitive Definitions	22
9.1	CCR Service Primitives Type	22
9.2	ACSE Service Primitives Type	27
9.3	Presentation Service Primitive Definition	29
9.4	General Service Primitive Type	32
10	PDU Types	33
10.1	CCR PDU Type	33
10.2	Application Protocol Data Unit	35
10.3	Transformation from CCR Service Primitive to CCR PDU	36
10.4	Transformation from CCR PDU to CCR Service Primitive	38
10.5	Key Queue Type	39
10.6	PDU Type	40
10.7	PDU Queue Type	40
10.8	Search CCR PDU Type	41
10.9	Concatination and Deconcatination Type	41
10.10	CCR Version Number Types	43

10.11 Atomic Action Branch Type	44
10.12 Failure Event Type	45
11 Global constraints of the CCR protocol specification behaviour	45
12 CCR Protocol Machine	45
12.1 CCR Protocol Version Negotiation	46
12.2 Process for CCR Initialize	46
12.3 CCR Fixed Version Process	47
12.4 Process for CCR Behaviour	47
13 Control Function	57
13.1 CF Router	57
13.2 CCR Part in CF	57
13.3 CCR Initializer in CF for Non-Negotiated Version Case	58
13.4 CCR Initializer in CF for Negotiated Version Case	59
13.5 Concatenator and Deconcatenator Process	63
13.5.1 Concatenator Process	63
13.5.2 Deconcatenator Process	64
13.6 Lower Service-PDU Mapping Process	65

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The main task of technical committees is to prepare International Standards. In exceptional circumstances a technical committee may propose the publication of a Technical Report of one of the following types:

- type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;
- type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;
- type 3, when a technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

ISO/IEC TR 11590, which is a Technical Report of type 2, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 21, *Open systems interconnection, data management and open distributed processing*.

Introduction

This Technical Report describes ISO/IEC 9805-1 CCR Protocol Specification using LOTOS.

LOTOS is a formal description technique(FDT) to define behavior of systems with a formal syntax and semantics.

Aim of this Technical Report is to give an unambiguous and precise description of ISO/IEC 9805-1.

However, in case of inconsistency between ISO/IEC 9805-1 and this Technical Report, ISO/IEC 9805-1 takes precedence over this Technical Report.

This Technical Report consists of as follows:

- In clauses 5, 6, and 7, the description that is, the structures of processes, types, and gates, are explained.
- In clause 8, the types which are used in the description are defined.
- In clause 11, global constraints of the CCR protocol specification are described.
- In clause 12, a CCR protocol machine is described.
- In clause 13, CF (control function) is described.

Information technology — Open Systems Interconnection — LOTOS description of the CCR protocol

1 Scope

This Technical Report describes ISO/IEC 9805-1 CCR Protocol Specification using formal description language LOTOS. The LOTOS specification includes not only CCR protocol machine, but also CF(control function) which controls the ASEs in order that the architecture of the CCR-ASE process is compatible with the ALS structure.

According to the ALS structure, this Technical Report employs the model as shown in Clause 6. LOTOS Specification covers CF(control function) as well as CCR Protocol machine, however, the description of CF includes only its functions concerning on CCR operations.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this Technical Report. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Technical Report are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 9804:1994, *Information technology – Open Systems Interconnection – Service definition for the Commitment, Concurrency and Recovery service element.*

ISO/IEC 9805-1:1994, *Information technology – Open Systems Interconnection – Protocol for the Commitment, Concurrency and Recovery service element: Protocol Specification.*

ISO 8807:1989, *Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour.*

3 Definitions

For the purposes of this Technical Report the definitions given in ISO/IEC 9805-1 apply.

4 Symbols and abbreviations

The abbreviations given in ISO/IEC 9805-1 apply.

The abbreviation “LOTOS” is defined in ISO 8807.

5 Conventions

Clause 7 through 13 of this Technical Report consists of a LOTOS text. All explanatory descriptions are caught between “(*)” and “*)” symbols and are treated as comments of the LOTOS text.

6 Introduction to the formal description

6.1 Model

CCR protocol in this Technical Report is modeled as shown in Figure 1.

This Technical Report specifies the specification `CCR_Protocol_in_ALS` which has five gates, `u`, `acse_u`, `acse_l`, `u_env`, and `env`, and one parameter `ver`.

`CCR_Protocol_in_ALS` has two processes, `CCR` and `CF`. Process `CCR` is actually a protocol machine for the CCR operation. `CF` is a process to perform a control functions to be compatible with the ALS structure.

`CF` includes three processes, `CFRouter`, `ConcDeconcProc`, and `LowerMapper`. `CFRouter` controls process `CCR` via two gates, `ccr_u` and `ccr_l`. `CFRouter` should also support the behavior for ACSE or other parts, but those parts are not essential for the CCR specification. Therefore, they are omitted in this specification. Instead, ACSE can communicate via two gates, `acse_u` and `acse_l`.

`ConcDeconcProc` performs to concatenate and deconcatenate the PDUs for CCR and other ASEs and send it to the presentation layer via process `LowerMapper`.

Process `LowerMapper` maps in between the PDUs concatenated by `ConcDeconcProc` and the presentation services.

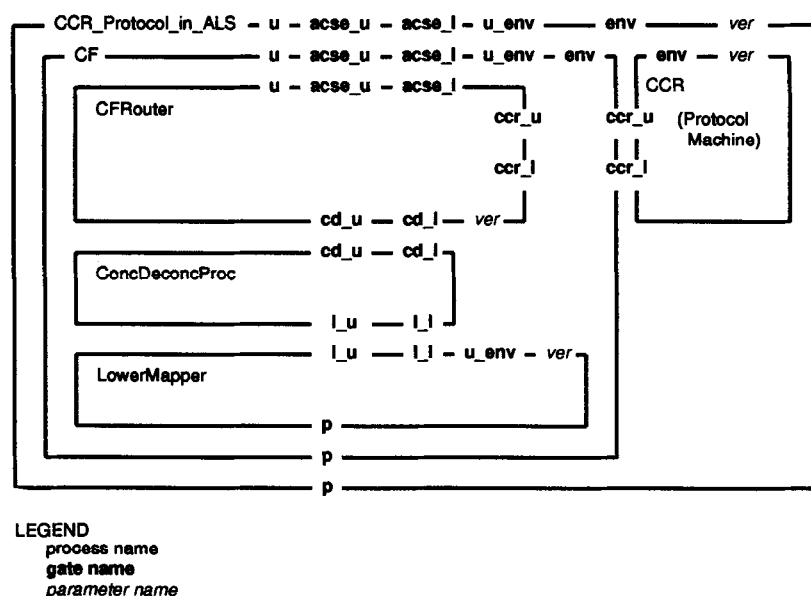


Figure 1 - CCR protocol model

6.2 Structure

The relationships among the processes and the types are shown in Figure 2 and 3, respectively. In these figures, “*” means that “the definition has already been listed above”.

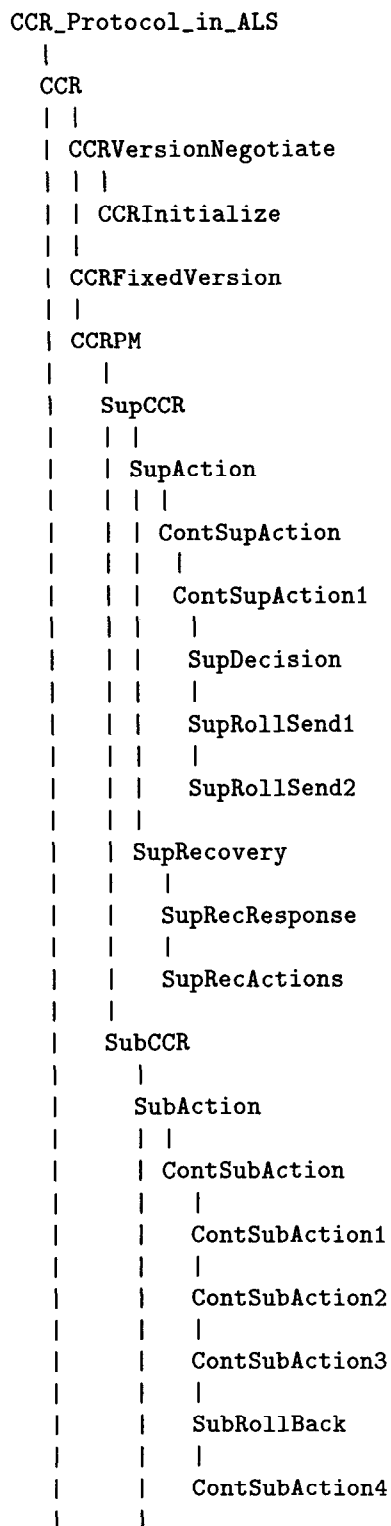


Figure 2 - Relationship among the processes

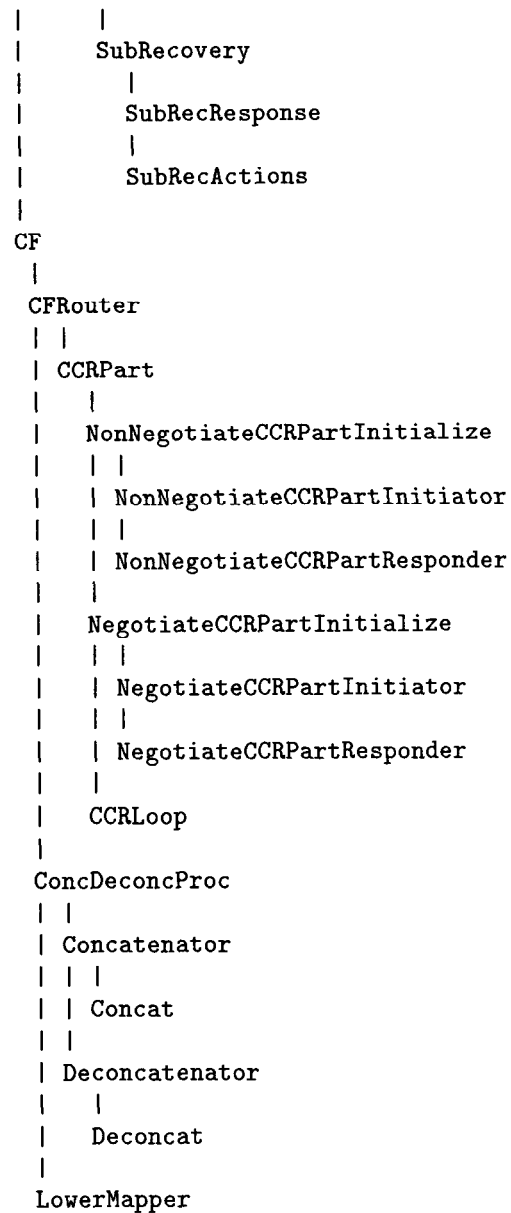


Figure 2 - Relationship among the processes (concluded)

```

Atomic_Action_Identifier
|
| F_Atomic_Action_Identifier
| |
| | F_Identifier
| | |
| | | F_Name
| | | |
| | | | Element*
| | | |
| | | F_Suffix
| | | |
| | | | Element*
|
| Atomic_Action_Suffix
| |
| | General
| | |
| | | Boolean*
|
| Masters_Name
| |
| | Name
| | |
| | | AE_title
| | | |
| | | | AP_title
| | | | |
| | | | | General*
| | | | |
| | | | AE_qualifier
| | | | |
| | | | | General*
| | | | |
| | | | Boolean*
| | | |
| | | Side
| | | |
| | | | Boolean*
| | | |
| | | Boolean*
| | | |
| | | CCR_Version_Number
| | | |
| | | | Boolean*
| | | |
| | | Key
| | | |
| | | | Boolean*
|
| Boolean*

```

Figure 3 - Relationship among the types

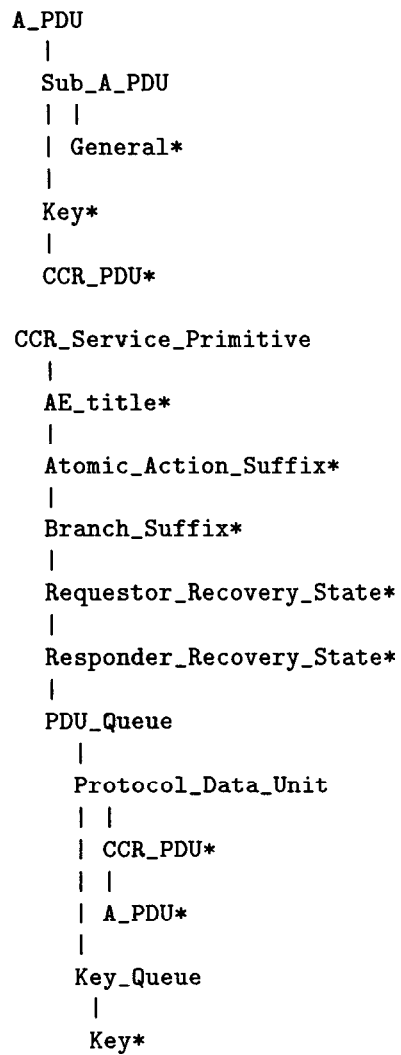
```

Branch_Identifier
|
F_Branch_Identifier
| |
| F_Identifier*
|
Branch_Suffix
| |
| General*
|
Superiors_Name
| |
| Name*
|
Boolean*

CCR_PDU
|
CCR_Version_Number
| |
| Boolean*
| |
| Key*
|
Atomic_Action_Identifier*
|
Branch_Identifier*
|
Branch_Suffix*
|
Requestor_Recovery_State
| |
| Key*
|
Responder_Recovery_State
| |
| Key*
|
User_Data
| |
| General*
|
Key*

```

Figure 3 - Relationship among the types (continued)

**Figure 3 - Relationship among the types (continued)**

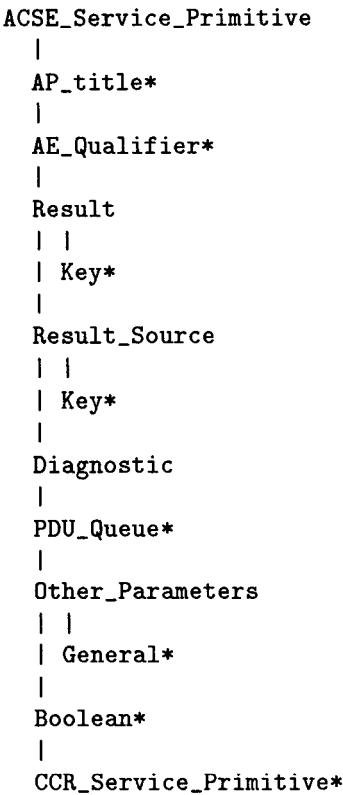


Figure 3 - Relationship among the types (continued)

```

Presentation_Service_Primitive
|
| Sync_Type
| |
| | Key*
|
| Sync_Point_Number
| |
| | General*
|
| Resync_Type
| |
| | Key*
|
| Token
| |
| | General*
|
| P_Cont_Id_List
| |
| | General*
|
| Data_Separation
| |
| | Boolean*
|
| PDU_Queue*
|
| Key*
|
| Boolean*
|
| ACSE_Service_Primitive*

Service_Primitive
|
| CCR_Service_Primitive*
|
| ACSE_Service_Primitive*
|
| Presentation_Service_Primitive*

```

Figure 3 - Relationship among the types (continued)

```
CCR_Service_Primitive_To_CCR_PDU
|
CCR_PDU*
|
CCR_Service_Primitive*
|
AE_Title*
|
Transformation_User_Data
|
PDU_Queue*
|
User_Data*

CCR_PDU_to_CCR_Service_Primitive
|
CCR_PDU*
|
CCR_Service_Primitive*
|
AE_Title*
|
Transformation_User_Data*

Search_CCR_PDU
|
PDU_Queue*

ConcDeconcType
|
Protocol_Data_Unit*
|
PDU_Queue*

Atomic_Action_Branch
|
Atomic_Action_Identifier*
|
Branch_Identifier*
|
CCR_PDU_To_CCR_Service_Primitive

Failure
```

Figure 3 - Relationship among the types (concluded)

7 Interface of the description of CCR protocol

CCR protocol is represented in terms of type definitions and process definitions. The static attribute, i.e. the type of the service primitives and their parameters are described in type definitions. The behaviour, i.e. the protocol behaviours issuing/receiving procedures are described in process definitions.

Issuing and receiving of protocol data units are represented as the events at the gates.

The model employed in this Technical Report is shown in Figure 1.

Each gate is used as follows:

u	: to communicate with the CCR user
p	: to communicate with the presentation layer
acse_u	: to communicate with ACSE (from ACSE to CCR)
acse_l	: to communicate with ACSE (from CCR to ACSE)
u_env	: to communicate with an environment of a CCR user, such information as major or minor synchronization points or tokens
env	: to communicate with an ALS environment, especially for the failure information

The parameter *ver* is a CCR version number, which is a constant either *ver1*, *ver2*, *ver12*, or *rejected*.

Formal data types, *Boolean* and *Element*, are imported from the LOTOS standard library.

```
specification CCR_Protocol_in_ALS[u,p,acse_u,acse_l,env,u_env]
                (ver:ccr_ver) : exit
```

```
library
    Boolean,Element
endlib
```

8 Type definitions

8.1 General Type

This type is a general type. There are some types which have a structure, but whose structure is not important in this LOTOS description. In such a case, this type is used as a place holder.

```
type General is Boolean
sorts general
opns
    null : -> general
    succ : general -> general
    _eq_,_ne_ : general,general -> Bool
eqns
forall x,y:general
ofsort Bool
    null eq null = true;
    null eq succ(x) = false;
    succ(x) eq null = false;
    succ(x) eq succ(y) = x eq y;
```

```

      x ne y = not(x eq y);
endtype (* General *)

```

8.1.1 Key Type

This type is used for identification which is necessary in other types.

```

type      Key is Boolean
sorts     key
opns
    0 : -> key
    succ : key -> key
    _eq_, _ne_, _le_ : key, key -> Bool
eqns
forall    x,x1,y,y1:key
ofsort    Bool
    0 eq 0 = true;
    0 eq succ(x) = false;
    succ(x) eq 0 = false;
    succ(x) eq succ(y) = x eq y;
    x ne y = not(x eq y);
    0 le 0 = true;
    0 le succ(x) = true;
    succ(x) le succ(y) = x le y;
    succ(x) le 0 = false;
endtype (* Key *)

```

8.1.2 Formal Identifier Type

This type is a formal type for the definition of “Atomic Action Identifier” and “Branch Identifier”. These two types have the same structure. Therefore, this formal type is useful for the definitions as a template.

```

type      F_Name is Element renamedby
sortnames
    f_name for Element
endtype (* F_Name *)

type      F_Suffix is Element renamedby
sortnames
    f_suffix for Element
endtype (* F_Suffix *)

type      F_Identifier is F_Name, F_Suffix
sorts
    f_identifier
opns
    f_identifier : f_name, f_suffix -> f_identifier
    get_fn : f_identifier -> f_name
    get_fs : f_identifier -> f_suffix
    _eq_, _ne_ : f_identifier, f_identifier -> FBool
eqns
forall    fn,fn1:f_name,fs,fs1:f_suffix,f,f1:f_identifier

```

```

ofsort  f_name
        get_fn(f_identifier(fn,fs)) = fn;
ofsort  f_suffix
        get_fs(f_identifier(fn,fs)) = fs;
ofsort  FBool
        f eq f1 = (get_fn(f) eq get_fn(f1)) and (get_fs(f) eq get_fs(f1));
        f ne f1 = not(f eq f1);
endtype (* F_Identifier *)

```

8.1.3 Atomic Action Identifier

This type consists of the pair of “Masters Name” and “Atomic Action Suffix”. Formal Identifier Type is used.

```

type    F_Atomic_Action_Identifier is F_Identifier renamedby
sortnames
        atomic_action_identifier for f_identifier
opnnames
        atomic_action_identifier for f_identifier
        get_mn for get_fn
        get_aas for get_fs
endtype (* F_Atomic_Action_Identifier *)

type    Atomic_Action_Identifier is F_Atomic_Action_Identifier actualizedby
        Atomic_Action_Suffix,Masters_Name,Boolean using
sortnames
        masters_name for f_name
        atomic_action_suffix for f_suffix
        Bool for FBool
endtype (* Atomic_Action_Identifier *)

```

8.1.4 Branch Identifier

This type consists of the pair of “Superiors Name” and “Branch Suffix”. Formal Identifier Type is used.

```

type    F_Branch_Identifier is F_Identifier renamedby
sortnames
        branch_identifier for f_identifier
opnnames
        branch_identifier for f_identifier
        get_sn for get_fn
        get_bs for get_fs
endtype (* F_Branch_Identifier *)

type    Branch_Identifier is F_Branch_Identifier actualizedby
        Branch_Suffix,Superiors_Name,Boolean using
sortnames
        superiors_name for f_name
        branch_suffix for f_suffix
        Bool for FBool
endtype (* Branch_Identifier *)

```

8.1.5 Masters Name

This type is constructed by the renaming of Name Type. This type and Superiors Name Type have the same structure.

```

type    Masters_Name is Name renamedby
sortnames
    masters_name for name
opnnames
    masters_name for name
endtype (* Masters_Name *)

```

8.1.6 Superiors Name

This type is constructed from Name Type. This type and Masters Name Type have the same structures.

```

type    Superiors_Name is Name renamedby
sortnames
    superiors_name for name
opnnames
    superiors_name for name
endtype (* Superiors_Name *)

```

8.1.7 Name Type

This is a general Name Type, and is used for the definition of Masters Name and Superiors Name. This type is composed of AE Title and Side.

```

type    Name is AE_title,Side,Boolean,CCR_Version_Number
sorts   name
opns
    name : ae_title -> name
    name : side -> name
    get_aet : name -> ae_title
    get_s : name -> side
    is_aet : name -> Bool
    is_s : name -> Bool
    _eq_,_ne_ : name,name -> Bool
    get_aet: name, ccr_ver, ae_title, ae_title -> ae_title
eqns
forall  aet,sender_aet,receiver_aet:    ae_title,
    s,s1:    side,
    n,n1:    name,
    ver:    ccr_ver
ofsort  Bool
    is_aet(name(aet)) = true;
    is_aet(name(s)) = false;
    is_s(name(aet)) = false;
    is_s(name(s)) = true;
ofsort  AE_title
    get_aet(name(aet)) = aet;

```

```

    get_aet(name(aet),ver,receiver_aet,sender_aet) = aet;
    get_aet(name(receiver),ver2,receiver_aet,sender_aet) = receiver_aet;
    get_aet(name(sender),ver2,receiver_aet,sender_aet) = sender_aet;
ofsort side
    get_s(name(s)) =s;
ofsort Bool
    is_aet(n) and is_aet(n1) =>
        n eq n1 = get_aet(n) eq get_aet(n1);
    is_s(n) and is_s(n1) =>
        n eq n1 = get_s(n) eq get_s(n1);
    is_aet(n) and is_s(n1) =>
        n eq n1 = false;
    is_s(n) and is_aet(n1) =>      n eq n1 = false;
    n ne n1 = not(n eq n1);
endtype (* Name *)

```

8.1.8 AE Title Type

AE Title is an important name in OSI Application Layer, which is composed of AP Title and AE Qualifier.

```

type    AE_title is AP_title,AE_qualifier,Boolean
sorts   ae_title
opns
    ae_null : -> ae_title
    aet : ap_title,ae_qualifier -> ae_title
    get_ap : ae_title -> ap_title
    get_aeq : ae_title -> ae_qualifier
    _eq_,_ne_ : ae_title,ae_title -> Bool
eqns
forall  apt,apt1:ap_title,aeq,aeq1:ae_qualifier,aet,aet1:ae_title
ofsort  ap_title
    get_ap(aet(apt,aeq)) = apt;
ofsort  ae_qualifier
    get_aeq(aet(apt,aeq)) = aeq;
ofsort  ae_title
    ae_null = aet(ap_null,aq_null);
ofsort  Bool
    aet(apt,aeq) eq aet(apt1,aeq1) = (apt eq apt1) and (aeq eq aeq1);
    aet ne aet1 = not(aet eq aet1);
endtype (* AE_title *)

```

8.1.9 AP Title Type

AP Title is defined in the Technical Corrigendum of OSI ACSE , which is Name defined by Directory Service or ASN1 OCTET IDENTIFIER. But, in this description, its structure is not important. AP Title is only used as a place holder.

```

type    AP_title is General renamedby
sortnames
    ap_title for general
opnnames
    ap_null for null

```

```
endtype (* AP_title *)
```

8.1.10 AE Qualifier Type

AP Title is defined in the Technical Corrigendum of OSI ACSE , which is RelativeDistinguishedName defined by Directory Service or ASN1 INTEGER. But, in this description, its structure is not important. AP Title is only used as a place holder.

```
type    AE_qualifier is General renamedby
sortnames
    ae_qualifier for general
opnnames
    aq_null for null
endtype (* AE_qualifier *)
```

8.2 Side Type

Side is used in CCR version 2. It includes “sender” or “receiver”. The notation of side depends on AE Titles which are transferred in the establishment of an association. It is used in CCR PDU.

```
type    Side is Boolean
sorts    side
opns
    sender,receiver : -> side
    _eq_,_ne_ : side,side -> Bool
eqns
    fororall s1, s2 : side
ofsort Bool
    sender eq sender = true;
    receiver eq sender = false;
    sender eq receiver = false;
    receiver eq receiver = true;
    s1 ne s2 = not(s1 eq s2);
endtype (* Side *)
```

8.2.1 Atomic Action Suffix Type

This type is a kind of identifier of Atomic Action, which is more local than Atomic Action Identifier. In this description, it is not important and is used as a place holder. It is used in CCR Service Primitive and CCR PDU.

```
type    Atomic_Action_Suffix is General renamedby
sortnames
    atomic_action_suffix for general
endtype (* Atomic_Action_Suffix *)
```

8.2.2 Branch Suffix Type

This type is a kind of identifier of Branch, which is more local than Branch Identifier. In this description, it is not important and is used as a place holder. It is used in CCR Service Primitive and CCR PDU.