

Technical Notes - PFA SNMP Traps

Disclaimer

No part of this document may be copied, reproduced, disclosed, transferred or reduced to any electronic medium or machine readable form without the express written approval of Ericsson Business Networks.

(c) Copyright 1995 by Ericsson Business Networks.

1. Introduction

This document lists SNMP TRAPS generated by the PFA. For a more formal description of these traps and the circumstances under which the PFA will generate each trap refer to [1].

Since SNMP traps are transported over UDP there is a possibility of a trap not reaching its destination. In order to assist an NMS to detect lost traps, every trap from the PFA will include a trap sequence number which is unique to each IP host configured to receive traps from the PFA. For traps specific to an interface, this parameter follows the ifIndex value. For all other traps, this parameter is sent as the first value..

1.1. Abbreviations

DNA	Dynamic Network Administration
MIB	Management Information Base
SNMP	Simple Network Management Protocol

2. Interface MIB objects

These objects are associated with an interface and will be part of the interfaces group MIB.

For each interface object, if enabled, a trap will be sent for each manually initiated state change or configuration change (manually deblocked, manually blocked, initialised or terminated) in the object (see the state transition diagram below).

The object state change traps will take the following form:

```
pfaObjectDeBlocked(ifIndex, ifName)
pfaObjectBlocked(ifIndex, ifName, ifDescr)
```

There will also be traps which inform the manager when an object is created or deleted. These object-configuration traps are configurable on a per-object basis. These traps are as defined in the EBC DNA MIB [2] and take the form shown below:

```
dnaInterfaceCreate(ifIndex, ifName, objectId)
dnaInterfaceDelete(ifIndex, ifName, objectId)
```

For the traps defined in this document:

ifIndex is a dynamically allocated interface number and is the key field for access to the RFC 1573 Interfaces Group MIB. This is a non-zero unique value allocated at object initialisation and not repeated until the PFA is rebooted.

ifName is a textual string from the RFC 1573 ifXTable and identifies the PFA port object (e.g. Port Number of the form 1-1-1-[1..18]). This value can be used to index in to a protocol-specific PFA MIB table.

objectId is the Object Identifier of the MIB table the port object is part of.

pfaPOPPAKType is a textual string describing the type of the POP PAK.

pfaLevel2iField contains the FRMR I field. This is 3 or 6 bytes depending on whether pfaL2Modulo indicate modulo8 or modulo128, respectively.

pfaHDLCEventReason is a string containing one of the following reasons why the pfaHDLCEvent trap was generated:

- SABM received in Data Transfer
- DISC received in Data Transfer
- DM received in Data Transfer
- UA received in Data Transfer
- N2*T1 expiry
- Unsolicited F bit
- N(S) outside window
- N(R) outside window
- TEST Unexpected
- XID Unexpected
- Illegal frame

pfaFRPVCStatus is the current status of a Frame Relay PVC, as defined in the Frame Relay DTE MIB (RFC 1315) and can take value from:

- ACTIVE = 2
- INACTIVE = 3

pfaX25PVCStatus is the current status of an X.25 or X.75 PVC or HVC. This contains one of the following:

- connected = 1
- invalidService = 4
- cleared = 5
- invalidAddress = 6

pfaL2Modulo is the Level 2 Modulo selected on a link level and is sent with pfaFRMR-Recd and pfaFRMRSent traps. The values are:

modulo8 = 1
modulo128 = 2

pfaNTNA is the Network Terminal Number (NTN) assigned to the A side of an X.25/X.75 PVC/HVC.

pfaNTNB is the Network Terminal Number (NTN) assigned to the B side of an X.25/X.75 PVC/HVC.

pfaLCNA is the Logical Channel Number (LCN) assigned to the A side of an X.25/X.75 PVC/HVC.

pfaLCNB is the Logical Channel Number (LCN) assigned to the B side of an X.25/X.75 PVC.

In addition the following objects are sent with traps, detailed in Sections 3.3 and 3.4, which are not related to a specific interface on the PFA.

pfaCaaReason is sent with the pfaAcctNonOperational trap and indicates the last event which caused the Call Accounting sub-system to become non-operational. The possible events are:

manuallyBlocked (1) - the user has just blocked Call Accounting by issuing a CDAAB command.

buffSizeExceeded (2) - the user-specified number of buffers, 1K each, for use for storing Call Accounting data has been used up.

naloLimitExceeded (3) - the user-specified threshold on the percentage free memory on the PFA has been exceeded causing Call Accounting to be suspended.

pfaCaaMemLimit specifies what percentage - fifty (1), ninety (2) or hundred (3) - of buffers allocated for Call Accounting has been used up.

pfaCaaMethod indicates which interface - mml (1) or ftp (2) - was used to delete Call Accounting records.

pfaFTPClientAddress contains the IP address of unauthorised host attempting to set up an FTP session to the PFA.

pfaFTPUserName contains an unauthorised user attempting to log-in to the PFA FTP Server.

Where applicable, each PFA interface will generate the generic linkUp or linkDown trap. This trap will be generated as defined in RFC 1573 but will include some additional variables.

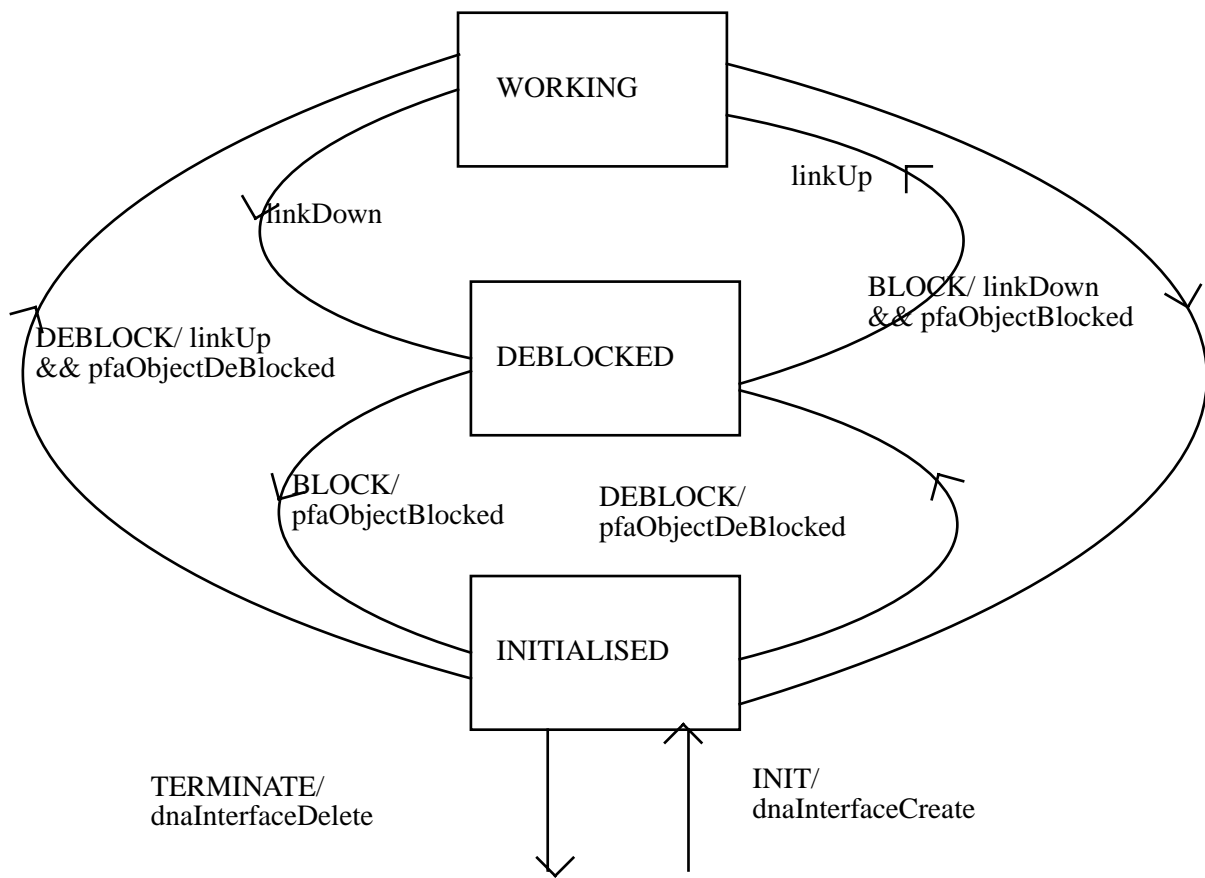


Figure 1 - PFA port Object State Transitions and Trap Generation

2.1. Synchronous PP port objects (FR + HDLC)

linkUp(ifIndex, ifName, ifDescr)par = LINKTRAP
linkDown(ifIndex, ifName, ifDescr)par = LINKTRAP

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
dnaInterfaceCreate (ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

pfaPopPakIn(ifIndex, ifName, pfaPOPPAKType)par = POPTRAP
pfaPopPakOut(ifIndex, ifName)par = POPTRAP

2.2. Synchronous LP port objects (LAPB + SDLC)

linkUp(ifIndex, ifName, ifDescr)par = LINKTRAP
linkDown(ifIndex, ifName, ifDescr)par = LINKTRAP

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

pfaDisturbanceSupervision(ifIndex,ifName)par = DISTTRAP
pfaLevel2FRMRRecd(ifIndex,ifName, pfaLevel2iField, pfaL2Modulo)
par = FRMRTRAP
pfaLevel2FRMRSent(ifIndex,ifName, pfaLevel2iField, pfaL2Modulo)
par = FRMRTRAP

pfaHDLCEvent(ifIndex,ifName,pfaHDLCEventReason)par = HDLCTRAP

2.3. NP port object (X.25, X.75, QLLC)

linkUp(ifIndex, ifName, ifDescr)par = LINKTRAP
linkDown(ifIndex, ifName, ifDescr)par = LINKTRAP

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

2.4. X.25/X.75 PVC/HVC

pfaX25PVCStatusChange(ifIndex, ifName, pfaNTNA, pfaNTNB, pfaLCNA,
pfaLCNB, pfaX25PVCStatus)par=PVCTRAP

2.5. Frame port objects

linkUp(ifIndex, ifName, ifDescr)par = LINKTRAP
linkDown(ifIndex, ifName, ifDescr)par = LINKTRAP

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

2.6. Frame Relay PVC objects

pfaFRPVCStatusChange(ifIndex, ifName, ppDLCI, pfaFRPVCStatus)
par=PVCTRAP

where ifName and ifIndex refer to the B-side of the PVC connection.

2.7. LA port object

linkUp(ifIndex, ifName, ifDescr)par=LINKTRAP
linkDown(ifIndex, ifName, ifDescr)par=LINKTRAP

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

pfaPopPakIn(ifIndex, ifName, pfaPOPPAKType)par = POPTRAP
pfaPopPakOut(ifIndex, ifName)par = POPTRAP

2.8. Ether NI port object

linkUp(ifIndex, ifName, ifDescr)par=LINKTRAP
linkDown(ifIndex, ifName, ifDescr)par=LINKTRAP

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

2.9. X.25 NI port object

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

2.10. FR NI port object

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

2.11. SLIP NI port object

linkUp(ifIndex, ifName, ifDescr)par=LINKTRAP
linkDown(ifIndex, ifName, ifDescr)par=LINKTRAP

pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
 pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
 dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
 dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

2.12. LCP object

linkUp(ifIndex, ifName, ifDescr) par = LINKTRAP
 linkDown(ifIndex, ifName, ifDescr) par = LINKTRAP
 pfaObjectBlocked(ifIndex, ifName, ifDescr) par = OBJTRAP
 pfaObjectDeBlocked(ifIndex, ifName, ifDescr) par = OBJTRAP
 dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
 dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP

2.13. MP object

linkUp(ifIndex, ifName, ifDescr) par = LINKTRAP
 linkDown(ifIndex, ifName, ifDescr) par = LINKTRAP
 pfaObjectBlocked(ifIndex, ifName, ifDescr) par = OBJTRAP
 pfaObjectDeBlocked(ifIndex, ifName, ifDescr) par = OBJTRAP
 dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
 dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
 pfaHDLCEvent(ifIndex,ifName,pfaHDLCEventReason) par = LINKTRAP

Note that the MP object generates a pfaHDLCEvent trap when it enters or leaves the backup state, using the event reason string "BACKUP MODE" or "NORMAL MODE" as appropriate.

This trap is controlled from the LINKTRAP parameter to reduce the number of parameters that need to be set in the user interface.

2.14. LLC-LP object

linkUp(ifIndex, ifName, ifDescr)par = LINKTRAP
 linkDown(ifIndex, ifName, ifDescr)par = LINKTRAP
 pfaObjectBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
 pfaObjectDeBlocked(ifIndex, ifName, ifDescr)par = OBJTRAP
 dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
 dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr) par = CONFTRAP
 pfaDisturbanceSupervision(ifIndex,ifName)par = DISTTRAP
 pfaLevel2FRMRRecd(ifIndex,ifName, pfaLevel2iField, pfaL2Modulo) par = FRMRTRAP
 pfaLevel2FRMRSent(ifIndex,ifName, pfaLevel2iField, pfaL2Modulo) par = FRMRTRAP
 pfaHDLCEvent(ifIndex,ifName,pfaHDLCEventReason)par = HDLCTRAP

2.15. LLC-NP object

linkUp(ifIndex, ifName, ifDescr)	par = LINKTRAP
linkDown(ifIndex, ifName, ifDescr)	par = LINKTRAP
pfaObjectBlocked(ifIndex, ifName, ifDescr)	par = OBJTRAP
pfaObjectDeBlocked(ifIndex, ifName, ifDescr)	par = OBJTRAP
dnaInterfaceCreate(ifIndex, ifName, objectId, ifDescr)	par = CONFTRAP
dnaInterfaceDelete(ifIndex, ifName, objectId, ifDescr)	par = CONFTRAP

3. Box-wide traps

3.1. SNMP

These traps are configured through the NANMS command.

authenticationFailure()	par="NAME=AUTHTRAP"
coldStart()	

3.2. DNA Network Topology

These box-wide Topology traps are configured through the NANMS command.

par=TOPOLOGYTRAP

dnaConnectionCreate(ifIndex, ifName, dnaInterfaceMyNeighbourAddress, dnaInterfaceMyNeighbourIfName)	
--	--

dnaConnectionDelete(ifIndex, ifName, dnaInterfaceMyNeighbourAddress, dnaInterfaceMyNeighbourIfName)	
--	--

3.3. Call Accounting

The following traps are configured through the CDAAx command:

pfaAcctOperational()	par=CAASTATUSTRAP
pfaAcctNonOperational(pfaCaaReason)	par=CAASTATUSTRAP
pfaAcctMemThreshold(pfaCaaMemLimit)	par=BUFFTRAP
pfaAcctRecordsDeleted(pfaCaaMethod)	par=DELETETRAP
pfaAcctRecordsLost()	par=RECSLOSTTRAP
pfaAcctCallsRejected()	par=CALLREJTRAP
pfaAcctCallsAccepted()	par=CALLREJTRAP

The following traps are configured through the NALOS command:

```
pfaLowFreeMemory()                par=LOWMEMTRAP
pfaFreeMemoryOK()                 par=LOWMEMTRAP
```

3.4. FTP Server

The following traps are configured through the CDFTS command:

```
pfaUnauthFTPConnection(pfaFTPClientAddress)  par=FTPTRAP
pfaUnauthFTPUser (pfaFTPUserName)           par=FTPTRAP
```

3.5. PFA 660 Hardware Traps

The following traps are relevant only to the PFA660 box.

3.5.1. Fan Fail Traps

The following traps are configured through the NAHWS/P commands.

```
pfaFanFail ()                      par=FANFAILTRAP
pfaFanOK ()                         par=FANFAILTRAP
```

3.5.2. Power Supply Unit Traps

The following traps are only relevant to a PFA 660 fitted with the optional Dual-Redundant Power Supply unit. They are configured using the NAHWS/P commands:

```
pfaPSUFail ()                      par=PSUFAILTRAP
pfaPSUOK ()                         par=PSUFAILTRAP
```

4. References

- [1] EBC/C/0173PFA Traps MIB
- [2] DNA MIB Document No. 96 03-CAA 112 1113