

Voice Response Virtualization Layer
(VRVL)
Implementation Manual

Neda Document Number: 103-102-04

Last Updated: 2000/03/23 18:31:25

Doc. Revision: 1.1.1.1

Neda Communications, Inc.

April 28, 1999

List of Tables

List of Figures

Contents

1 Introduction	3
1.1 Significant Events	3
1.2 Development Environment	3
2 PP Layer Interface	3
2.1 Initialization	3
2.2 SAP Management	4
2.3 Event Handling	4
2.4 Event Handling	4
2.5 Action primitives	4
3 PP Internal Functions	4
3.1 Internal functions	4
4 Implementation – D4X, SCO	5
4.1 System Revisions	5
4.2 System Configuration	5
4.3 Internal Software Design	5
4.4 Known Problems	6
4.5 Dialogic Contact Logs	6
5 PP Implementation using D4x on UNIX	6
5.1 Initialization	6
5.2 SAP management	6
5.3 Event Handling	8
5.4 Action Primitives	9
6 PP Implementation using D4x on Windows NT	11
6.1 General Information	11
7 Internals	11
7.1 Dialogic d4x interface	11
7.2 Internal functions	11

Preface

This document reflects the PP layer as it exist today. It is our intention to add updates as more functionality is created.

Software implementation of PP layer adhere to the "Open C Environment" (OCE). Realizing efficient real open system.

Implementors, system architects, system programmers, application programmers and any one who is interested in developing a Interactive Voice Response System that is hardware independent can benefit from reading this book.

Example of a upper layer over PP is recordex and V-COMPILER. "VoRDE" integrates the built-in applications, the "V-COMPILER", and the voice response system management capabilities into a single Voice Response Program.

Example of a lower layer is D40-LIB. A "C" library that makes use of the driver-provided Services conveniently availabe to "C"application programs.

1 Introduction

This document describes the PP Layer. PP provides a hardware independent interface to the programer for the developement of Interactive Voice Response programs.

1.1 Significant Events

The following is a list of significant events that result into execution of code inside of PP layer.

- Initialization.
- SAP Management.
- Action Primitives.
- Event Primitives.

1.2 Development Environment

A C compiler and linker are expected of a development environment .

2 PP Layer Interface

Following are the PP Layer Interface based on Unix Dialogic Implementation. PP Layer Interface functions are listed below:

2.1 Initialization

- PP_init Initialize the PP_module

2.2 SAP Management

- PP_sapBind Bind an Address to a Service User
- PP_sapUnBind
- PP_sapEventGet
- PP_attach Associate Phone-Port to SAP
- PP_detach DeassociatePhone-Port from SAP

2.3 Event Handling

- PP_portEventGet Get a event associated with the port
- PP_onHook Go on hook
- PP_wink Send a wink on the specified port
- PP_waitRing Wait for a ring

2.4 Event Handling

- PP_eventTriggerMaskSet
- PP_eventTriggerWait Block until specefied trigger
- PP_portEventGet
- PP_waitRing
- PP_getDtmfs
- PP_getMfs

2.5 Action primitives

- PP_onHook Go on hook
- PP_offHook Go off hook
- PP_playN Play a file
- PP_plaympN
- PP_recordN Record a file
- PP_dial Dial a number
- PP_setDtmfDurations
- PP_genDtmf Generate DTMF or MF tones

3 PP Internal Functions

3.1 Internal functions

- genEvent
- genInd
- genCnf
- carTermToDialEnd Call Analysis Result
- showCap
- getBoardDevName

- getDevName convert port Id to device name
- isGoodDtmf Check for valid DTMF
- dtmfFlush flush old DTMF's

Consistent with the naming conventions mentioned in the Open C Platform, all exposed interfaces of this module are prefixed by PP_.

4 Implementation – D4X, SCO

4.1 System Revisions

Diallogic board D4x/D. Firmware downloaded through Diallogic SpringWare Support.

Description of the system.

```
System = eblis
Node = eblis
Release = 3.2v4.2 SCO UNIX
KernelID = 93/04/28
Machine = i80486
BusType = ISA
Serial = 2AF005439
```

Version of drivers.

- 1 Diallogic Generic Voice Development Platform - Version 4.1
- 2 Diallogic SpringWare Support - Version 4.1
- 3 Diallogic Demonstration Programs - Version 4.1

4.2 System Configuration

D4x/D board, IRQ 5

Diallogic SpringWare Support setup:

Name	Nch	Prot	Subdev	Type	Addr	memlen	Port	TS	FWL	File	FrontEnd	ClockSrc
d4xdb1	0	1	S	D/4XD	D0000	2000	0	1	d4x.fwl	analog	default	

Diallogic Generic Voice Development Platform setup:

Name	Nchn	Prot	Subdev	Addr	Smemlen	Port	Type
dxxxB1	4	1	C	D0000	2000	0	D4X

4.3 Internal Software Design

Note all the important things you had to do to get it to work. /usr/diallogic/dx_demos/d4xtools needs to be link.

4.4 Known Problems

Loop current is not detected. Compiled alarm with the latest PP_ interface. Not test todate. AMX related code is not ported.

4.5 Dialogic Contact Logs

Who did we buy the proms from?
Who are the good guys at Dialogic.

5 PP Implementation using D4x on UNIX

Following are the PP Layer Interface based on Unix Dialogic Implementation. PP Layer Interface is listed below:

5.1 Initialization

PP layer needs to be initialized before providing any services. During the initialization the module obtains its required resources.

```
void PP_init()  
Initialize the PP_ module. Initialize the Trace Module for PP_.
```

5.2 SAP management

A hierarchical SAP addressing scheme is used

```
PP_SapDesc      PP_sapBind( sapDesc )  
PP_SapDesc sapDesc;  
Bind an Address to a Service User.  
  
SuccFail      PP_sapUnBind( sapDesc )  
PP_SapDesc sapDesc;  
Unbind an Address from a Service User.  
  
SuccFail PP_sapEventGet( sapDesc, event )  
PP_SapDesc      sapDesc;  
PP_Event      *event;  
Get Any events associated with the Port.  
  
typedef struct PP_PortInfo {  
    struct PP_PortInfo *next;  
    struct PP_PortInfo *prev;  
    PP_PortId portId;  
    Int dev;
```

```

    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

```

```

typedef struct PP_Event {
struct PP_Event *next;
    struct PP_Event *prev;
    PP_PortDesc portDesc;
    PP_EventId evtId;
    union EventData {
        PP_PlayCnf playCnf;
        PP_GetDtmfCnf getDtmfCnf;
        PP_RecordCnf recordCnf;
        PP_DialCnf dialCnf;
    } evtData;
} PP_Event;

```

```

PP_PortDesc PP_attach( sapDesc, portId )
PP_SapDesc      sapDesc;
PP_PortId      portId;

```

Request that the named Phone-Port be associated with this SAP. Get a PP_portDesc. Open at the

```

typedef struct PP_SapInfo {
    struct PP_SapInfo *next;
    struct PP_SapInfo *prev;
    struct PP_PortInfoSeq portInfoSeq;
    PP_EventSeq eventSeq;
} PP_SapInfo;
typedef PP_SapInfo *PP_SapDesc;

```

```

typedef struct PP_EventSeq {
    struct PP_Event *first;
    struct PP_Event *last;
} PP_EventSeq;

```

```

typedef Int PP_PortId; /* Name By which the Phone Port is identified */

```

```

SuccFail PP_detach( portDesc )
PP_PortDesc      portDesc;

```

Request that the named Phone-Port be deassociated with this SAP

```

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
} PP_PortInfo;

```

```

    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

```

5.3 Event Handling

```

SuccFail PP_eventTriggerMaskSet (PP_PortDesc portDesc, PP_EventTriggerMask mask)
Not implemented

```

```

SuccFail PP_eventTriggerWait (PP_PortDesc portDesc)
Not implemented

```

```

SuccFail PP_portEventGet ( portDesc, event ) /* Get Any events associated with the Port */
PP_PortDesc                portDesc;
PP_Event                    *event;

```

```

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;

```

```

typedef struct PP_Event {
    struct PP_Event *next;
    struct PP_Event *prev;
    PP_PortDesc portDesc;
    PP_EventId evtId;
    union EventData {
        PP_PlayCnf playCnf;
        PP_GetDtmfCnf getDtmfCnf;
        PP_RecordCnf recordCnf;
        PP_DialCnf dialCnf;
    } evtData;
} PP_Event;

```

```

SuccFail PP_waitRing ( portDesc, nuOfRings )
PP_PortDesc portDesc;
int nuOfRings;

```

```

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

```

```

SuccFail PP_getDtmfs(portDesc, nuDtmfs, termDtmf, wait)
PP_PortDesc portDesc;
int      nuDtmfs;
char     termDtmf;
int      wait;

```

```

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

```

5.4 Action Primitives

```

SuccFail PP_onHook( portDesc )
PP_PortDesc portDesc;

```

```

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;

```

```

SuccFail PP_offHook( portInfo )
PP_PortInfo *portInfo;

```

```

typedef struct PP_PportInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;

PP_playN( portDesc, vMsgDesc, nuDtmfs, typeahead ) /* Play a message */
PP_PortDesc      portDesc;
VM_MsgDesc       vMsgDesc;
int              nuDtmfs;
BOOLEAN         typeahead;

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

SuccFail PP_plaympN( portDesc, msgp, nuDtmfs, typeahead ) /* Play a message or series */
PP_PortDesc      portDesc;                /* of messages. */
char             *msgp;                    /* pointer to message or filename */
int              nuDtmfs;
BOOLEAN         typeahead;

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

SuccFail PP_recordN( ppNu, vMsgDesc, maxDuration, maxSilence, nuDtmfs )
int      ppNu;
VM_MsgDesc vMsgDesc;
int      maxDuration;    /* Seconds */
int      maxSilence;    /* Seconds */

```

```

int      nuDtmfs;

SuccFail PP_dial( portDesc, phoneNu )
PP_PortDesc    portDesc;
String         phoneNu;

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

SuccFail      PP_setDtmfDurations( boardDev )
int boardDev;
Set new DTMF duration and inter-digit delay.

SuccFail PP_genDtmf( portDesc, phoneNu )
PP_PortDesc    portDesc;
char          *phoneNu;

```

6 PP Implementation using D4x on Windows NT

6.1 General Information

Computer Platform: Windows NT 4.0 (Service Pack 3)
 Dialogic Board Type: D/41ESC
 Dialogic Software: Dialogic System Software SDK for Windows NT DNA Version 3.1

7 Internals

7.1 Dialogic d4x interface

TBD.

7.2 Internal functions

Following are the PP Layer internal functions.

```

void genEvent( ppEvtId, portDesc, csb, evtInfo ) /* Translate Dialogic Events */
PP_EventId   ppEvtId; /* into PP_Events - Expected Event Id */
PP_PortDesc  portDesc;
DL_CSB       *csb;
char         *evtInfo; /* this is a pointer CAR */

typedef enum PP_EventId {
    PP_EvtDisConInd, /* No Data */
    PP_EvtErrorInd,
    PP_EvtOnHookCnf, /* No Data */
    PP_EvtOffHookCnf, /* No Data */
    PP_EvtRingInd, /* No Data */
    PP_EvtPlayCnf,
    PP_EvtGetDtmfCnf,
    PP_EvtRecordCnf,
    PP_EvtDialCnf,
    PP_EvtGenDtmfCnf, /* No Data */
    /* -- Triggering Events -- */
    PP_EvtTrigRingInd,
    PP_EvtTrigSilenceInd,
    PP_EvtTrigNoSilenceInd,
    PP_EvtTrigLoopCurrentInd,
    PP_EvtTrigNoLoopCurrentInd
} PP_EventId;

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

typedef struct dl_csb { /* Channel Status block */
    unsigned char cs_lstat; /* Current raw line status */
    unsigned char cs_state; /* Current state */
    unsigned char cs_dtmfcnt; /* Size of DTMF queue */
    unsigned char cs_evntcnt; /* Size of event queue */
    unsigned char cs_hookstate; /* Hook state */
    unsigned char cs_callstate; /* Last dial state */
    unsigned char cs_rfu[2]; /* RFU for packing-independence */
    unsigned long cs_termtype; /* Last command termination reas on */
    unsigned long cs_trcount; /* Bytecount play or recorded */

```

```

} DL_CSB;

typedef struct dl_car {
    unsigned char cr_termtype; /* Termination reason */
    unsigned char cr_frqout; /* % of freq. out of bounds */
    unsigned short cr_frqhz; /* Freq. detect in Hz. */
    unsigned short cr_sizehi; /* Duration of non-silence (10mS units) */
    unsigned short cr_shortlow; /* Duration of shorter silence */
    unsigned short cr_longlow; /* Duration of longer silence */
    unsigned short cr_ansrsize; /* Duration of answer (10mS units) */
} DL_CAR;

SuccFail genInd( portDesc, csb, evtInfo )
PP_PortDesc portDesc;
DL_CSB *csb;
char *evtInfo;

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;
typedef PP_PortInfo *PP_PortDesc;

SuccFail genCnf( ppEvtId, portDesc, csb, evtInfo )
PP_EventId ppEvtId; /* Expected Event Id */
PP_PortDesc portDesc;
DL_CSB *csb;
char *evtInfo;

typedef struct PP_PortInfo {
    struct PP_PortInfo *next;
    struct PP_PortInfo *prev;
    PP_PortId portId;
    Int dev;
    struct PP_SapInfo *sapInfo;
    Char dtmfBuf[PP_K_DtmfNuMax];
} PP_PortInfo;

typedef struct dl_csb {
    unsigned char cs_lstat; /* Current raw line status */

```

```

unsigned char cs_state; /* Current state */
unsigned char cs_dtmfcnt; /* Size of DTMF queue */
unsigned char cs_evntcnt; /* Size of event queue */
unsigned char cs_hookstate; /* Hook state */
unsigned char cs_callstate; /* Last dial state */
unsigned char cs_rfu[2]; /* RFU for packing-independence */
unsigned long cs_termtype; /* Last command termination reason */
unsigned long cs_trcount; /* Bytecount played or recorded */
} DL_CSB;

/* PP_DialEnd */
carTermToDialEnd( car )
DL_CAR *car; /* Call Analysis Result */

typedef struct dl_car {
unsigned char cr_termtype; /* Termination reason */
unsigned char cr_frqout; /* % of freq. out of bounds */
unsigned short cr_frqhz; /* Freq. detect in Hz. */
unsigned short cr_sizehi; /* Duration of non-silence (10mS units) */
unsigned short cr_shortlow; /* Duration of shorter silence */
unsigned short cr_longlow; /* Duration of longer silence */
unsigned short cr_ansrsize; /* Duration of answer (10mS units) */
} DL_CAR;

void showCap( myCap )
DL_CAP *myCap;

typedef struct dl_cap {
unsigned short ca_nbrdna; /* # of rings before no answer. */
unsigned short ca_stdely; /* Delay after dialing before analysis. */
unsigned short ca_cnosis; /* Duration of no signal time out delay. */
unsigned short ca_lcdly; /* Delay after dial before lc drop connect */
unsigned short ca_lcdly1; /* Delay after lc drop con. before msg. */
unsigned short ca_hedge; /* Edge of answer to send connect message. */
unsigned short ca_cnosil; /* Initial continuous noise timeout delay. */
unsigned short ca_lo1tola; /* % acceptable pos. dev of short low sig. */
unsigned short ca_lo1tolb; /* % acceptable neg. dev of short low sig. */
unsigned short ca_lo2tola; /* % acceptable pos. dev of long low sig. */
unsigned short ca_lo2tolb; /* % acceptable neg. dev of long low sig. */
unsigned short ca_hiltola; /* % acceptable pos. dev of high signal. */
unsigned short ca_hiltolb; /* % acceptable neg. dev of high signal. */
unsigned short ca_lo1bmax; /* Maximum interval for short low for busy. */
unsigned short ca_lo2bmax; /* Maximum interval for long low for busy. */
unsigned short ca_hilbmax; /* Maximum interval for 1st high for busy */
unsigned short ca_nsbusy; /* Num. of highs after nbrdna busy check. */

```

```

unsigned short ca_logltch; /* Silence deglitch duration. */
unsigned short ca_higlth; /* Non-silence deglitch duration. */
unsigned short ca_lo1rmax; /* Max. short low dur. of double ring. */
unsigned short ca_lo2rmin; /* Min. long low dur. of double ring. */
unsigned short ca_intflg; /* Operator intercept mode. */
unsigned short ca_intfltr; /* Minimum signal to qualify freq. detect. */
unsigned short ca_frqmin; /* Lower freq. detect limit */
unsigned short ca_frqmax; /* Upper freq. detect limit */
unsigned short ca_devmax; /* Upper freq. limit deviation */
unsigned short ca_smpsize; /* No.of samples used in freq. detection */
unsigned short ca_hisiz; /* Used to determine which lowmax to use. */
unsigned short ca_alowmax; /* Max. low before con. if high >hisize. */
unsigned short ca_blowmax; /* Max. low before con. if high <hisize. */
unsigned short ca_nbrbeg; /* Number of rings before analysis begins. */
unsigned short ca_hilceil; /* Maximum 2nd high dur. for a retrain. */
unsigned short ca_lolceil; /* Maximum 1st low dur. for a retrain. */
unsigned short ca_lowerfrq; /* Lower allowable frequency in hz. */
unsigned short ca_upperfrq; /* Upper allowable frequency in hz. */
unsigned short ca_timefrq; /* Total duration of good signal required. */
unsigned short ca_rejctfrq; /* Allowable % of bad signal. */
unsigned short ca_maxansr; /* Maximum duration of answer. */
unsigned short ca_ansrdgl; /* Silence deglitching value for answer. */
}

```

```

char * /* A pointer to a device name. NULL pointer if range error. */
getBoardDevName( portId, channelNum ) /* Figure device name for a board */
PP_PortId portId; /* given a port ID. */
int *channelNum;

```

```

typedef Int PP_PortId; /* Name By which the Phone Port is identified

```

```

char * /* A pointer to a device name. NULL pointer if range error. */
getDevName( portId ) /* Figure device name given a port ID. */
PP_PortId portId; /* portId - a number from 1 - 32. */

```

```

BOOLEAN isGoodDtmf(c)
char c;

```

```

SuccFail dtmfFlush( portInfo )
PP_PortInfo *portInfo;

```