



PBXpress

Ultimate VoIP Business Solution

**PBXpress XML-RPC
Integration Guide**

Copyright Notice & Disclaimers

Copyright (c) 2004-2010 PortaOne, Inc. All rights reserved.

**PBXpress XML-RPC Integration Guide
V.2.1.3 May 2010**

Please address your comments and suggestions to: Sales Department,
PortaOne, Inc. Suite 408, 2963 Glen Drive Coquitlam, BC V3B 2P7
Canada

Changes may be made periodically to the information in this publication. Such changes will be incorporated in new editions of the guide. The software described in this document is furnished under a license agreement, and may be used or copied only in accordance with the terms thereof. It is against the law to copy the software on any other medium, except as specifically provided in the license agreement. The licensee may make one copy of the software for backup purposes. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopied, recorded or otherwise, without the written permission of PortaOne, Inc.

The software license and limited warranty for the accompanying product, are set forth in the information package supplied with it, and are incorporated herein by this reference. If you cannot locate the software license, please contact your PBXpress representative for a copy.

All product names mentioned in this manual are for identification purposes only, and are either trademarks or registered trademarks of their respective owners.

Table of Contents.....
PBXpress XML-RPC Integration Guide.....	1
Preface.....	4
Introduction.....	5
XML-RPC.....	6
Accessing the XML-RPC Interface.....	6
Data Structures.....	7
XML-RPC Calls.....	9
Appendices.....	11
Sample Program for XML-RPC Communication.....	11

Preface

This document provides information on how to integrate external applications (such as CRM) with the PBXpress VoIP PBX.

Where to Get the Latest Version of This Guide

The online copy of this guide is always up-to-date, integrating the latest changes to the product. You can access the latest copy of this guide at: www.pbxpress.com/support.

Conventions

This publication uses the following conventions:

- Commands and keywords are given in **boldface**
- Terminal sessions, console screens, or system file names are displayed in fixed width font



Caution indicates that the described action might result in program malfunction or data loss.

NOTE: Notes contain helpful suggestions about or references to materials not contained in this manual.



Timesaver means that you can save time by performing the action described in the paragraph.



Tips provide information that might help you solve a problem.

Introduction

PBXpress is an IP PBX enabling users to send and receive calls from their IP phones and providing advanced features such as call hold, call transfer, voicemail and auto-attendant.

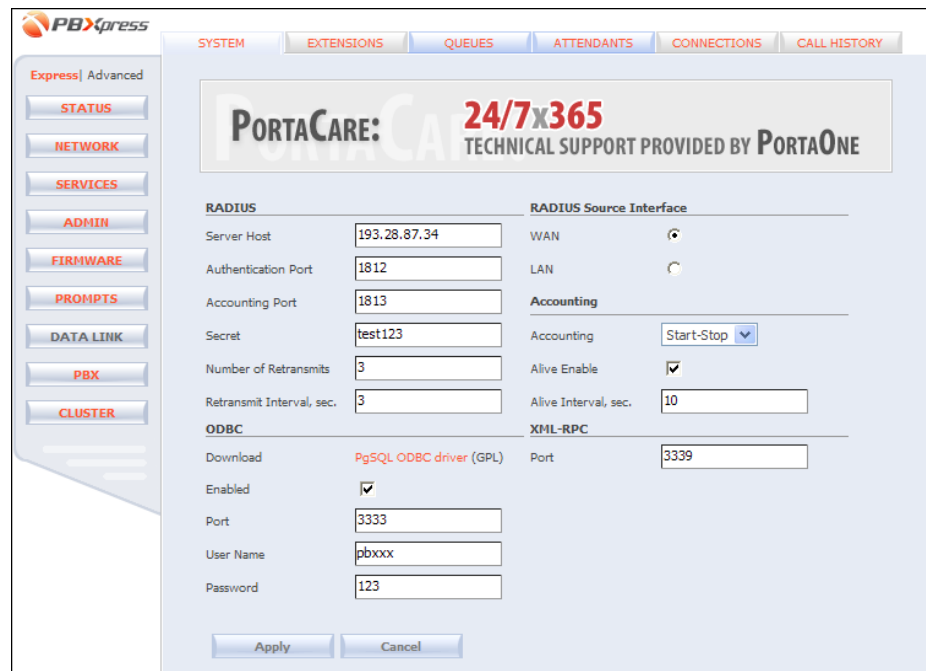
You may be interested in integrating an external application with PBXpress. For instance, when an incoming call rings on your phone, your CRM will display a pop-up with the name of the person calling you and other important information. In order to do this, the application must read certain information from PBXpress, such as your extension status or incoming call info. This is done via XML-RPC.

XML-RPC is a Remote Procedure Calling protocol that works over the Internet. An XML-RPC message is an HTTP-POST request, and the body of the request from your application is in XML. A procedure executes on the PBXpress server and returns the requested information formatted in XML. This enables you to develop your applications in any programming language and on any platform. For more information about XML-RPC, visit <http://www.xmlrpc.com/>.

XML-RPC

Accessing the XML-RPC Interface

Either a WAN IP or LAN IP may be used to access the XML-RPC interface. The XML-RPC port is configured on the **System/Data Link** page of the admin interface.



The screenshot shows the PBXpress admin interface. The top navigation bar includes tabs for SYSTEM, EXTENSIONS, QUEUES, ATTENDANTS, CONNECTIONS, and CALL HISTORY. The left sidebar has a menu with options: STATUS, NETWORK, SERVICES, ADMIN, FIRMWARE, PROMPTS, DATA LINK, PBX, and CLUSTER. The main content area is titled 'PORTACARE: 24/7x365 TECHNICAL SUPPORT PROVIDED BY PORTAONE'. Below this, there are configuration sections for RADIUS, RADIUS Source Interface, Accounting, ODBC, and XML-RPC. The XML-RPC section includes a 'Port' field set to 3339. At the bottom of the form are 'Apply' and 'Cancel' buttons.

Sample URL:

<http://192.168.0.123:3339/RPC2>

The access credentials for the XML-RPC interface are the same as those used for self-care interface access. Some limitations apply to the management of extensions, favorites and calls:

- Users can access favorites from their own admin group.
- Users can get information about extensions from their own admin group.
- Users can manage calls from their own admin group only.

Data Structures

Extension Information

The extension record is an associative array, i.e. a set of attributes represented by key/value pairs. The following attributes are found:

Key	Description
id	Internal id (integer)
type	Describes the type of extension; can have one of three values: <ul style="list-style-type: none"> • extension • queue • attendant
abbreviated	Extension number
callerid	Structure with two fields: <ul style="list-style-type: none"> • phone – complete phone number • name – display name (caller info) for this extension
voicemail	0 or 1; indicates whether or not voicemail is enabled
email	External email address associated with this extension
state	One of two possible values (online or offline) indicating the following: <ul style="list-style-type: none"> • for extensions – whether the IP phone with this extension is currently connected to PBXpress; • for queues – whether at least one of the phones registered in the queue is currently online and available; • for attendants – whether the attendant is currently active, according to its activity period definition.

Each entity (e.g. extension or queue) is uniquely identified by a combination of two parameters: **id** and **type**.

Call Information

The call record is an associative array, i.e. set of attributes represented by key/value pairs. The following attributes are found:

Key	Description
id	Call identifier (string)

state	Current call state; can have one of these values: setup, connected, onhold, ringing, parked, unknown
from	Structure with the following fields: <ul style="list-style-type: none"> • id – ID of extension record • number – phone number • name – display name (caller info) associated with this phone number • type – one of the following: extension, queue, attendant, vm, unknown
to	Structure with the following fields: <ul style="list-style-type: none"> • id – ID of extension record • number – phone number • name – display name (caller info) associated with this phone number • type – one of the following: extension, queue, attendant, vm, unknown
duration	Elapsed call duration in seconds
via	Name of connection used to transport this call (empty for calls between two extensions within PBXpress)
qcall	0 denotes a normal call, 1 indicates a call to a queue; a call to a queue is reported via several call records: one (incoming) call to the queue and a multiple outgoing call, i.e. one for each ringing extension

Favorites Information

A “favorites” entry is an associative array, i.e. a set of attributes represented by key/value pairs. The following attributes are found:

Key	Description
id	Internal ID (integer)
slot	Slot number
size	Whether this favorite occupies a standard space (single) or larger area (double) on the operator console screen
type	Describes the type of entity; can have one of three values: <ul style="list-style-type: none"> • extension • queue • attendant

XML-RPC Calls

Operator Plug-in Methods

get_calls()

Returns a list of current calls on PBXpress.

Input: none

Output: array of call records

get_extensions()

Returns a list of currently configured extensions (including queues and attendants).

Input: none

Output: array of extension records

get_favorites()

Returns a list of currently defined favorites for the operator console.

Input: none

Output: array of favorites

set_favorites()

Updates the list of favorites for the operator console.

Input: array of favorites

Output: none

transfer()

Performs a call transfer to a new destination for a specific call.

Input: callid, destination number (in vm-abbreviated form for vm box destination numbers)

Output: none

drop()

Disconnects a specific call.

Input: callid

Output: none

Call Parking Methods

get_parking_info()

Returns a list of currently parked calls.

Input: none

Output: <parking_extension, slots, parked_calls>

(parking_extension: the number of the first extension number allocated for parked calls;

slots: maximum number of parked calls;
parked_calls: currently parked calls) The information is passed as an associated array, with parked slot numbers as keys and callids as values.

park()

Parks a specific call.

Input: callid, slot (optional)

Output: slot taken

Storage Plug-in Methods

The purpose of the following methods is to allow users' personal settings to be stored on PBXpress.

dict_set()

Stores a single configuration value.

Input: key, value (both are strings)

Output: none

dict_set_many()

Stores a set of configuration values.

Input: associated array of <key, value> pairs to be saved

Output: none

dict_get()

Retrieves a configuration value.

Input: key

Output: value

dict_set_serialized()

Input: key, obj (key is a string, obj is either an array or associated array)

Output: none

dict_get_serialized()

Input: key

Output: obj

dict_delete()

Deletes a configuration value.

Input: key

Output: none

Appendices

Sample Program for XML-RPC Communication

```
#!/usr/local/bin/python
# vim:et:ts=4 sw=4:
# $Id: sample.py,v 1.1 2007/06/06 14:35:34 gonzo Exp $

import getopt
import httplib
import logging
import os
import socket
import string
import sys
import time
import xmlrpclib

from base64 import encodestring
from logging import FileHandler

class BasicAuthTransport(xmlrpclib.Transport):
    def __init__(self, username=None, password=None):
        self.username = username
        self.password = password
        self.verbose = 0

    def request(self, host, handler, request_body, verbose=0):
        # issue XML-RPC request

        h = httplib.HTTP(host)
        h.putrequest("POST", handler)

        # required by HTTP/1.1
        h.putheader("Host", host)

        # required by XML-RPC
        h.putheader("User-Agent", self.user_agent)
        h.putheader("Content-Type", "text/xml")
        h.putheader("Content-Length", str(len(request_body)))

        # basic auth
        if self.username is not None and self.password is not None:
            h.putheader("Authorization", "Basic %s" % string.replace(
                encodestring("%s:%s" % (self.username,
self.password)),
                "\012", ""))
            h.endheaders()

        if request_body:
            h.send(request_body)
```

```
errcode, errmsg, headers = h.getreply()

if errcode != 200:
    raise xmlrpclib.ProtocolError(
        host + handler,
        errcode, errmsg,
        headers
    )

return self.parse_response(h.getfile())

def handle_server_exception(action, em):
    """Handle server-side exception. action is performed action and em
    is a Fault instance"""
    reason = em.faultString.decode('string-escape')
    if em.faultCode > 0:
        print 'Internal Server Error: %s' % reason
    else:
        print '%s' % reason
    print "%s failed: internal server error: %s\n" % (action, reason)

if __name__ == "__main__":
    server = xmlrpclib.Server("http://testpbx.portaone.com:8080/RPC2", \
        BasicAuthTransport('203', '203'))

    try:
        print "Extensions:"
        print server.get_extensions()
        print "Calls:"
        print server.get_calls()
        print "Favorites:"
        print server.get_favorites()
        print "keys/values:"
        print server.get_all()
    except socket.error, (err, errstr):
        print errstr
        sys.exit(1)
    except Exception, em:
        print em
        handle_server_exception("xml-rpc sample", em)
        sys.exit(1)
```