PSG Client SDK- for VFP 9.0

PSG
Programming Solutions Group

PSG 2.0 Client Programming Manual

---

by Programming Solutions Group

PSG - a client/server application development platform for database applications.

PSG Client SDK- for VFP 9.0

Printed: August 2012

Publisher

...enter name...

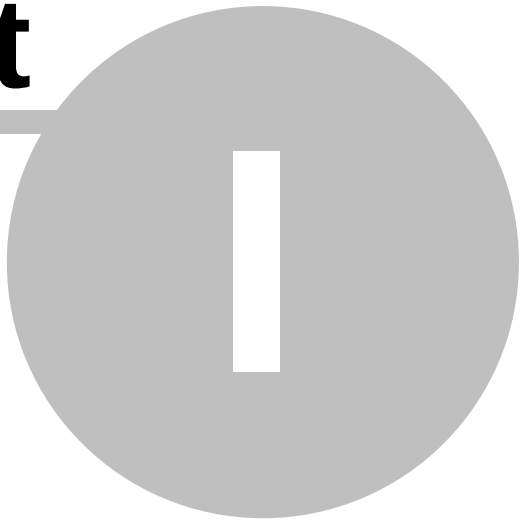
Technical Editors

...enter name...

Table of Contents

Index 0

Part



1 Introduction

PSG 2.0 a client/server database applications development platform.
It offers a secured solution for building applications that can efficiently function in heterogenous networks like Internet, Intranet and local networks. The variety of built-in components and classes for different programming languages makes integration faster when working with a PSG platform and also allows developing modular applications.

PSG platform allows database applications to efficiently perform over Internet or Intranet. It enables easy loading of thousands of records instantly, using them locally and updating secured server. It builds rich Internet applications operating as good and efficient as any desktop applications using different programming languages. Among them .NET, Visual FoxPro or Java using PSG as an intermediate communication layer that's secured by default.

PSG 2.0 can help you with:

- multi user applications based on relational databases
- ERP's , Accounting, Warehouse, Production management, Human resource applications and any other similar areas
- CRM's, SFA's
- Document management applications
- Database applications for custom purposes
- secured applications
- applications that require an additional level of security
- Software as a service (SAAS) solutions

PSG 2.0 can be used for clients applications developed with:

- Microsoft programming languages like .Net C#, VFP
- JAVA (crossplatform client - Windows, Linux)
- support for other languages like Delphi, Ruby and maybe others, currently on research (January 2012 - check psgsdk.com).

PSG 2.0 enables faster development of application compared with other solutions.
The system provides a series of facilities by default, simplifying the development of complex applications, reducing developing time with 40 to 60 percent:

- user management and user access rights per user and users groups
- integrated powerful reporting engine
- templates and classes
- integrated simple help system (as HTML pages)

- integrated international toolkit

For client/server applications:

- compared to .NET web services can decrease 3-4 times the time required to develop application modules (from days to hours)
- compared to WEB AJAX technologies development time could be 5 times shorter.

For using PSG 2.0 you need only:

- programming experience with one of next languages: C#, VFP or JAVA
- same programming language could be used client and server side

Some experience with client/server N-tier applications could be a plus, but not mandatory.

PSG 2.0 uses as additional resources:

- PSG server, which can be installed on any Microsft Operating system starting with Windows XP. The client is compatible with Microsoft XP,2000-2003, Vista, Win 7.
- database server, no other software or facilities are required.
- licenses for software development, which are not provided with PSG (.NET studio or VFP).

To run your application over Internet:

- just set your NAT router/firewall to allow access to the PSG server secured port.

In terms of security over the Internet:

- PSG server runs behind firewalls in a totally protected environment. No need for DMZ's.

- PSG server use's SSL 128 bit encryption and PKI up to 2048 bytes. Communication protocol is HTTPS.

- While VPN's are not required can be used if necessary to comply to the company internal rules.
- The users are authorized with:
  - username and password
  - username/password and hardware key
- The server protects itself against brut force password crackers.
- The SSL security mechanism is implemented by default
  - tools to create custom certificates are provided.

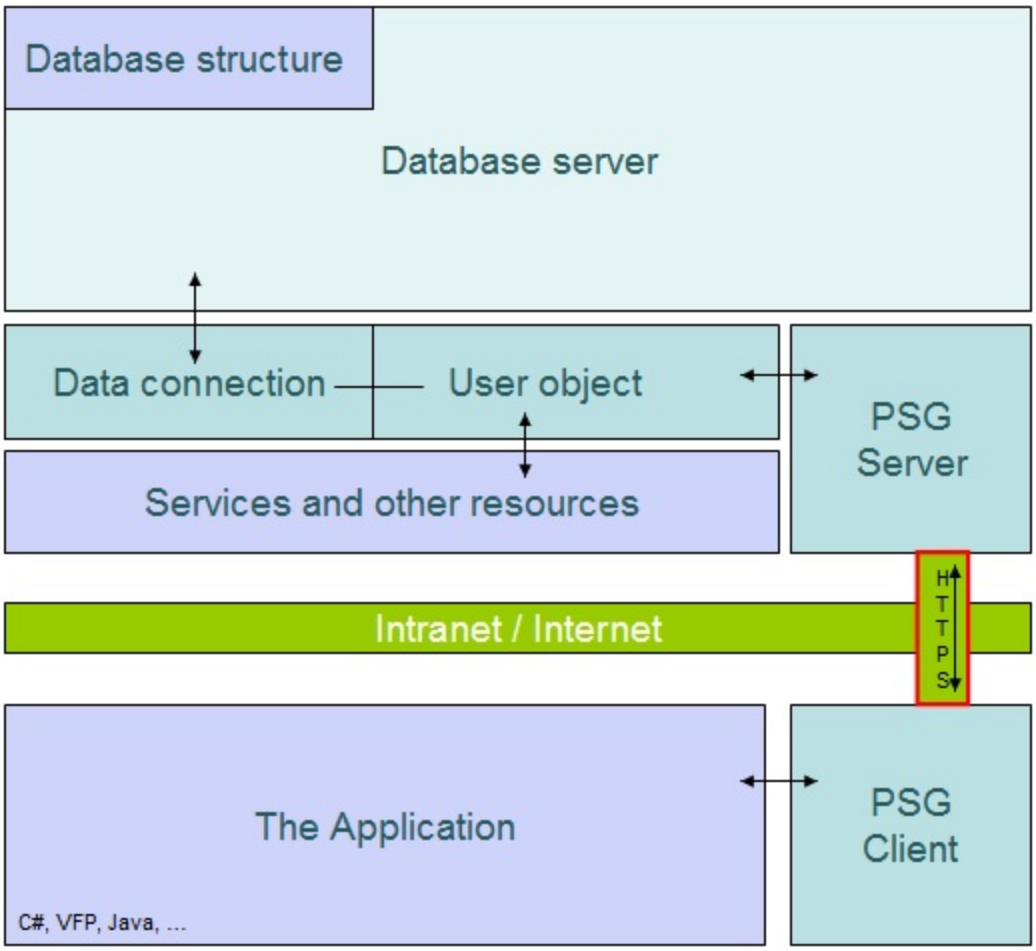
Maintenance costs

- For the client the maintenance cost is close to zero, same as a Web Browser maintenance.
- Client application will be automatically updated with the last releases from the server.
- Database server should be administrated by a specialist according to size and complexity (like any other application database servers)
  - for small databases (few hundreds Mb) might be that there is be nothing to be done
  - medium and big databases need maintenance depending on the database server specifications (ie. some databases like Oracle require a database administrator as Oracle is generally used for complex databases).

1.1 Technical view

PSG is a platform used to develop and run client server database applications based on REST services model.

The platform is built on a 4-tier architecture model.



- User Interface (UI) and local business logic
- Communication layer (PSG Client/server)
- Server part business logic unit
- Database server

PSG is based on a HTTPS WEB server and client for data exchange. The communication layer optimizes the data transfer between the client application and the database server, enabling PSG to serve also as WEB static and dynamic pages.

While PSG serves also WEB static and dynamic pages the first option is to optimize the data transfer between the client application and the database server as a communication layer.

The platform offers an easy way to implement new services in minutes given the REST used technology.

Built as a platform for software development offers from the start:

- user management and access rights
- system main menu interface
- help system (an online HTML help can run on a separate server port), and context help can be offered as well.
- reports and data analysis engine
- base class objects for easy implementation of the client side applications

- international toolkit

The PSG server internally supports secure communication (SSL). Basic principle of such communication requires that both client and server have their private and public keys. During the process those two export their public keys to each other, and any data sent from one side to another is encrypted using those keys. ONLY other side is able to decrypt the data (with private key), and therefore transmissions like this are secure. In case that a third part is logging information sent from one side to another would be unable to decrypt it alone in a reasonable amount of time (could take few years to do that).

By using MD5 files checksum PSG implements a very strict control of modules that can run client side. Client software modules are first downloaded from the server, being then stored into the local cache. They can be used only if the MD5 checksum matches the server records, which means that only approved versions can run. Each user group or individual user can have their own access rights on modules making this security measure also useful for software updates. When adding the new module on server updates for all clients are done automatically.

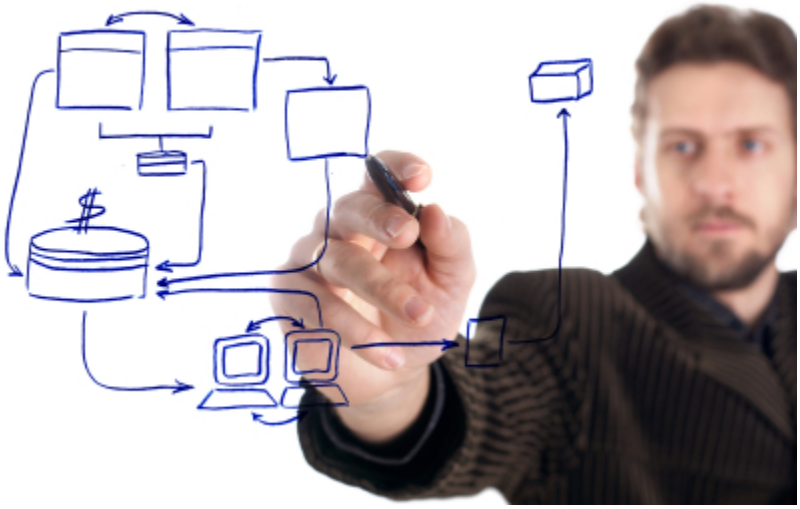
PSG can serve up to thousands users, depending on server hardware. More PSG servers can be configured to offer access to the same database cluster.

PSG platform is open to extend the capabilities on server and client side. The server accepted services can be easily customized and enhanced to fit all needs. To start programming an Intranet/Internet application in days, only basic programming knowledge is needed. Database applications are developed fast in the preferred programming language.

Client side programming. By using common development tools as .NET C#, Visual FoxPro or JAVA IDE's, applications can be built faster. Classes and templates that handle all communication processes are provided.

PSG solutions are highly scalable. An enterprise solution could provide services to hundred thousands users in different system configuration.

1.2 About



WEB www.psgsdk.com
vfp.psgsdk.com
csharp.psgsdk.com
Support support@psgsdk.com

History (major steps)

2012 - January -Client side hardware key support

2011 - September - client side support for running several applications in the same time.
- C# script programming server side

2011 - January - final release 2.0

2010 - July - Release 2.0 A
First release of the new server application.
Client interfaces for .NET C# and Visual FoxPro.

2009 - September - the communication engine changes from TCP/IP proprietary socket secured protocol to standard HTTPS communication protocol.

2005 - first functional release of the platform.

Development of PSG was started in **2004**.

Part II

2 Programming using VFP

The PSG server and client depends only by a common implemented service syntax.

Client side applications can be created using almost any well known programming language, the PSG team offers support only for C#, VFP and JAVA (please check the www.psgsdk.com for an updated list).

Client development is open and a complete description of communication protocol structure is available upon request to help implementing a different programming language base classes and tools by third part developers.

PSG client developent SDK's provided by psgsdk.com are free, but other providers for different PSG SDK's may charge for a fee.

PSGSDK.COM provides free tools for the following languages (please check www.psgsdk.com for more information):
- .NET C#
- VFP
- JAVA (NetBeans)

Source code may be available on request, please send the request to support@psgsdk.com .

Visual FoxPro was the first programming language used client side by PSG from 2004.

Building network database applications becomes faster and easier when using PSG platform and VFP.
All PSG applications can be operated over Internet.

Visual FoxPro IDE helps to develop PSG client side applications.
SDK for VFP provides:

- base classes to be used in projects
- templates (prg,forms,containers)
- samples applications based on Ms. Northwind database.

2.1 Overview

PSG platform is a complex structure, but the development of an application follows a very simple procedure.
Developing n-tier client/server application involves writing code on both sides client and server.

Visual FoxPro 9.0 service pack 2 can help to create PSG applications.

SDK for VFP provides:

- base classes to be used in projects
- templates (prg,forms,containers)
- samples applications based on Ms. Northwind database.

All these are used client side mainly.

Server side code is implemented as text scripts.
PSG services are implemented automatically using declared scripts, no registration is required.

The PSG services editor is available on server configuration utility in "Utils" page.
Services can be saved as XML files to be used as backup or to export to other server.

The PSG client side application model usually looks as follows:

- Main application interface
- Reports application - embedded (if installed)
- PSGCON object - communication object
- HELP - html pages
- International toolkit
- Application menu
    - Documents
        Forms and Containers (methods provided)
            - controls (Text, Edit, Combo, Spinner, Check)
            - communication controls (GETDATA)

PSGCON is used for communications. Based on this all controls are implemented as easy as possible.

2.2 Main application

The end user main application interface.
It is initiated by the login startup module. When started it receives parameters like session Id and others.
Application menu is built locally depending on user rights.

The PSGCON object is instantiated into the main application. PSGCON is part of the communication layer of PSG platform and it is coordinating all communications with the server such as services, uploads and downloads.

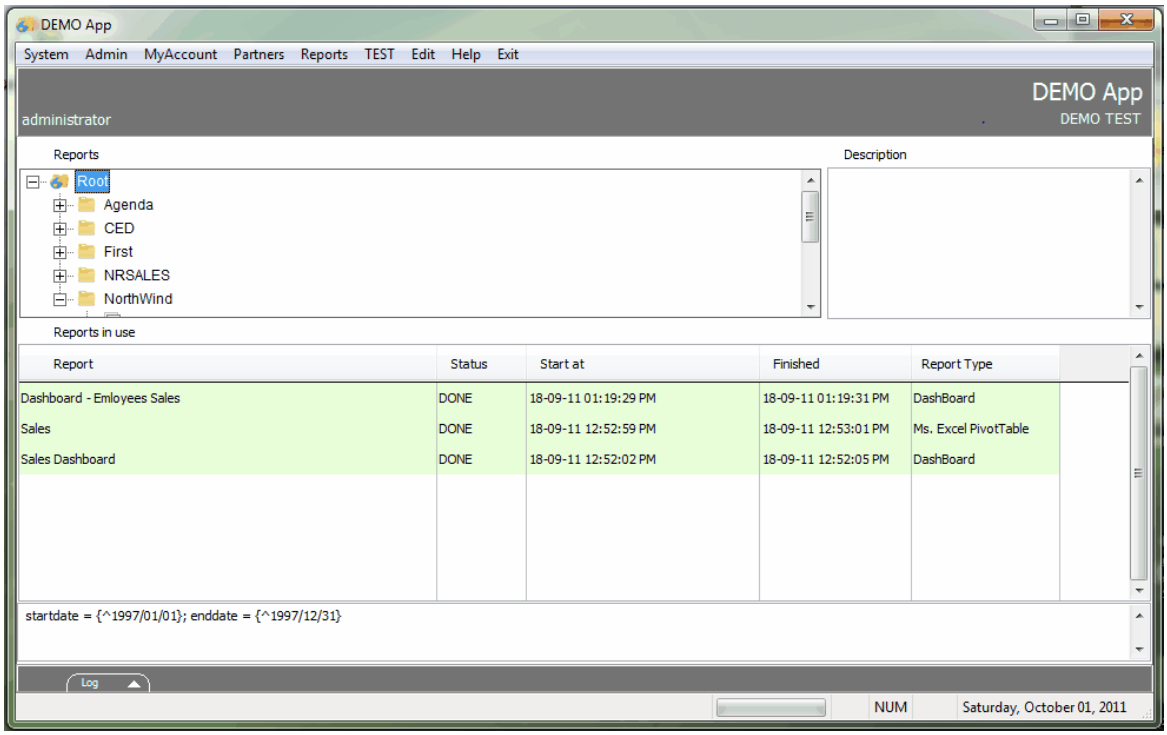
The main application is provided by PSGSDK.COM .

A custom main application to serve a specific purpose could be created using the default one as a sample.

The PSGCON object uses third party components for HTTPS communication and encryption, royalty free licensed with the main application. Source code is provided for free, but additional components should be licensed from third party vendors to be used for development.
In order to develop PSG applications there is no need for the additional licenses until you need to redesign the main user interface.

Main application properties that can be used in the modules:

_screen.server_ip
_screen.server_port
_screen.logincode
_screen.loginname
_screen.loginpass
_screen.company
_screen.appname



2.3 PSGCON

PSGCON deals with all the communication forms between the client and the PSG server. With its server corespondent creates the communication layer of PSG platform.

- access PSG services
- download files
- upload files

The PSG client server communication is asynchronous. The server can not send any data to the client until requested.

One PSG service responds with one answer and optional parameters. The answer can be a command to be executed locally by the client application.
Some of the commands implemented into the PSGCON deal with usual functionality of the client.

In order to extend the client side functionality or to accept new custom commands client side, a new object is used into each module code (see GETDATA object).
Combining server services with local commands the communication becomes slightly complex, however first communication is initiated by the client.

PSGCON Object:

Methods:

send_command	used to send commands to the PSG server (services)
sendfile	send a file (see the tutorial for details)
getfile	get a file (see the tutorial for details)
run	used to properly check and run an application module, used by the main application menu
getinfo	used to check a module licence code (when modules are licensed separately)
getnewobject	used to create a new object unique name to be used later
rnum2char	properly create a string from a number
loadhelp	loads the help engine with the help page set - could provide help context

Sample:

using from code
psgcon.<method>(<parameter list>)

like

psgcon.send_command('ECHO#"HELLO"',thisform)

server service in the sample is ECHO and the parameter of ECHO is HELLO.
"thisform" object as a second parameters is provided, the response will be interpreted by the object sent as parameter and the embedded GETDATA methods .

Properties:

loginname	login user name
logincode	session key
help_page	html help page, can be set from each application module in order to work as context help the help system is created using HTML pages
remotehost	server IP or name
remoteport	server port
company	registered to
done_response	PSG service response

usrcode	user ID - PSG users list
password	login password
devaddress	developer address
devappid	application ID
devappname	application name
devcompany	developer
devemail	application support email
devwebaddress	application support site

Properties can be used in code when needed.

```
m.usrcode  = psgcon.usrcode
```

2.4 Help system

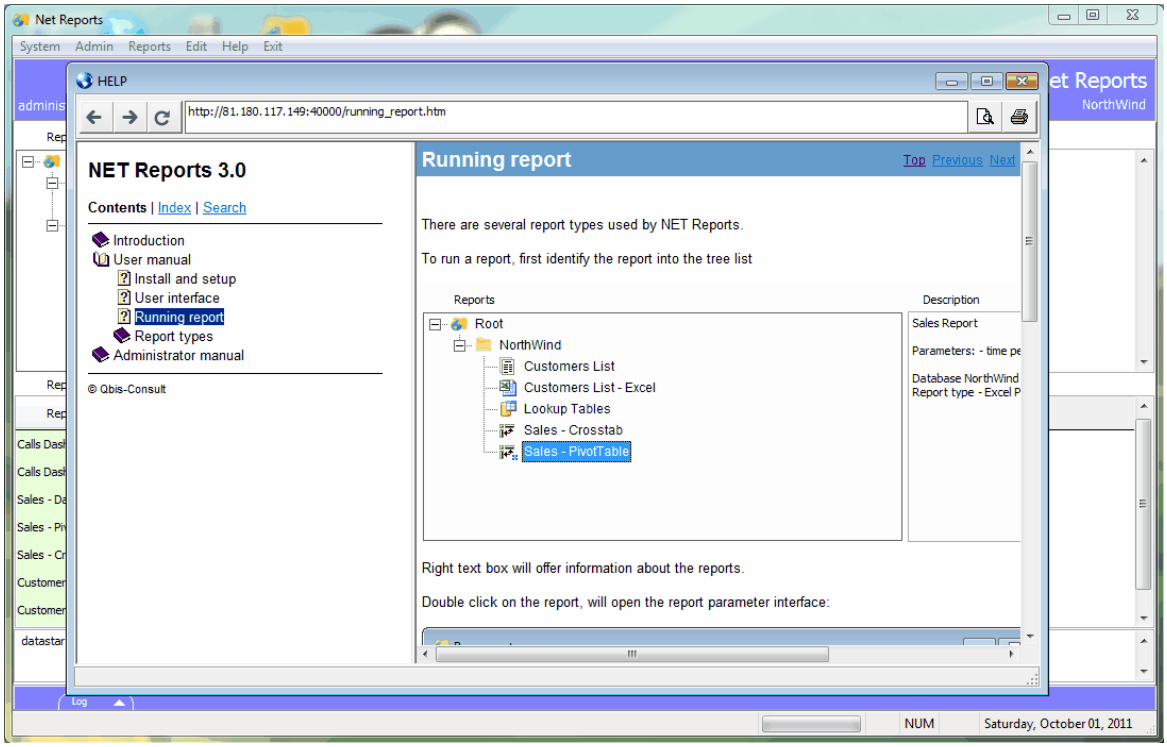
PSG platform also offers the application help system.

Server side, a WEB HTTP server listens on the server port minus ten. (Ex: server port 25010; ie help port 25000)

Help is provided as a WEB site that can be created using any available tools. We recommend Help & Manual from www.helpandmanual.com

The help start page should be "index.htm". Help is available by pressing F1 key.
In order to provide context help, the psgcon.help_page property should be set to the context page whenever is needed. Do not forget to set it back to "index.htm" when the module/object is released.

The Help module is based on IE ActiveX.

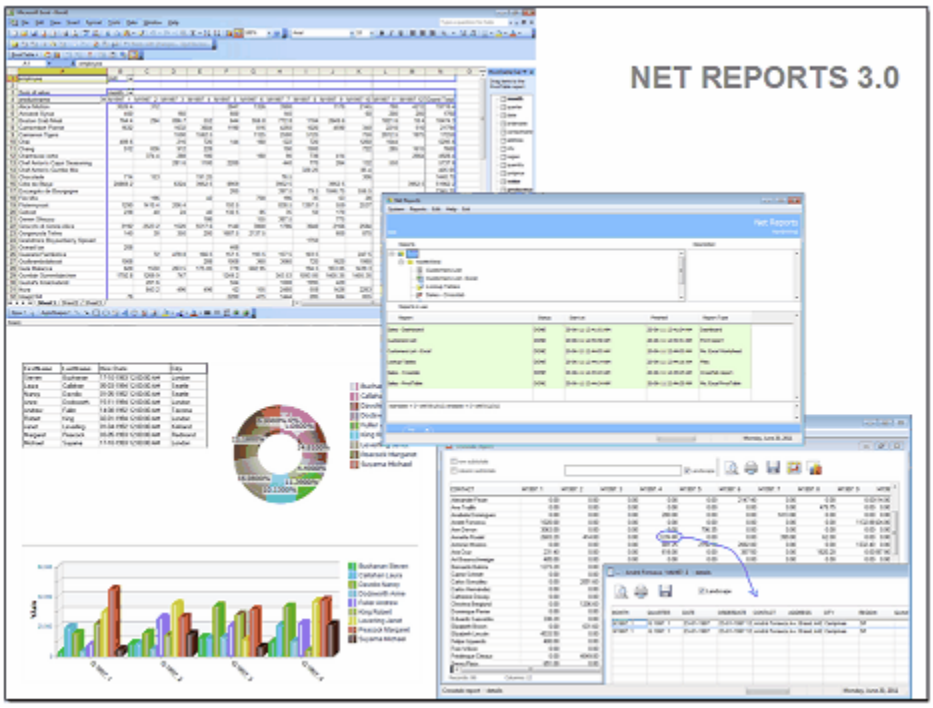


2.5 Embeded reports engine

PSG client could embed the light release of NET Reports 3.0, limited to work with one database only (the application main database).
NET Reports it is not mandatory, it works like an add on for the PSG Platform.
The NET Reports information is not part of this documentation. Please check the NET Reports help file for more information.

The reports access rights can be set per individual user or user groups.
Available reports are:

- Export files
- Print reports (classic reports)
- CrossTab reports - drill down
- Ms. Excel PivotTable report
- Ms. Excel list
- Dashboard (uses graphs, drill down cross tabs, tables and text created from different data cursors)



2.6 Application main menu & modules

Application main menu & modules are simple and easy to use.

The application main menu is declared server side using the server config utility/ "Installed documents" tab..

Document = single file that launches an application part (a VFP project compiled into an exe file).

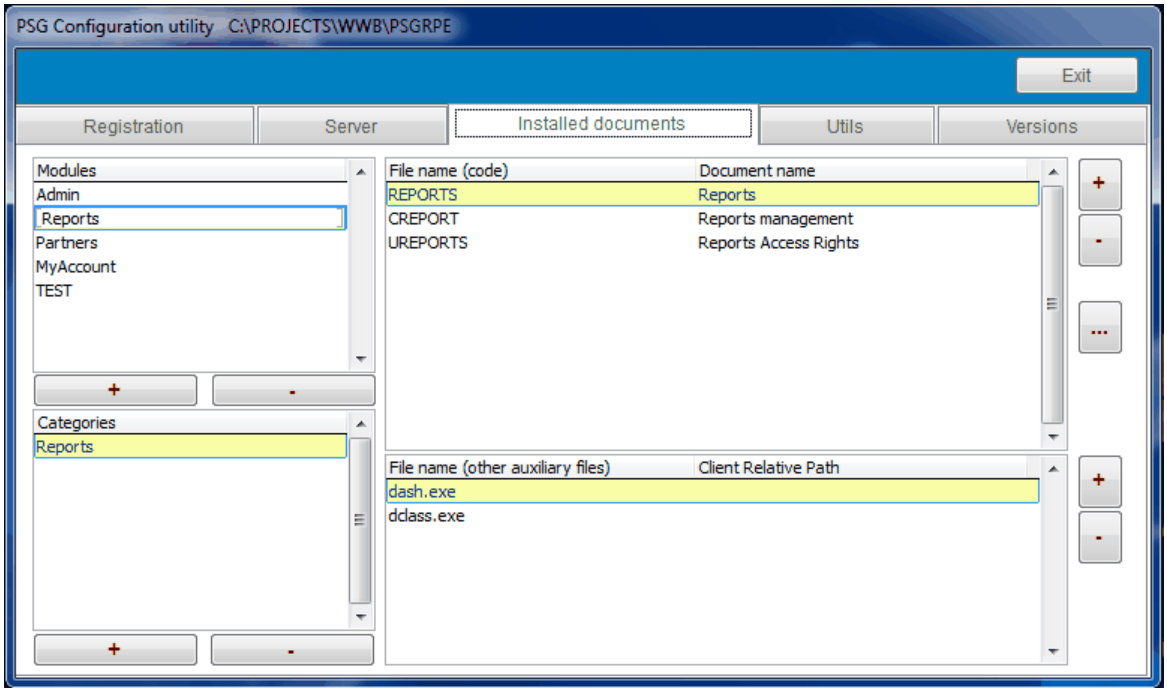
The documents are stored under Categories and Modules.
Modules will become pads into the main menu, and documents become bars of the menu.

When a bar is selected the "document" will be checked for MD5 conformity and executed.
If one pad has only one document it will activate the document in the same way as a bar when selected.

For each "document" we can have dependencies or auxiliary files that are needed to run the "document". These files will be also checked for MD5 checksum before use.

If one document or auxiliary file is not found client side or the MD5 checksum is not the correct one, the file will be downloaded from the server.
This serves the security roles but also as the client application update mechanism.

Please check also USERS MANAGEMENT to see how to allocate users rights to use documents.



2.7 Application documents

Application documents are projects build as executable files that can run on top of the main application interface and use the PSG functionality.

One document should deal with data communication methods.

The proposed "document" model:

- one form that load the data
- when the dataset is loaded the form loads a container with controls to show and edit the dataset.
- each change into the dataset is automatically sent to the server using the PSGCON.
- the communication mechanism is embedded into the controls that fires the WHEN and VALIDATE methods

This is a recommended basic scenario, but not limiting the possible "documents" to be created. One could do anything in a project in order to achieve the application goals. But almost anytime will deal with data load, show and modify. PSG services should be used given the fact that a lot of jobs could be implemented server side as well.

See the application main menu for "documents" integration into the application.

Each "document" is a VFP project, that spares the application in small pieces easier to maintain and update.
To be able to use a "document" it is mandatory that it is published on the server after it is created.

Please check the PSG SDK for VFP for samples and templates.

2.7.1 Loading form - dataset

The loading form is a special form that prepares the command for data request and finally receives and loads the dataset.

Data request is done in the "INIT" method of the form.

Sample from a project.

```
PARAMETERS CE_,report_code
THIS.Name = m.CE_

this.interface_class = 'creport'
this.interface_container = 'usrcontrol2'

this.getdata.addtable('report')
this.getdata.addparameter(m.report_code)
this.getdata.addtable('rrep_details')
this.getdata.addparameter(m.report_code)
this.getdata.addtable('reports_databases')

this.getData.gettables()
```

In the INIT method, the GETDATA object prepares the PSG DATA service request in an easy and convenient way.
Here is set the user interface to be used by the form after data loading. (interface class and container)

See GETDATA for details.

The DATA command is sent to the server and a zip file with the dataset tables is prepared server side. See DATA command /service.
For VFP a database container with the tables is prepared in order to preserve the eventually long names for fields.
If JAVA or .NET are used, the server will prepare XML files. DBF method is faster as no xml parser is used.

When download done event occurs the GETDATA object fires the form LOAD_DATA method.
The data tables are opened and the user interface is loaded.
Please see the templates and samples for details. Source code is available.

2.7.2 Base classes VCX

Classes will be used when developing client side application modules.

container	instantiated by the LOAD_DATA method of a loading form. (not each loading form should instantiate a container, this depends on the application logic)
texbox	
editbox	
checkbox	
combobox	
spinner	
optionbox	
label	

All controls embeds code that deals with PSG data communication using the PSGCON object.
Just fill in the boxes and configure some parameters.
Check also the samples form the SDK files.

Grids could be used by replacing columns base controls with the modified classes.
To insert a new class into a grid column (designer mode);

- edit the grid to the desired column (open the properties window and position to the column).
- drop the control to the grid column
- position to the previous control in the properties window and close the properties window
- press delete - this will remove the previous used control.
- open the properties window on column and set the SPARSE property to .T.
- set the PSG control properties as described

2.7.2.1 Base container

Instantiated by the LOAD_DATA method of a loading form.

A basic container class has only INIT method code, generally used to position the window into the screen.

The container should be populated with all necessary controls. The controls are set into the container at design time instead on form, as the form is loaded before we have any data to be used by controls. The form LOAD_DATA method actually opens the used tables.

If one control is not properly configured one error message will be shown into the application and the form LOAD_DATA method stops.

Sample INIT method of the container :

```
thisform.Height = this.Height + 2
thisform.Width  = this.Width + 2
thisform.Refresh
```

```
this.Parent.Top  = INT((_screen.Height - this.Parent.Height)/2)
this.Parent.Left = INT((_screen.Width  - this.Parent.Width)/2)
this.Anchor = 15

thisform.Caption = "Form NAME"
this.Visible = .t.
```

2.7.2.2 TextBox

Based on the TextBox class

psgtext.vcx

Used for character,numeric, integer, date and date-time fields; can edit up to 200 character sized text fields, for larger fields use the EDITBOX instead.

New properties:

initial_value	set by the control WHEN method
key_field	to be set at design time
servertablename	to be set at design time

Modified methods:

when	- set the intial_value property, to be used by the VALID method
valid	- fires only if the control value was modified - prepares and send the update command to the server service "UPDATE" - the command is sent using PSGCON - if the service fails the control value is reset to the previous value and an error message is shown - see the actual class code into the SDK for details

2.7.2.3 EditBox

Based on the EditBox class

psgedit.vcx

New properties:

initial_value	set by the control WHEN method

key_field	to be set at design time
servertablename	to be set at design time

Modified methods:

when	- set the intial_value property, to be used by the VALID method
valid	- fires only if the control value was modified - prepares and sends the update command to the server service "UPDATE" - the command is sent using PSGCON - if the service fails the control value is reset to the previous value and an error message is shown - see the actual class code into the SDK for details

2.7.2.4 CheckBox

Based on the CheckBox class

psgcheck.vcx

New properties:

initial_value	set by the control WHEN method
key_field	to be set at design time
servertablename	to be set at design time

Modified methods:

when	- set the intial_value property, to be used by the VALID method
valid	- fires only if the control value was modified - prepares and send the update command to the server service "UPDATE" - the command is sent using PSGCON - if the service fails the control value is reset to the previous value and an error message is shown - see the actual class code into the SDK for details

2.7.2.5 ComboBox

Based on the ComboBox class

psgcombo.vcx

New properties:

initial_value	set by the control WHEN method
key_field	to be set at design time
servertablename	to be set at design time

Modified methods:

when	- set the intial_value property, to be used by the VALID method
valid	- fires only if the control value was modified - prepares and send the update command to the server service "UPDATE" - the command is sent using PSGCON - if the service fails the control value is reset to the previous value and an error message is shown - see the actual class code into the SDK for details

2.7.2.6 Spinner

Based on the Spinner class

psgspinner.vcx

New properties:

initial_value	set by the control WHEN method
key_field	to be set at design time
servertablename	to be set at design time

Modified methods:

when	- set the intial_value property, to be used by the VALID method
valid	- fires only if the control value was modified - prepares and send the update command to the server service "UPDATE" - the command is sent using PSGCON - if the service fails the control value is reset to the previous value and an error message is shown - see the actual class code into the SDK for details

2.7.2.7 OptionBox

Based on the OptionBox class

psgoption.vcx

New properties:

initial_value	set by the control WHEN method
key_field	to be set at design time
servertablename	to be set at design time

Modified methods:

when	- set the intial_value property, to be used by the VALID method
valid	- fires only if the control value was modified - prepares and send the update command to the server service "UPDATE" - the command is sent using PSGCON - if the service fails the control value is reset to the previous value and an error message is shown - see the actual class code into the SDK for details

2.7.2.8 Label

Used as a replacement for base class for international applications.

New property to be set at design time:
- lid - numeric - the label ID to be used as caption

Depending on selected language will load the actual caption for the label.

Check the label class code and use the same code for grid headers and other captions in the application.

2.7.2.9 PSGInsert

Stored in PSGDATA.VCX

The "insert" class is used to send an INSERT command to the server.
Drop the class to the container or form.

Used method:

INSERT

```
PARAMETERS table_,remote_table,prim_key
```

table_ = local table cursor
remote_table = server table name
prim_key = primary key name

Sample:

(insert form a click method of a button)

<primary key> = sys(2015) + psgcon.usrcode
<field1> = <field1 value>
...

this.parent.psginsert1.insert(<local table name>,<remote table name>,<primary key>)

All fields that need to be inserted must have a declared variable with the same name. The method checks for all fields in the local table and search for corresponding declared variable to be use in the insert.
Keep in mind that some database servers like MS.SQL or PostgreSQL do not accept empty date and datetime fields.

2.7.2.10 PSGDelete

Stored in PSGDATA.VCX

The "delete" class is used to send a DELETE command to the server.
Drop the class to the container or form.

Used method:

DELETE

```
PARAMETERS table_name,key_name,record_id,local_table

table_name  = server side table name
key_name    = primary key
record_id   = primary key value (to be deleted)
local_table = local table/cursor
```

2.7.2.11 GetData

Used by the loading form to prepare and send the DATA command.
DATA sends the instructions to the server to prepare the needed local tables.

Another purpose is to extend the client capabilities giving the power to respond to custom commands that come from the server.

It is used into the loading form template.

If the template is not used and the form is built from scratch:
- drop the class to the form or container
- change its name to "getdata" (by default will have "getdata1"

Methods:

addtable	.getdata.addtable(<table name>) server side a service named as <table name> should be prepared. This special kind of service is used to prepare the table cursor on server side. Check the PSG server side programming for details.
addparameter	.getdata.addparameter(<parameter name>) adds a parameter to a table. It is used to load a filtered record set from the server
gettables	.getdata.gettables() it prepares and sends a DATA command to the server which prepares the local tables to be sent (server side).
accept	PSG platforms relays on communication between the client and server. The client always initiates the communication as it works asynchronous. The server could respond with a command to be executed locally. In order to trigger the local command as soon as it arrives from the server a calling mechanism is in place that use the ACCEPT method from GETDATA class. The PSGCON ACCEPT method checks if it has an object to send the response to: If the object is found it sends the command string to the ACCEPT method of GETDATA from the object. If no object is found, the command is solved locally by PSGCON. There is a limited list of commands for PSGCON in the special instructions set. Communication between the PSG client and server could be extended to cover all the requirements of the application. The PSGCON SEND_COMMAND method has two parameters: - the actual command to be sent - the initiator object (not mandatory) psgcon.send_command(<command>,[object]) If the "object" is sent to the PSGCON, the accept method will fire on the object set as default. See PSG server side programming.

2.8 Server side programming

The PSG server is open to extend it's services. Services are stored as scripts in a script database.

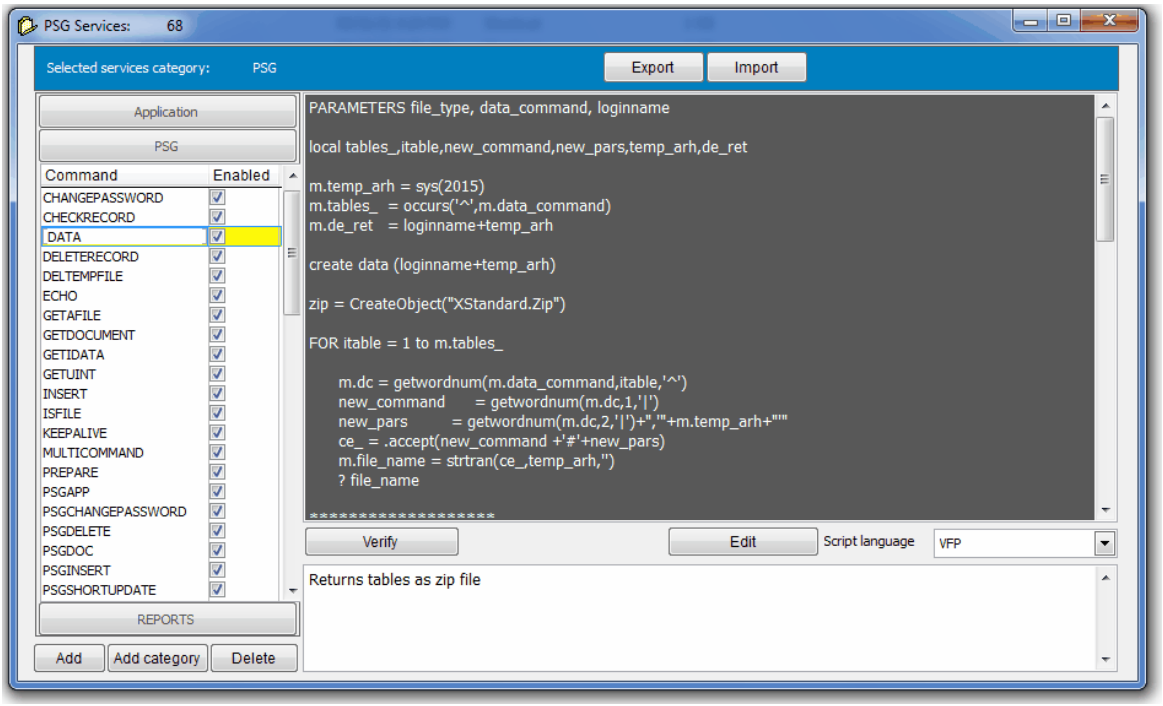
The PSG services could access some server objects and properties. Please check the PSG SERVER SDK for a complete documentation.

Here we will refer only to the basic server side programming.

2.8.1 PSG Services

The services scripts could use VFP, C#, VB script, Javascript as programming language. The PSG base implemented services are written using VFP.
One service should use one programming language, but on the same server there can be services written with different programming languages.
They all offer the same functionality.

Services could be managed from an editor application that starts from Utils tab of PSG server config application or as standalone PSGCOMMAND.EXE in the server directory.
Services scripts are stored into a SQLITE database named psgcommand.db



2.8.1.1 PSG Services programming

The services editor stores the services split by categories.

By default there are 3 categories:
- PSG - stores system services
- Reports - stores NET Reports services
- Application - stores the application custom services

New categories could be added, taking into account that a service name is unique on server. One PSG service can use another service. Samples to be found in DATA service.

Generally, the services should respond as soon as possible (in few seconds). In case a lengthy process should be used it is recommended to launch it out of service process and check later for a response. REPORTS_RUN launches an external program, and the client will check ISREPORTDATA in order to see if the job is completed. Sample are available in Reports.

The service command request has the following syntax:

<servicename>#<parameter1>,<parameter2>,...,<loginname>

Each service should have at least one parameter, the user login name that is added by the PSG server automatically.
Each service should return a string as result.

The result string could be also a command to the client application (see the client GETDATA accept method).

Ex:

RETURN "DONE#OK"

Sample ECHO service:

PARAMETERS test_value ,loginname

RETURN 'ECHO#'+test_value

The service command sent by the client is:

ECHO#"Echo check"

2.8.1.2 Base services

Most common services, are generally used by the PSG templates classes.

CHANGEPASSWORD	used by system configuration to change the password for the logged in user
	PARAMETERS new_password,loginname
CHECKRECORD	checks if a record exists
	PARAMETERS table_name, primary_key,primary_key_val, loginname
DATA	this is one of the most common ones, and it is described separately
	PARAMETERS file_type, data_command, loginname
DELETERECORD	deletes one record
	PARAMETERS table_name,key_name, key_value,loginname

DELTEMPFILE	deletes a file in the WORK_ directory on server PARAMETERS file_, ccommand_, loginname returns a command to be executed client side after the temporary file is deleted server side
ECHO	used to test the communication
INSERT	inserts a record PARAMETERS insert_command, loginname parses the fields values from the "insert_command" and sends the INSERT command to the database server some servers do not accept empty fields for date and datetime fields returns OK/NOK
ISFILE	checks if a file exists on server (work_ directory)
KEEPALIVE	used by the client to keep the session open backward compatibility, new servers can recover a session after expiring (no commands for more than one minute)
MULTICOMMAND	used to fire more than one service at the same time
UPDATEFIELD	updates a field in the database
UPDATETEXT	updates a text field in the database (more than 240 characters)

Generally one service can't fail if the initial conditions are the same from development (database structures, connections and others).

The development time errors that could be encountered when working with a service are:
ERROR 1000 - invalid command format (the command do not respect the standard syntax - ie. the '#' mark or the name is incompatible (only alphanumeric and no spaces).
ERROR 1001 - command not found, the service is not set server side
ERROR 1002 - error in command, there are errors in the service that should be solved.

Please check services script code for the service editor.
Other services in PSG category are related to the system management.

2.8.1.3 Data services

Data services command is prepared by GETDATA class and used by the loading form.

It is a special kind of services, one that uses other services to accomplish its task.

In order to develop PSG applications the base services cover almost all task required by the usual requests.
However each application uses a different database and different datasets at one time.

The request of datasets or temporary cursors to be sent client side are solved by services.
There are services that are called only by the DATA service.
To call a service the ACCEPT method of the connection object is used:
.accept(<service name>#<parameters>)

Each table request prepared by GETDATA has a corresponding service here. The DATA service will not be modified.

There is no general service while each request could have a different number of parameters. However one could implement that if needed.

DATA receive the request for tables/XML files, prepare the files into one ZIP archive and send a DATA command back to the client:

```
RETURN 'DATA#'+de_ret
```

The client main interface downloads the file, sends a command to delete the temporary ZIP archive on server and fires the "DOWNLOADDONE" command in GETDATA ACCEPT method. The loading form opens the tables to be used into the application. Table names are unique, but the alias will have the name of the requested table to be used later.

Sample:

If we need to create an invoice form (a form that loads one invoice from an invoices list)
- tables to be used
    - invoice (invoice header) - filtered by invoice_id
    - invoice details - filtered by invoice_id
    - customers list (to populate a combo box, only few fields and customer ID)
    - products list
    - other look up tables if needed.

The command to be sent to the server is prepared by GETDATA like the following:

```
invoice_id = "ID00234"

PARAMETER ce_, invoice_id

this.getdata.addtable('invoice')
this.getdata.addparameter(m.invoice_id)
this.getdata.addtable('invoice_details')
this.getdata.addparameter(m.invoice_id)
this.getdata.addtable('customers_list')
this.getdata.addtable('products_list')


this.getData.gettables()
```

the actual command is:

DATA#"DBF","invoice|0,'ID00234'^invoice_details|0,'ID00234'^customers_list|0^products_list|0^"
- There is no need to construct this as it is parsed by the GETDATA into the command.

Server side there are required the following additional services:

invoice
invoice_details
customers_list
products_list

INVOICE service script:

```
PARAMETERS re_,invoice_id,temp_file,loginname    && re_ is used for backward compatibility, in previous command 0 is used all the time.
&& temp_file,loginname are automatically added by the data service command

.connect(server.main_connection)
.sqlcon.cname =  'invoice'
.sqlcon.sqlcommand = .getsqlstring('invoice')      && set the SQL script to be used from the SQL's list

* or use next
* .sqlcon.sqlcommand = "select * from invoice where invoice_id = ?m.invoice_id"
* not recommended to select all fields, only necessary to avoid additional data traffic even the data is compressed

RETURN .returnsql(temp_file)   && execute the SQL statement and return the cursor
```

Please check demo applications and tutorials available in the PSG VFP Client SDK for samples and tutorials.

2.8.2 SQL Scripts

It is difficult to place a large SQL script into the service code.
In order to simplify the service code, large SQL scripts could be stored externally. PSG offers a storage into a SQLITE database and a text editor to write/manage the scripts.

The actual PSG SQL editor cannot check for script errors. The scripts should be verified with the database server management tools.

2.8.3 Database Server

The PSG platforms do not depend on a specific database server.

A recommended list follows (alphabetic order):

Ms.SQL	any
MySQL	any
Oracle	any
PostgresSQL	any release that provides a working ODBC driver 8.2, 9.0
Sqlite	for few users/small projects

Any database/database server that provides an ODBC driver can be used.

The database server can be changed any time. It is only required to check if stored SQL scripts work properly with the new server.

Database server can be installed:
- on the same computer as PSG
- to another computer
- a database cluster (PostgreSQL, Oracle, ...)

The database connection string should be set using the server config utility in "Server" tab.

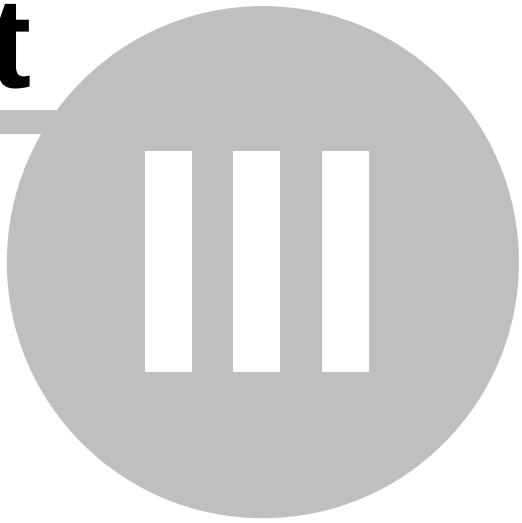
2.9 International Applications

PSG includes an international toolkit able to translate captions to different languages and which can be used with one application.
Check the LABEL class to see how it can be useful.

Use the International module from server config, utils tab.

Client side the user can set the language to be used by using "System" from the menu.

Part

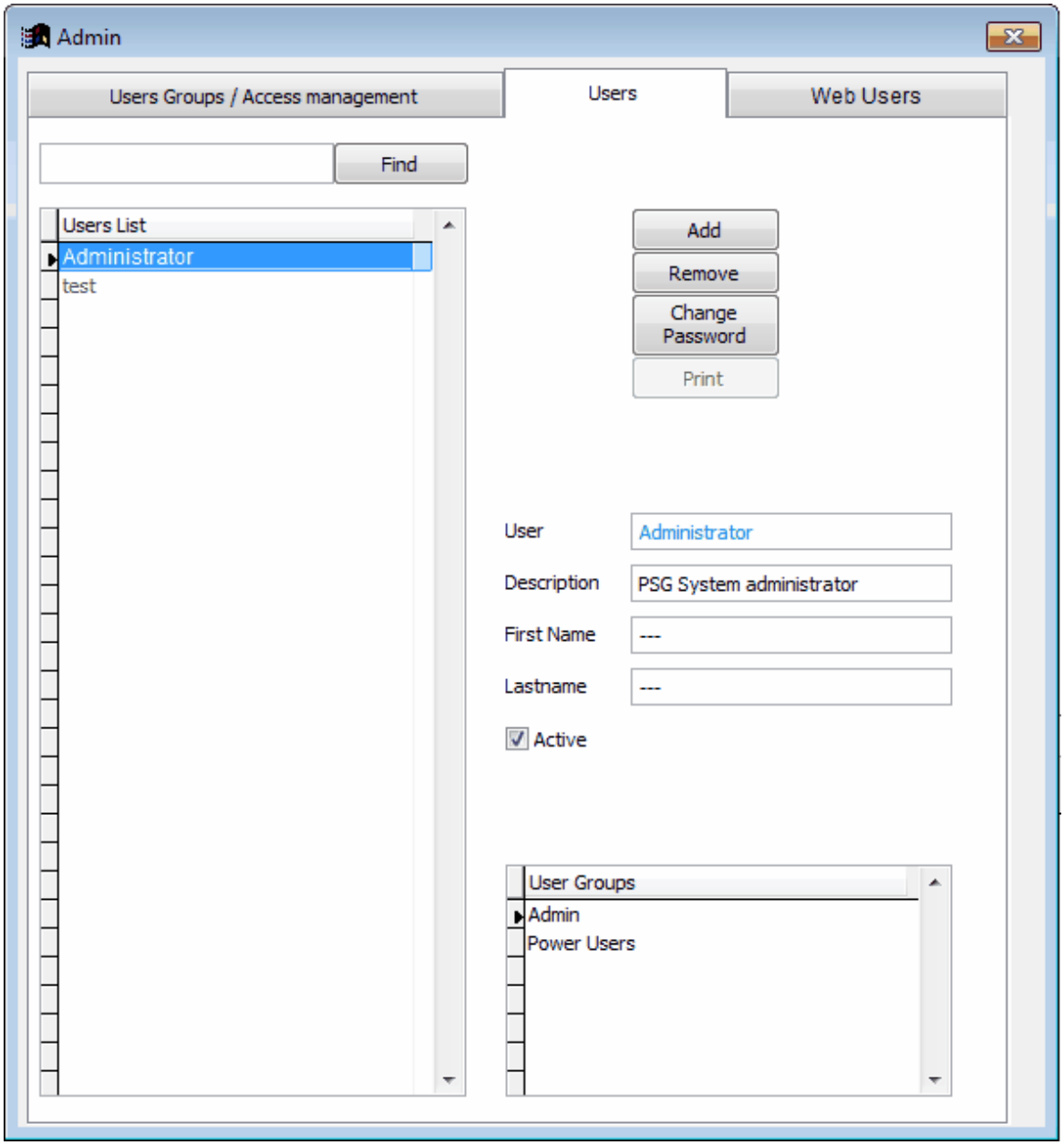


3 Users management

"Users management" section is available only for administrator in the client interface under "Admin".

3.1 Add user

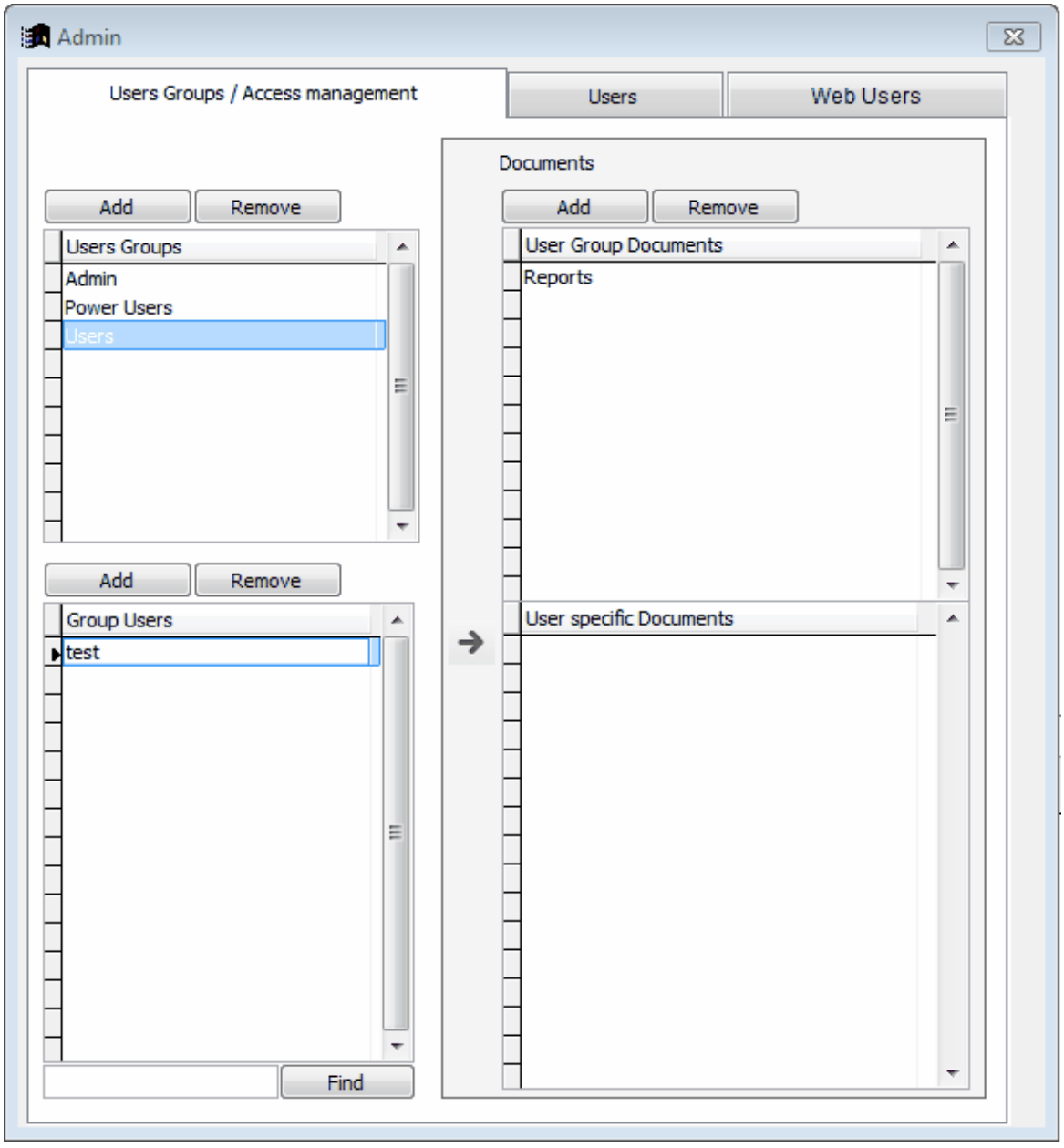
First step is to set the users accounts. Administrator account can not be removed.



Next step - set the users groups, add users per groups and rights to users groups.

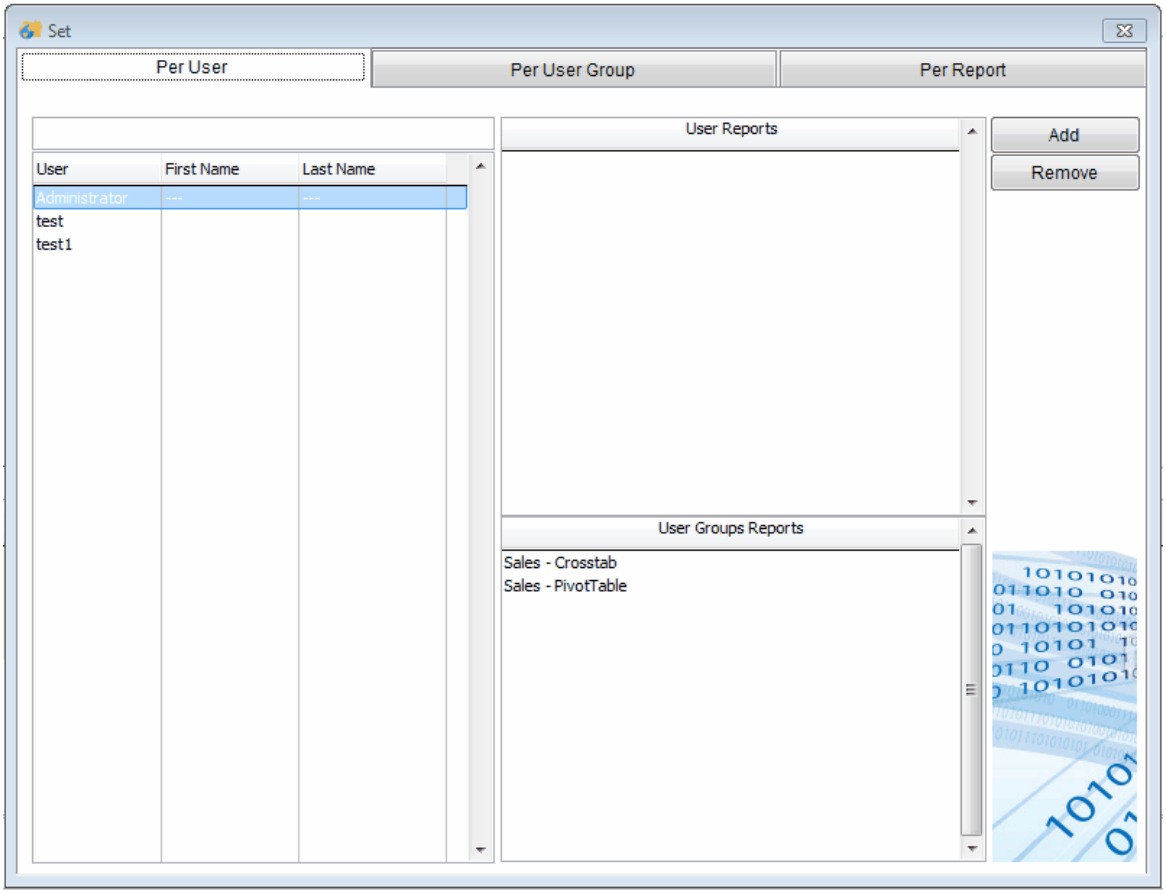
3.2 Set user rights

After setting the users groups, add users per groups and rights to users groups.
The "Reports" document should be added to the user group access rights list or to the user access rights list in order to allow the user to use reports.
The same for other application documents.



3.3 Set user rights - NET Reports

Reports access rights are configured in Reports menu.
Set the user or users groups access rights to declared reports, this can be done per user, per user group or per report.



Part IV

4 Distributing the application

Distribute the server:

- prepare the PSG server directory to be distributed.
    - Add a demo license file for the application or standard demo license file
- archive the PSG directory as ZIP file

The ZIP file could be used for server installation using the PSG server manager utility. (check the SERVER SDK manual)
The server database distribution should be solved separately depending on the database server provider:

- instructions to install a database server
- instructions to create the application database from a backup file
- the application database backup file

Distribute the client:

Client side only the PSG client application should be installed, distributed as MSI file.
A dedicated PSG installation kit for one application could be build if a personalized one is needed.
One PSG client could be used to connect more servers.

Files to be distributed:

- PSG server manager installation kit as MSI file
- PSG client installation kit as MSI file
- application PSG server ZIP file

Check also the PSG server SDK manual for server installation.

PSG Client installation kit will install all required components and the PSG client application and shortcuts.
Administrator rights are required when installing the kit for proper installation of all components, the PSG client will create local files for each computer user profile.
To connect a PSG server use the "setup" button into the "login" interface and set the server address (IP/NAME/DOMAIN and port).