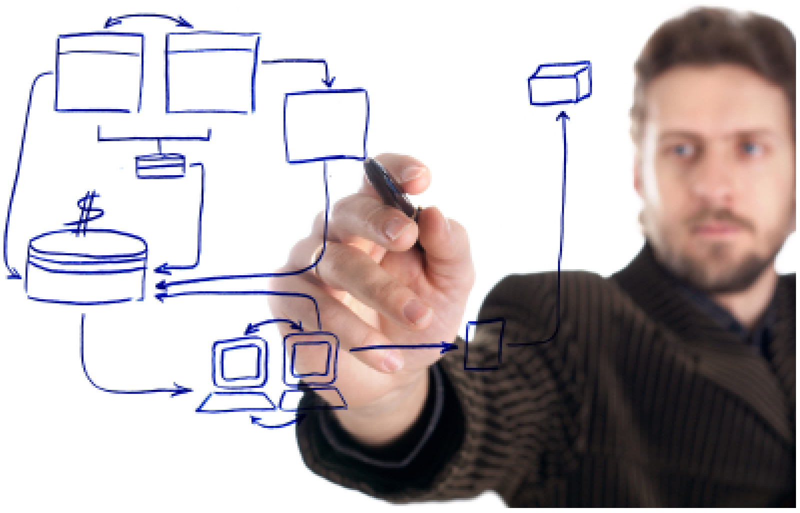


PSG SDK - Application server

PSG 2.0 Server Manual

by Programming Solutions Group

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

PSG SDK - Application server

Printed: September 2012

Publisher

...enter name...

Technical Editors

...enter name...

Table of Contents

Part I

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 application
- 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 bit. 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).

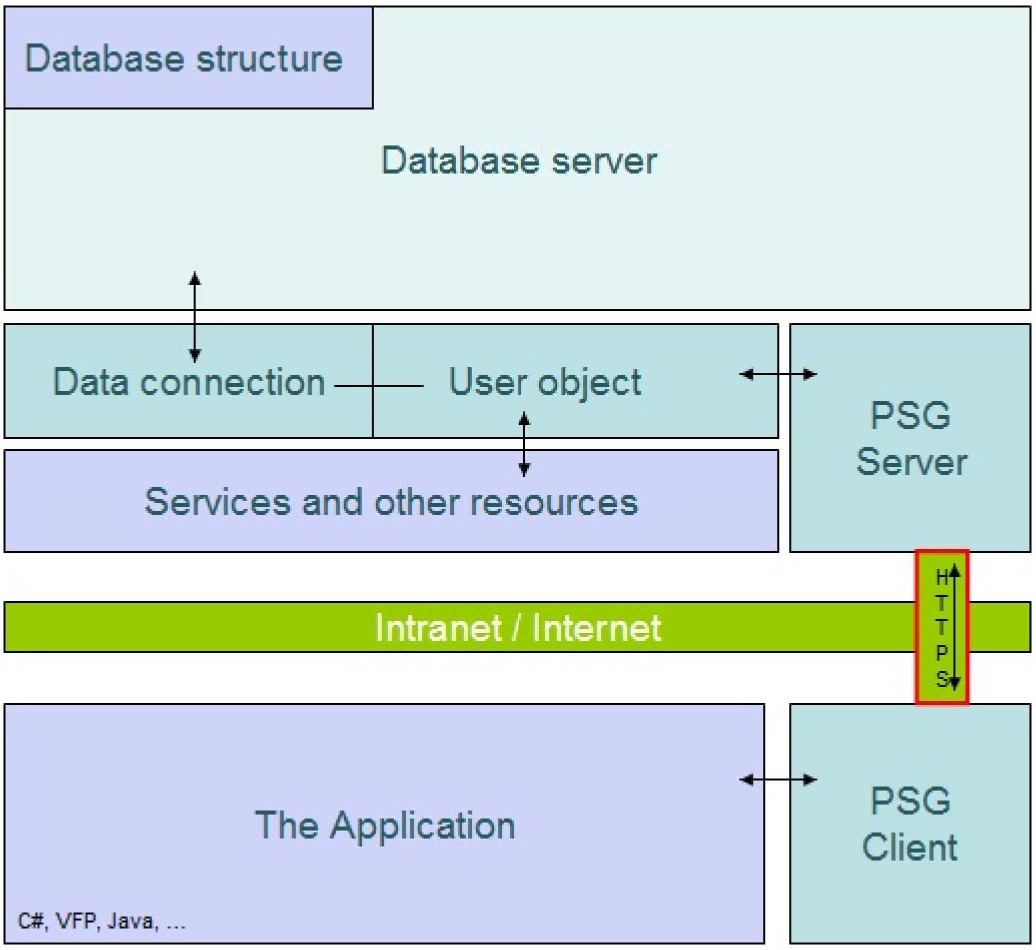
History

- August 2012 PSG goes public
- August 2012 - PSGSDK website ready
- November 2011 - "Programming Solutions Group LLC" where registered in Delaware to offer commercial support for the software.
- January 2011 - Final release 2.0 - closed solution used only for few applications
- October 2010 - Beta release 2.0 - closed solution used only for few applications
- January 2009 - Development for version 2.0 started
- technology changes to asynchronous communication for better reliability
- December 2006 - Final release 1.0 - closed solution used only for few applications

- November 2004 - developments starts - synchronous communication system
Please check the website for latest news.

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
- The 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.
The PSG WEB client it is not a WEB browser.

The platform offers an easy way to implement new services in minutes.

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), 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 the requirements. 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 by using industry standard solutions to provide scalability to servers farms.

* We have started with RESTFul architecture, but for application sake we do not respect entirely the RESTFul principles.

- For security reasons one part of the client login is not open.
- For stand alone servers, in order to increase the power (fast response/number of users per server) while the communication is asynchronous the server keeps client login session record in memory. Check the website for latest information related to PSG for enterprises solutions.

1.2 Server installation and configuration

Requirements:

Operating system:
Windows XP, Windows 2000 server or above, Windows 7
Recommended:
- Windows 2003/2008 server.
- Fast hard drives for better performance and estimated 1Mb memory for each user.

PSG does not depend on specific hardware or software installation excepting the operating system that should be MS. Windows.

Installation:

- installation kit
    - server components like DLL's, Activex and runtimes
    - a management interface for the PSG server instances
- The PSG server will be installed using the management interface from a ZIP archive.
- More PSG servers can run on the same hardware computer using different listening ports.

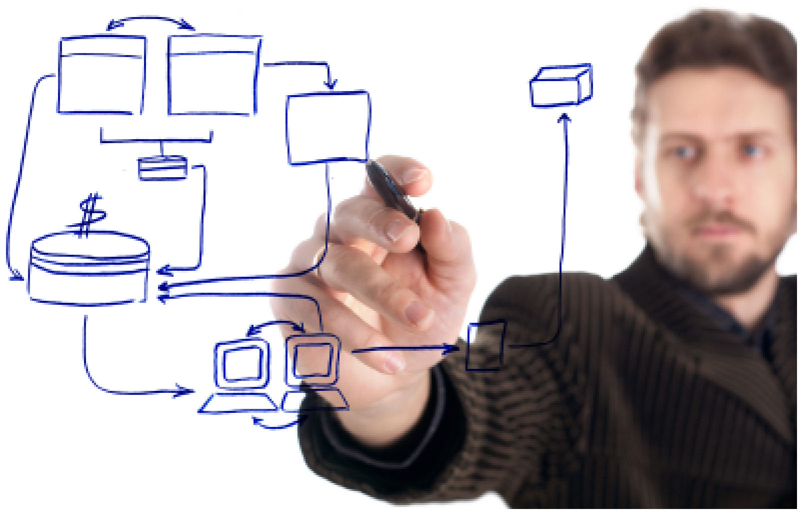
The database server should be installed separately.

To install the server management tool please use the CD provided or downloaded installation KIT.
Please check the installation kit readme.txt file for updated information.
The installation kit will install the system components as required.

After installation:
Set a default certificate using the certificate utility or add a verified certificate. Set the NT service for the new PSG server.
Please check "server configuration" in this manual.

For its services PSG opens two ports, one is SSL secured for application communication needs and the second one - a standard HTTP port for system help files. You will need to set your network infrastructure like firewalls and routers to allow access/route these ports according to your specific needs.

1.3 About



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

History

2012 - January - Enterprise release 2.0 A, based on collaboration of many PSG servers.

2011 - July - Client interface for Java.

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 Server Programming

The server is using HTTPS for communications. The PSG applications use a multi tier programming model.

All communication packages are available into HTTP body excepting larger data responses which are zipped and downloaded/uploaded as needed.

PSG programming model is object oriented. Server side services are implemented as scripts that can access the server objects.
Client programming uses templates and classes that have methods to deal with the communication object. HTML coding is not required.

Services programming is open.

All relational databases servers like Ms.Sql, PostgreSql, Oracle, MySql and others can be used, only an ODBC driver is required.
One application client module has no dependencies to the database server that can be changed as required by system organic grow.

2.1 PSG server

The client/server programming model is REST based on web services named here "services" to avoid confusion with standard web services.

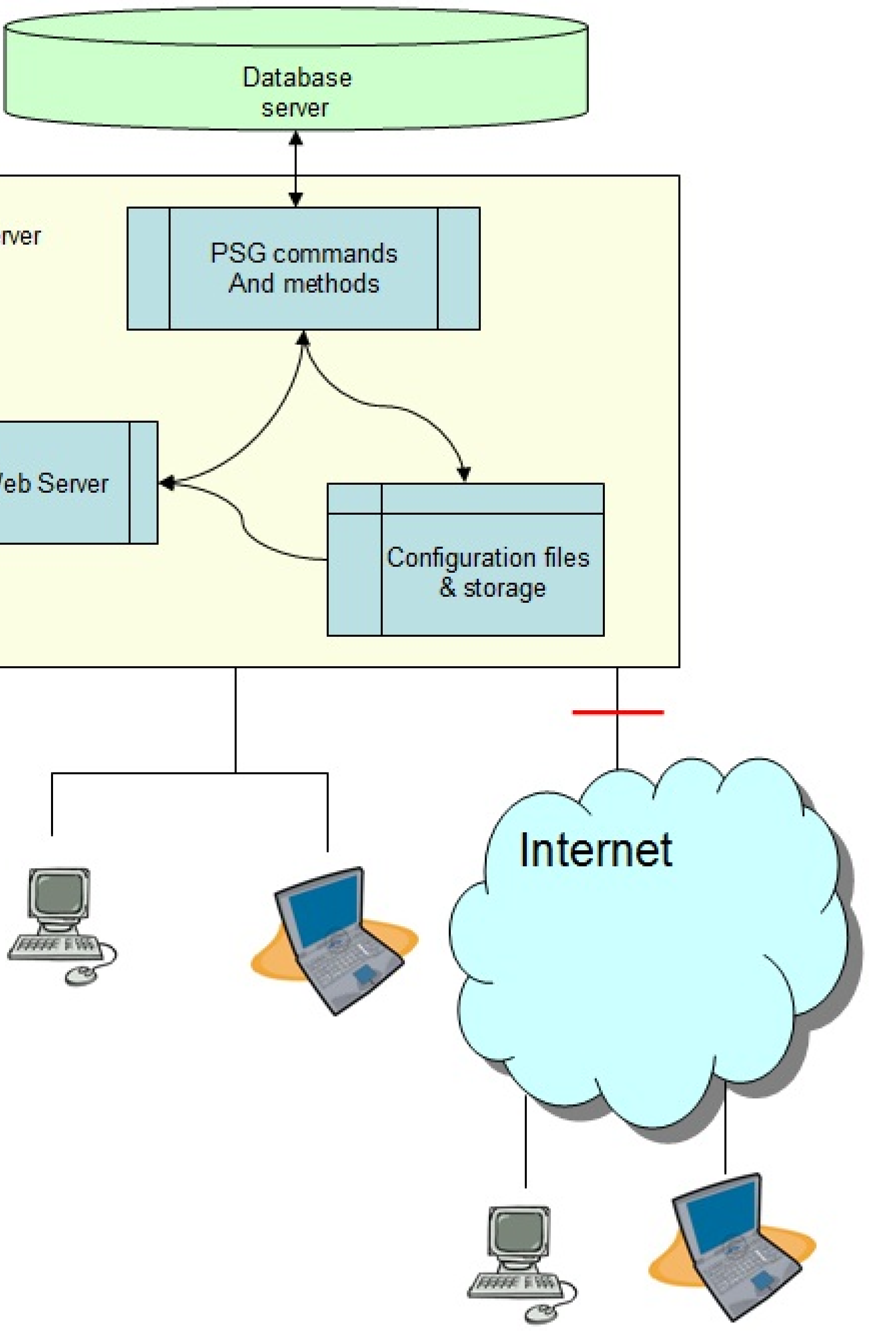
For security reasons PSG services could be used only by a PSG client application.
Secured bridges to other platforms could be build as requested.

The PSG services engine is implemented into the PSG WEB server, which enabled a fast response.
The services are implemented as scripts in different languages at programmer choice (C#, VBS, JS, VFP). No registration required.

The client wraps the command to a HTML body request message.
Client SDK provides templates and base classes that deals with the connection object automatically. The client side accepted messages could be easily extended.

Basic services are already implemented, the solution is ready to be used for general programming. Special needs require new services.

The server services list can be easily adapted to fit any needs.



2.1.1 Server object

Properties:

psgdata_connection	connection string to PSG internal database (users management, international file, server settings)

main_connection	connection string for main database (application database) - should not be changed at runtime
server_name	server name
server_path	server path
server_port	WEB HTTPS server is listening on this port WEB HTTP port (help server) is listening on server_port - 10 by default

Example:

m.port = server.server_port

Public Objects:

PSGUSR - users collection

2.1.2 Users collection

**PSGUSERS - ** users collection, each user has an user object here

server.psgusers

Methods:

getuserprop(<username>,<property to return>)

Available properties of user object:

- active_usr	returns .T. if the server accept connections from this user
- computer	returns user computer name
- logintime	returns datetime of last login
- remoteIP	returns user remote IP, or the user router's IP
- remotePort	returns user remote port or user router's port
- uid	returns user unique ID
- uname	returns user full name
- usrcode	returns user code
- online	Returns 1 if user is online

Example:

server.psgusers.getuserprop(loginname,'usrcode')

Returns the user code.

2.1.3 User object

Properties:

active_usr	returns .T. if the server accept connections from this user
computer	returns user computer name
logintime	returns datetime of last login
remoteIP	returns user remote IP, or the user router's IP
remotePort	returns user remote port or user router's port
uid	returns user unique ID
uname	returns user name, the name used for login
usrcode	returns user code
online	Returns 1 if user is online

Use GETUSERPROP of psgusers to access the USER properties

Example:

server.psgusers.getuserprop(loginname,'usrcode')

Objects:

CONNECTION

For each user the "connection" object is created at login and destroyed at log out.

2.1.4 Connection

Instantiated at user login, this object receives the PSG request from WEB server and prepares the response.
The communication is asynchronous, but the connection object is also used to keep the session key and the user logged in.

All requests come as commands for a specific service, accepted psgservices are loaded into the memory with connection object and are interpreted on request.

CONNECTION has his own child object SQLCON

SQLCON is a SQL connection to the database server. It offers all needed properties and methods to simplify the data transfer to database server.

Please see services for more details on using SQLCON and SQLCON methods and properties.

Methods:

ACCEPT	it is fired when a new command is received from the client, but can be also used from another script .accept(<command string>) Example: run a command from a script .accept('DELETERECORD#"test","test_id","W123rtede5",'+m.loginname)
GETSQLSTRING	large SQL scripts can be stored in SQL Scripts list for a better management .getsqlstring(<string name>) Example: .sqlcon.sqlcommand = .getsqlstring('psgidata')

CONNECT	Format:
	.connect(<connection string>)
	
	Example:
	.connect(server.main_connection)

RETURNSQL	Run the SQL command and return the result (used for data requests) Format: RETURN .returnsql(<temp cursor>) Example: RETURN .returnsql(m.temp_file)

2.1.5 PSGservices

PSG services are dynamic implemented from defined PSG services scripts. The editor is found in server configuration application in 'utils' page.
C#,VFP, JavaScript or VBScript can be used for services code.

Received commands are interpreted by the **connection** object.

Commands/requests are sent by the PSG client
<command_name>#<parameter1>,<parameter2>

Services rules:

- each service has at least one parameter <loginname> added by the PSG server, last parameter in list if there are more
- each service should return something as string, it should be a valid response or a new command for the client.
  <RETURN 'DONE#OK'> or <RETURN 'DONE#NOK'>

Services are implemented as simple scripts. Programming languages like C#, VFP, JavaScript or VBScript could be used with minor differences in implementation.

Some server and user objects can be used in code here.

Implemented services (default)

Services that start with PSG serve for system configuration, mainly related to user management. The modules using them are provided as they are. Recommendation is to not modify these services.
All major PSG services are presented here. Please check the PSG services editor to see the complete list. Services presented hereafter are based on VFP as scripting language.

CHECKRERCORDS	PARAMETERS table_name, primary_key,primary_key_val,loginname - it checks if one record exists in the main database returns RECORD#OK / RECORD#NOK

DATA

system service - should not be modified
it helps to get data from server

PARAMETERS data_command, logginname
returns tables/XML files to be used client side
data_command is parsed automatically from client side

for each requested table will be created a separated command named after the returned table name

Returns the zip archive file name of the results

Example:

we need to return the table TNAME with the parameter PARAMETER

DATA command (created by client constructor)

DATA#"DBF","TNAME|0,'PARAMETER'^" or
DATA#"XML","TNAME|0,'PARAMETER'^"

This format is handled automatically client side. There are clear methods to create such command string. For details please see the client SDK for details.

The command DATA is split in parts like TNAME service (the separator is '^')

TNAME should be declared as service as following :

```
PARAMETERS re_,real_parameter,temp_file, logginname

** last two parameters temp_file & logginname are automatically added by the server
** connection
** parameter re_ is necessary  for backward compatibility with older systems
** here the parameter is < real_parameter > it will be used within the SQL string
** add as many parameters as the application requires

** connect to the database server
.connect(server.main_connection)

** set the cursor name
.sqlcon.cname =  'tname'

** get the SQL string from declared SQLs
.sqlcon.sqlcommand = .getsqlstring('tname')

** return the temporary file
RETURN  .returnsql(temp_file)
```

the <tables> data commands should be declared under APPLICATION category using the commands editor

DELETERECORD	PARAMETERS table_name,key_name, key_value,loginname deletes one record returns DELETE#OK or DELETE#NOK
DELTEMPFILE	system service - should not be modified PARAMETERS file_, ccommand_, loginname Returns next client action/command
ECHO	PARAMETERS test_value ,loginname returns 'echo:' + test_value
GETDOCUMENT	system service - should not be modified prepare an application module for download (application modules are not in the server root)
GETIDATA	system service - should not be modified prepares the initial data on login - access rights and user details
GETUINT	The client application main interface is set on server, if local copy do not match the version or is not found the GETUINT command will prepare the file for download
INSERT	PARAMETERS insert_command, loginname used to insert one record the <insert_command> string is created client side automatically and parsed here to be used. Returns: INSERT#OK or INSERT#NOK
UPDATEFIELD	do not modify the service name. It is back linked to the connection object for parameters parsing (field value) PARAMETERS psgtable,psgkey,key_value,psgfield,field_value,loginname updates one field in the database, limited to 200 characters for text fields classes that automatically update the data can be used client side for each type of data control (text, combobox, spinner, checkbox etc) each class has code for communication with the server and send updates
UPDATETEXT	do not modify the service name. It is back linked to the connection object for parameters parsing (field value) It is used for large texts, can upload a reasonable amount of data,use the upload method to upload files.

Almost all services deal with data exchange, which is SQLCON object job.

If one received command is not found as a service an error message is sent to the client.
If the service identifies errors, an error message is sent to the client.

2.1.5.1 Using JavaScript or VBScript

JavaScript and VB-Script are implemented using Microsoft Script Control (`MSScriptcontrol.scriptcontrol.1`).

All accepted scripting languages can access same server objects, the only difference being noticed in the approach.

While VFP commands starts with "PARAMETERS ..." and end with "RETURN ...", for other scripting languages the parameters are passed to an object array and the return string should be passed back to an object propriety.

JavaScript/VB-Script list of objects to be used in code:

SERVER

CONNECTION

CONNECTION.PARAMETERS - the parameters array.
like something = connection.parameters[1]

CONNECTION.SQLCON

SQLCON

CONNECTION.RESPONSE should be set before the end of the script to avoid having a generic response like "DONE#" generated. This may not reflect the command result.

The implementations of JavaScript and VB-Script are new. For more documentation and news related please check the support forum.

2.1.5.2 Using C#

PSG services can be implemented using C#.

Sample and details on creating PSG C# services are provided within the Client SDK for C#.

2.1.6 SQLCON

SQLCON is the connector to the database server.

At start it is connected to the main database (server.main_connection). Connection can be changed to any database in your services code when required.

Properties:

cname	result cursor name (if the result is a cursor)

connectionstring	ODBC connection string
	each database server has a different connection string
	Please check http://www.connectionstrings.com for details on different servers
	connection string. The corresponding one for the main database is stored in server.main_connection
	
	Example:
	
	`IF .sqlcon.connectionstring # server.main_connection`
	`    .sqlcon.connectionstring = server.main_connection`
	`    .sqlcon.sqlstringconnect()`
	`ENDIF`
	
	Only the main database connection string is stored in a server property variable

sqlcommand	Set command to be executed by sqlexec method, any command accepted by the database server.
	
	Example:
	
	`.sqlcon.cname = 'test'`
	`.sqlcon.sqlcommand = 'select * from <table_name>'`
	`.sqlexec()`
	
	returns a cursor named 'test'

sqlresult	1 if executed method was successful -1 if executed method has errors
nhandle	database connection handle, used internally by sqlcon

Methods:

sqlexec	executes the previous set sqlcommand
sqlinsert	prepares and sends a INSERT command to the database server `.sqlcon.sqlinsert(<tablename>)` all required fields should be defined as variables before the command, return .sqlcon.sqlresult = 1 for success some database servers do not accept empty fields for date and datetime, use nulls for in case

sqlupdate	prepares and sends an UPDATE command to the database server
	
	.sqlcon.sqlupdate(<table_name>,<where clause>,<field list>)
	
	updates fields. Fields should be defined as variables before the update command
	
	Example:
	
	m.field1 = 1
	m.field2 = 'test'
	m.field3 = {2010-07-15}
	m.test_id = m.test_id_parameter
	
	.sqlcon.sqlupdate('test','test_id = ?m.test_id','field1,field2,field3')

sqlstringconnect	connects to the database, resulted 'nhandle' should be bigger than zero and sqlresult equal 1
sqldisconnect	closes the database connection

2.1.7 Transactions

Transactions commands are depending on database server type.

Could be implemented any time by using the SQLCON object to send commands to the server:

```
** Sample as VFP script ...

.sqlcon.sqlcommand='BEGIN TRANSACTION'
.sqlcon.sqlexec()

.sqlcon.sqlcommand='COMMIT'
.sqlcon.sqlexec()

.sqlcon.sqlcommand='ROLLBACK'
.sqlcon.sqlexec()
```

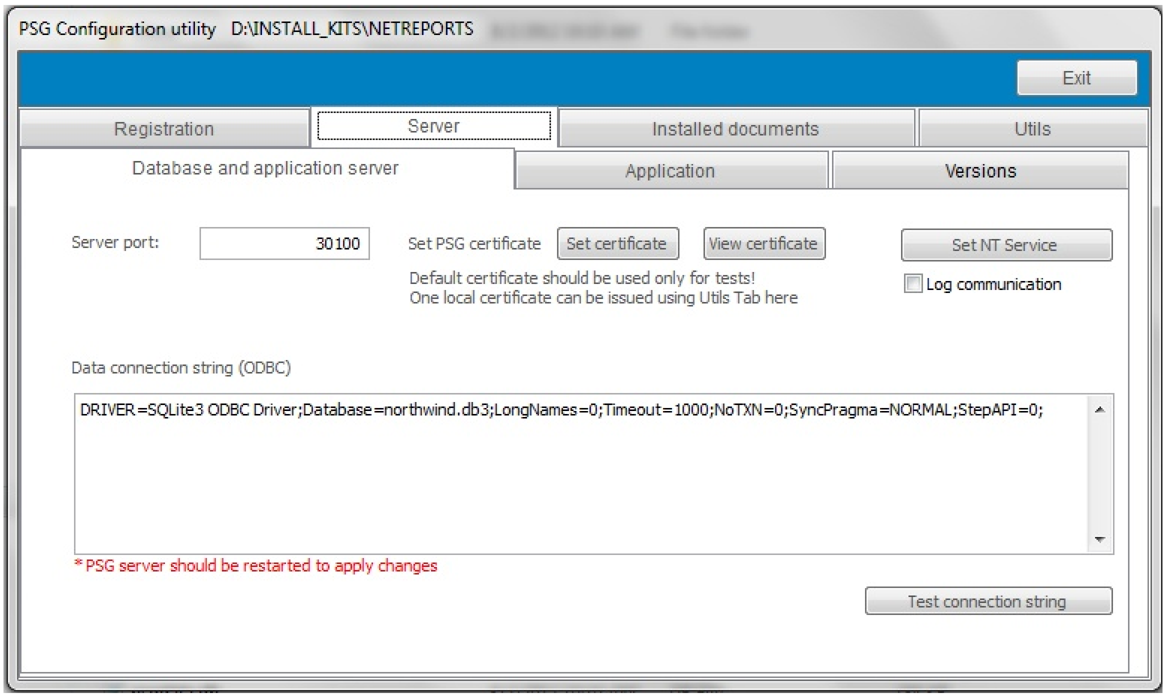
Codes like previous could be inserted in any PSG service when required to raise the application's reliability.
Please check your database server help file for accepted commands syntax and features.

2.1.8 Server configuration

CONFIG utility is used here. Found into the PSG server folder, the configuration utility is config. exe .
All others utils can be launched from CONFIG in utils page.

SERVER configuration:

Database and application server



Server port: 30100

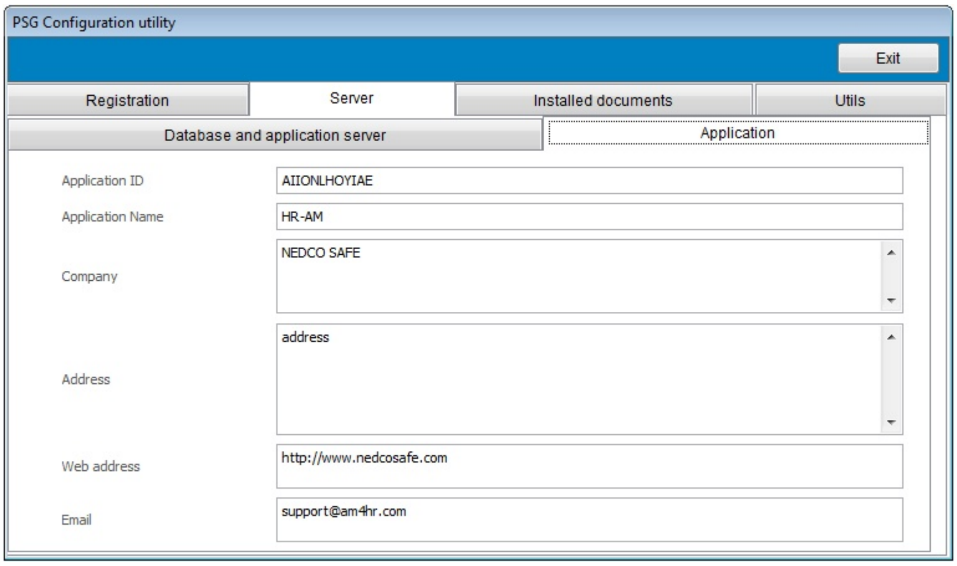
Default certificate should be used only for tests!
One local certificate can be issued using Utils Tab here

```
DRIVER=SQLite3 ODBC Driver;Database=northwind.db3;LongNames=0;Timeout=1000;NoTXN=0;SyncPragma=NORMAL;StepAPI=0;
```

* PSG server should be restarted to apply changes

- server port - the listening port of the PSG server
- set certificate - set the PKI certificate that will be used by SSL encryption engine. Changing the default certificate is recommended (check the utils page for details on new certificate definition)
- view certificate - preview of the current used PKI certificate
- data connection string (ODBC) - the place where it is stored the main database connection string. Press the "Test connection string" to check if the syntax is correct.
- Set NT Service - set the server service properties (start and stop).
- Log communication - log file of each server request
- Database type - one type, SQL servers and alike
  covers:
    - PostgreSQL
    - Ms. SQL
    - Oracle
    - MySQL
    - SQLITE
    and other relational SGBD that provides ODBC drivers and use SQL syntax

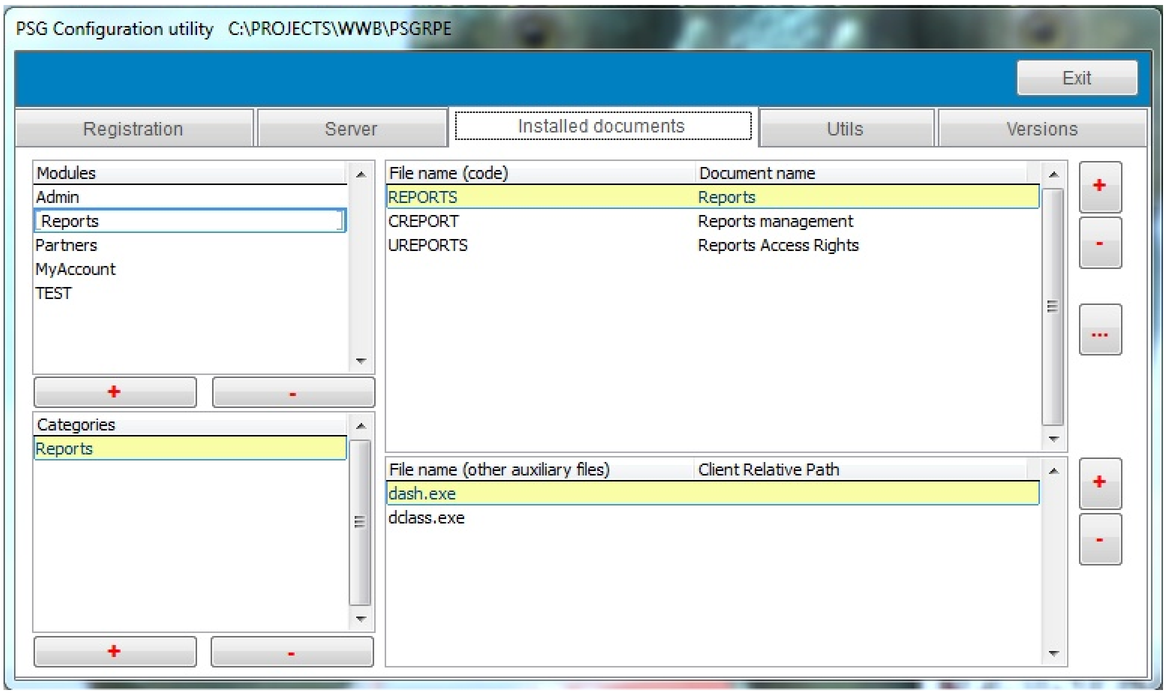
Application



Application ID	AIIONLHOYIAE
Application Name	HR-AM
Company	NEDCO SAFE
Address	address
Web address	http://www.nedcosafe.com
Email	support@am4hr.com

Stored information will be used in "About" window on client loaded application. The information is loaded from the licence file.

2.1.8.1 Server configuration - application modules



Modules
Admin
Reports
Partners
MyAccount
TEST

Categories
Reports

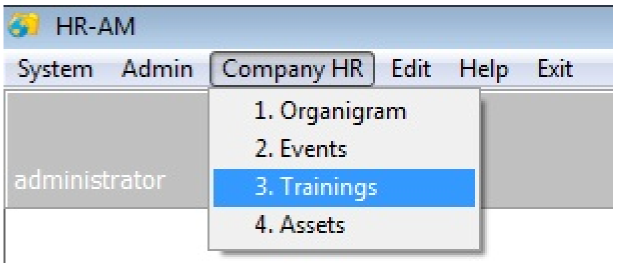
File name (code)	Document name
REPORTS	Reports
CREPORT	Reports management
UREPORTS	Reports Access Rights

File name (other auxiliary files)	Client Relative Path
dash.exe	
dclass.exe	

The 'Installed documents' :

By "documents" there are defined application modules executable/interpreted files (EXE, DLL, JAR).

In order to make a module available it should be registered here. The defined modules and "Document name" will create also the main application menu client side.

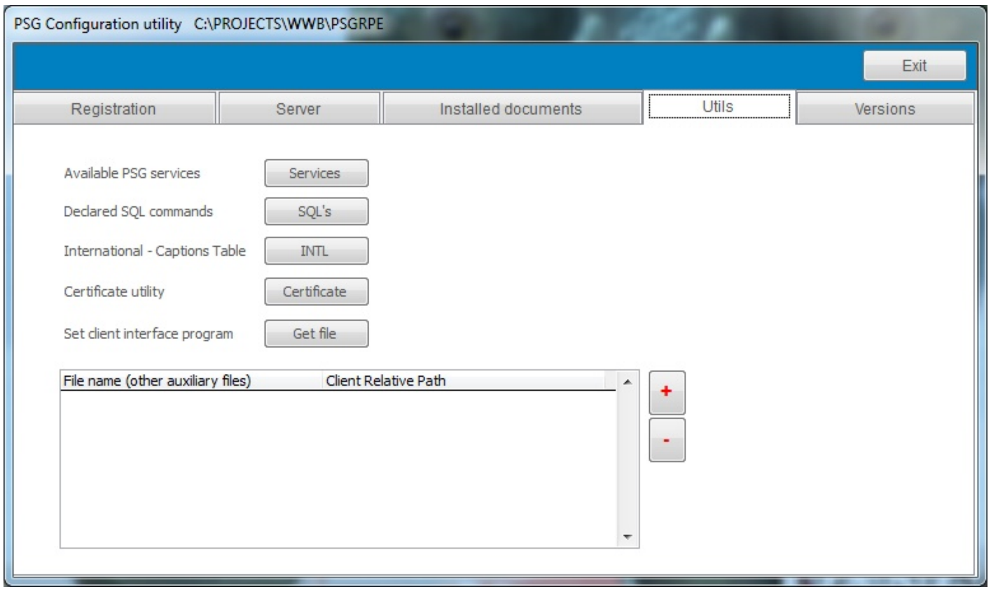


When adding the application file here it will be registered as the last one to be used (the MD5 checksum is stored along with the file name).
To register a new release of one application module, just publish here the new file.

To manage the user rights for access to the published documents the 'ADMIN' module should be used. This module is installed client side and could be accessed by the Administrator using the PSG client. In previous picture the 'Admin' menu could be seen as the login name was 'administrator'.

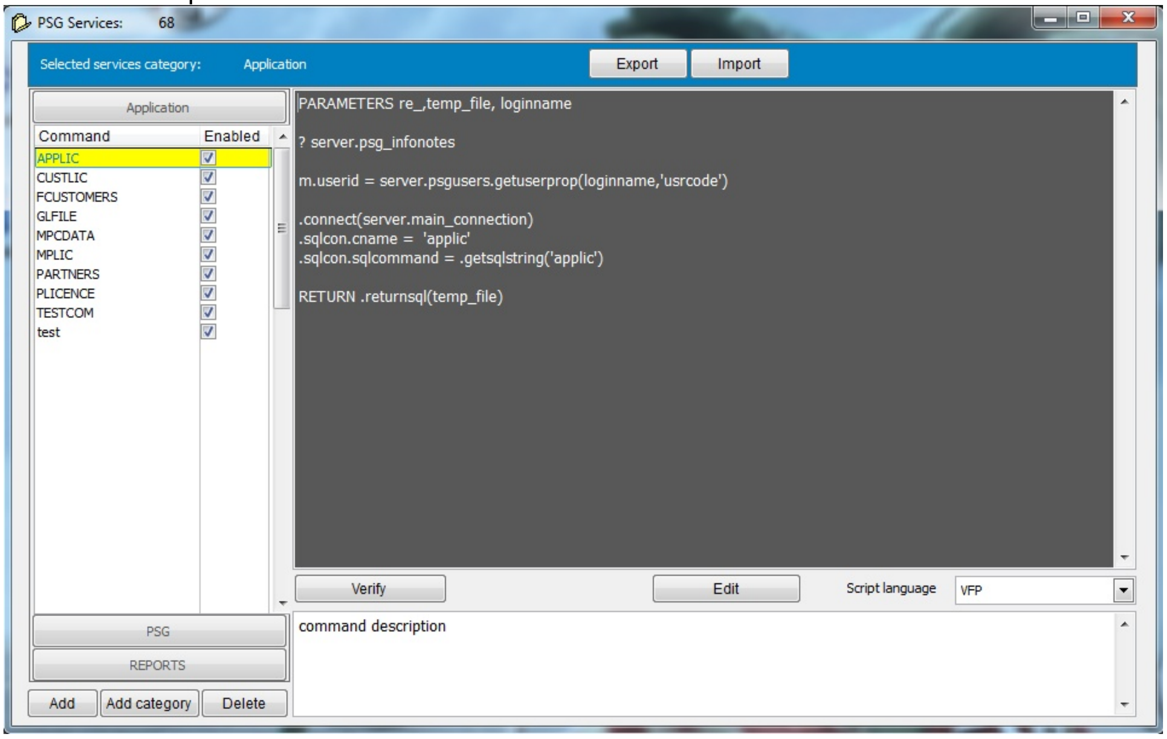
2.1.9 Utils

Found in server configuration module (Page Utils):



Available PSG services

The service scripts editor



```
PARAMETERS re_,temp_file, loginname

? server.psg_infonotes

m.userid = server.psgusers.getuserprop(loginname,'usrcode')

.connect(server.main_connection)
.sqlcon.cname = 'applic'
.sqlcon.sqlcommand = .getsqlstring('applic')

RETURN .returnsql(temp_file)
```

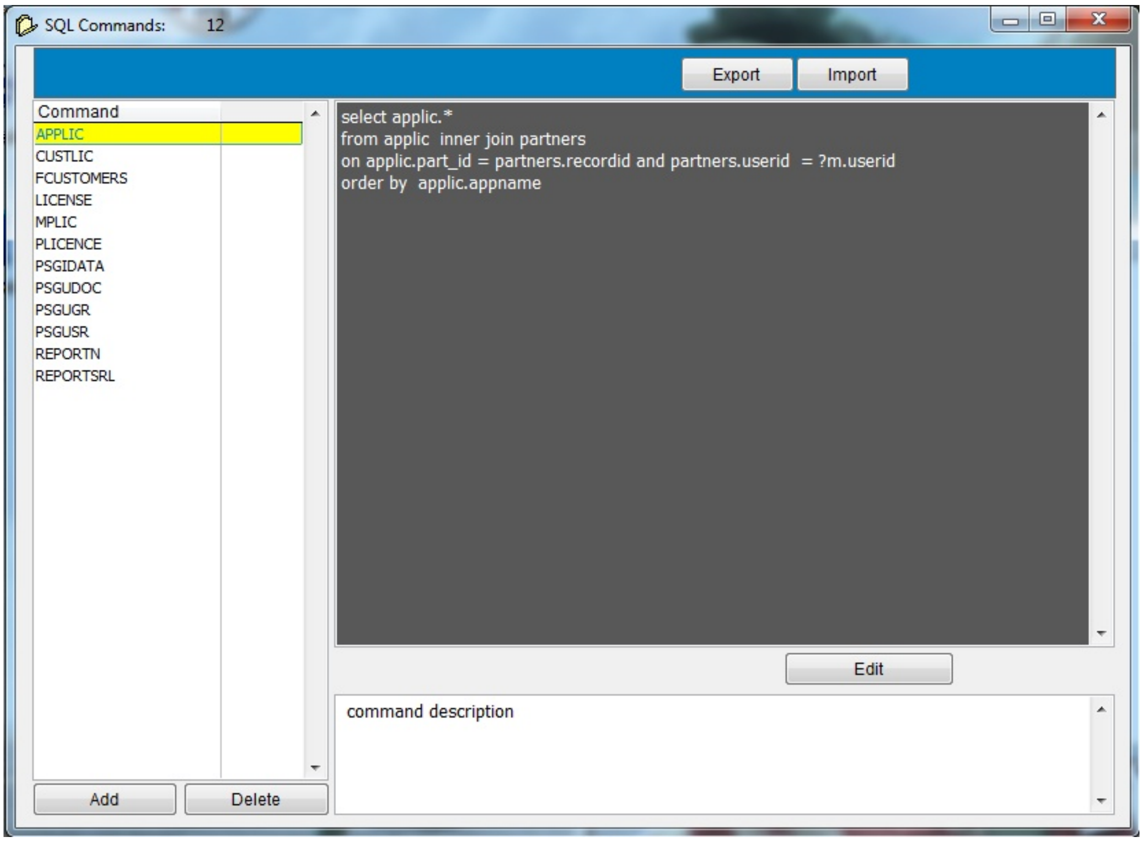
Services are under categories: PSG - platform common services; REPORTS - services used by the reports engine; APPLICATION - custom services for the application.

Services are registered as scripts working with server objects to create the client response message.

For simple applications where nothing than data transfer is required between the client and server, the only services implemented here are related to record sets request. PSG serve data as zip archive with XML or VFP database files, the client maps data as required into the application.

Declared SQL scripts

The SQL scripts editor



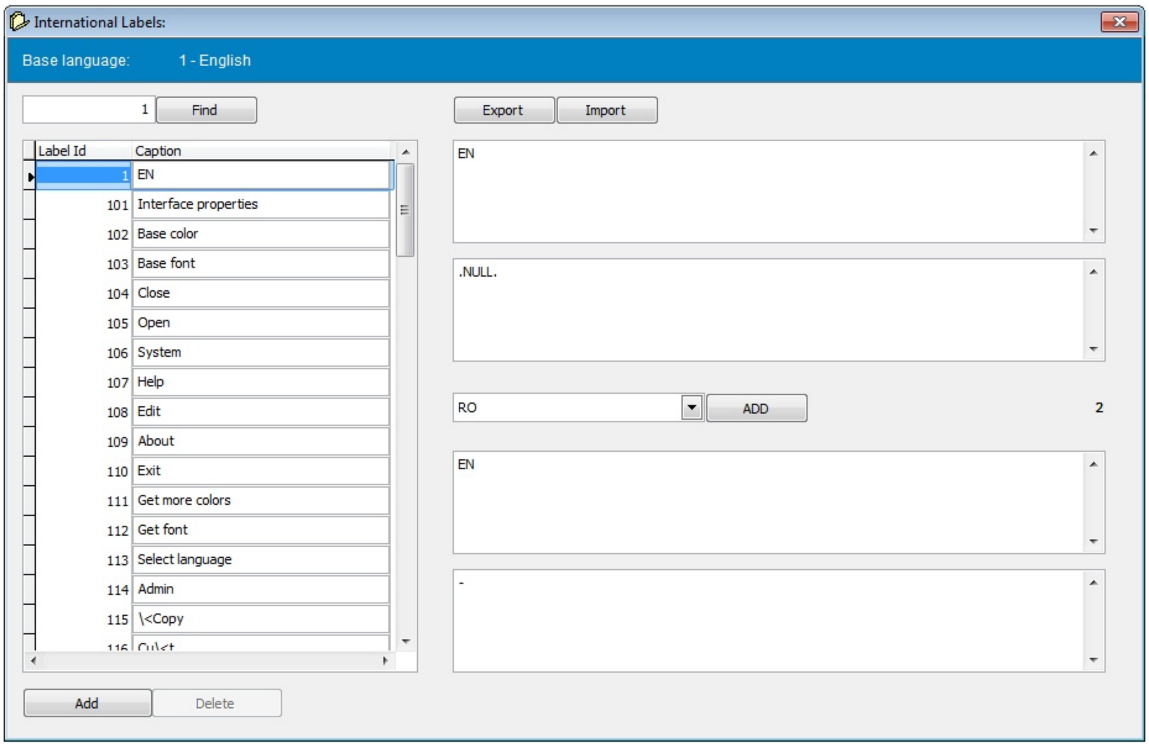
```
select applic.*
from applic  inner join partners
on applic.part_id = partners.recordid and partners.userid  = ?m.userid
order by  applic.appname
```

The SQL's should be written according to the database server accepted SQL syntax. SQL's written here could be retrieved by the user connection object. The purpose of this storage is to offer an easy way to have big SQL scripts available.

A SQL statement can be used as follow:

m.sqlscript = .getsqlscript(<sqlname>)

International - Caption's table (INTL)



Base language: 1 - English

1

Label Id	Caption
1	EN
101	Interface properties
102	Base color
103	Base font
104	Close
105	Open
106	System
107	Help
108	Edit
109	About
110	Exit
111	Get more colors
112	Get font
113	Select language
114	Admin
115	\<Copy
116	Cu\<t

EN

.NULL.

RO 2

EN

-

PSG platform can be used to build multi languages applications. To do that a table with translations should be created and updated.
Each 'text' here has a language ID and one label ID. Into the client application declared labels should be mapped to controls as LABEL, HEADER caption, PAGE caption and wherever are needed using the ID's.
Please check the PSG CLIENT SDK for more information related to each programming language. The definition is the same for all, difference being the side client implementation.

Set client interface program to be used with the installed PSG application.

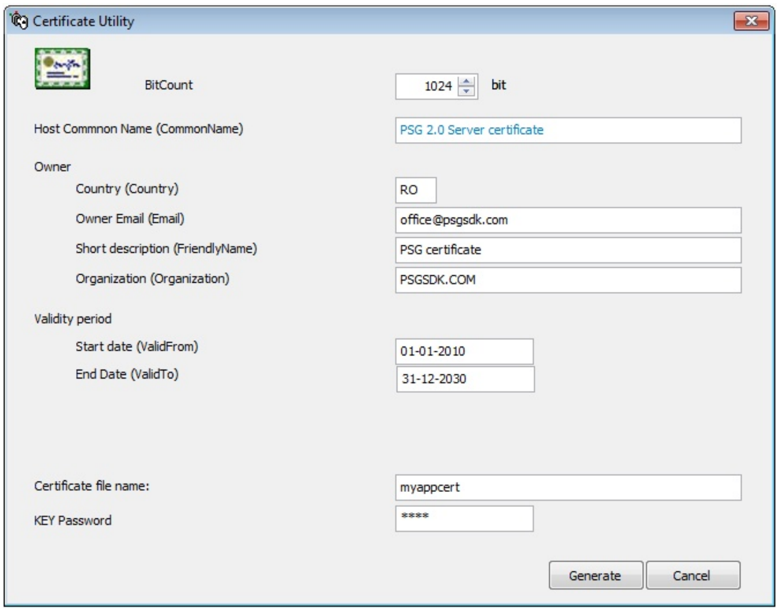
The client interface program is the main application that runs on the PSG client, deals with the application menu, data communication with the server and the modules MD5 security check. It is open and can be modified and used according with the application requirements. This main module is certified by the server and loaded when first use or when is found different than the module stored server side. The client login application loads the interface and passes the necessary parameters to enable the communication with the server.

Main modules are to be found for each supported programming language.
Please check the PSG CLIENT SDK for more information related to each programming language.

Certificate utility

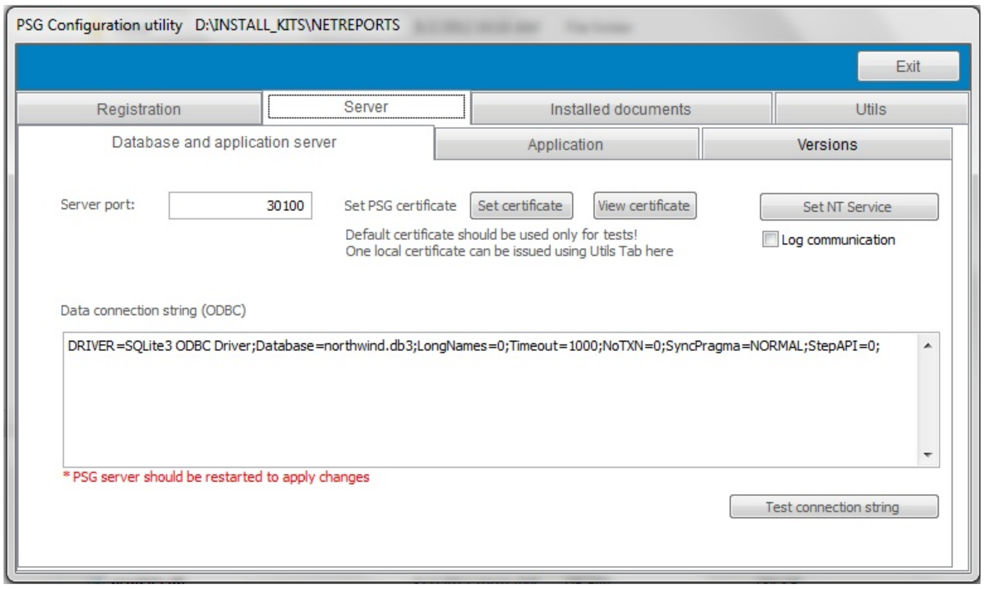
PSG platform uses HTTPS for data communication between clients and the server.

HTTPS encryption is SSL and uses public keys to encrypt data. In order to use PKI infrastructure a certificate is needed. While there are certified certificates created by Verisign and other companies alike, here it is also possible to create a certificate named 'self signed'. It offers protection as well but cannot be validated by a third party service as Verisign (the client recognizes, and shows the nag screen seen in WEB browsers for HTTPS not signed certificated). PSG client will make no difference between the signed or self-signed certificates, as the only person administrating the certificates is the server administrator. In this case the main objective becomes offering SSL protection.
A signed certificate is needed when the HTTPS PSG is used as WEB server, to avoid the 'not certified' messages in the WEB browser.



BitCount	1024 bit
Host Commnon Name (CommonName)	PSG 2.0 Server certificate
Owner	
Country (Country)	RO
Owner Email (Email)	office@psgsdk.com
Short description (FriendlyName)	PSG certificate
Organization (Organization)	PSGSDK.COM
Validity period	
Start date (ValidFrom)	01-01-2010
End Date (ValidTo)	31-12-2030
Certificate file name:	myappcert
KEY Password	****

This utility is to create a certificate.
To set the default certificate use the "SET CERTIFICATE" button in the "SERVER" page.



Server port: 30100

Set PSG certificate

Default certificate should be used only for tests!
One local certificate can be issued using Utils Tab here

☐ Log communication

Data connection string (ODBC)

```
DRIVER=SQLite3 ODBC Driver;Database=northwind.db3;LongNames=0;Timeout=1000;NoTXN=0;SyncPragma=NORMAL;StepAPI=0;
```

* PSG server should be restarted to apply changes

2.2 WEB server

HTPS web server can be used with any Mozilla compatible web browser.
Available web pages/sites should be published to server root directory (webroot).

This solution comes naturally with the server application type. There are others more powerful servers as Apache or IIS.

Static websites are fully supported, however there is under testing an option for dynamic pages not a priority at this time.
Please check the website for latest news.

2.2.1 Dynamic Pages

Scripting language is experimental.
- FoxScript
- JavaScript
- VB-Script

2.2.1.1 FoxScript

FoxScript will be checked for HTM or PSG extension requested pages.
Experience with HTML code is required as the script response should be proper formatted to fit the web pages design.

Special character strings are used to declare the script into the page:

<% to start the script
%> to end the script

All text between the '<%' and '%>' will be interpreted as fox script.

```
<%
PARAMETER user,sqlcon

* code here
* set the response string

RELEASE SQLCON
RETURN <response>
%>
```

Two parameters are passed each time to the script. The 'PARAMETER statement is mandatory for each script.
Both are objects that can be used in code:

USER	WEB user object, used to retrieve logged user properties, not the same object as USER in PSG server
SQLCON	a SQL connection object, used to retrieve data from the database, sqlcon should be initialized each time (it will not be used any time and connections consumes the database server resources)

SQLCON should be released at the end of the script!

Example:

```
<%
PARAMETER user,sqlcon

sqlcon.cname = 'psgusr'
sqlcon.ConnectString = server.psgdata_connection
sqlcon.sqlstringconnect()
sqlcon.sqlcommand = 'select * from psgusr'
sqlcon.sqlexec() && retrieve the PSG users list

response_string = CHR(13) + 'Database users/ codes' + CHR(13) +
'----------------------------' + CHR(13)

scan
        scatter memvar
        response_string = response_string + psgusr.uname + psgusr.usrcode
        response_string = response_string + str(server.psgusers.getuserprop(alltrim
(psgusr.uname) ,'online'))
        response_string = response_string + CHR(13)
endscan

response_string = response_string + '----------------------------'
response_string = response_string + CHR(13)

use
sqlcon.sqldisconnect

RELEASE sqlcon

RETU DTOC(DATE()) +CHR(10) ;
+CHR(13)+;
server.caption+;
+CHR(13)+'server:'     + server.server_name+;
+CHR(13)+'connection:' + server.main_connection+;
+CHR(13)+response_string

%>
```

The response in this sample is simple text without any HTML tags. Response will be inserted into the HTML page to be sent to the client. HTML tags should be inserted into the response body. HTML tags are not a subject for this help file. Please check HTML coding for more information's.

2.2.1.2 JavaScript

Under development.

Same as FoxScript, just the scripting language is JavaScript.
The page extension should be PJS, like start.pjs .

2.2.1.3 VBScript

Under development.

Same as FoxScript, just the scripting language is VBScript.
The page extension should be PBS, like start.pbs .

2.3 Database server

PSG as a platform for database applications requires a database server or a flat tables database to be installed, and it is generally independent of the chosen database server.

While is not dependent on any database server procedures, nothing keep your application away from them. Just be aware that database server stored procedures link you to a specific database server.

The database server is not provided with PSG platform. Even if it should be functional with any relational database that offer an ODBC driver, it was tested with :

- Ms.SQL server
- PostgreSQL
- Oracle
- SQLite - can be used with several users and not heavy data loading.

(the order in this list is arbitrary)

Please refer to your database server help and manual for installation and maintenance.
Please check the support forum for an actual list of supported databases.

ODBC is the fastest reliable way to communicate with the relational database server by using REST technologies as our several stress tests proved.

See server configuration.

2.3.1 Transactions

Depending on database server, transactions can be implemented by sending commands to the database like:

BEGIN TRANSACTION, ROLLBACK, COMMIT, END TRANSACTION

This should be implemented into PSG services that work with data:

Example:

```
.sqlcon.sqlcommand = 'BEGIN TRANSACTION'
.sqlcon.sqlexec()
```

Please pay attention to properly close the opened transactions, given the fact that this is not done automatically!

2.3.2 Advices

General database construction rules should be followed as much as possible based on your application requirements.

Load only the data that you really use at one moment client side. PSG can load thousand records at a time, but it is not mandatory to do it every time.

We do recommend the usage of unique string generated primary keys, like UIDS for C# or sys (2015)+PSG userID for VFP.
Autoincrement has a great value for one tier but here it is not the case. By default, one insert command comes with the key from the client to avoid unnecessary traffic to load the key back to the client. This way an integer autoincrement key is not functional. In order to continue to use autoincrement integer keys, the INSERT command should be modified server side and client side accordingly. Feel free to use the best approach that helps your application.

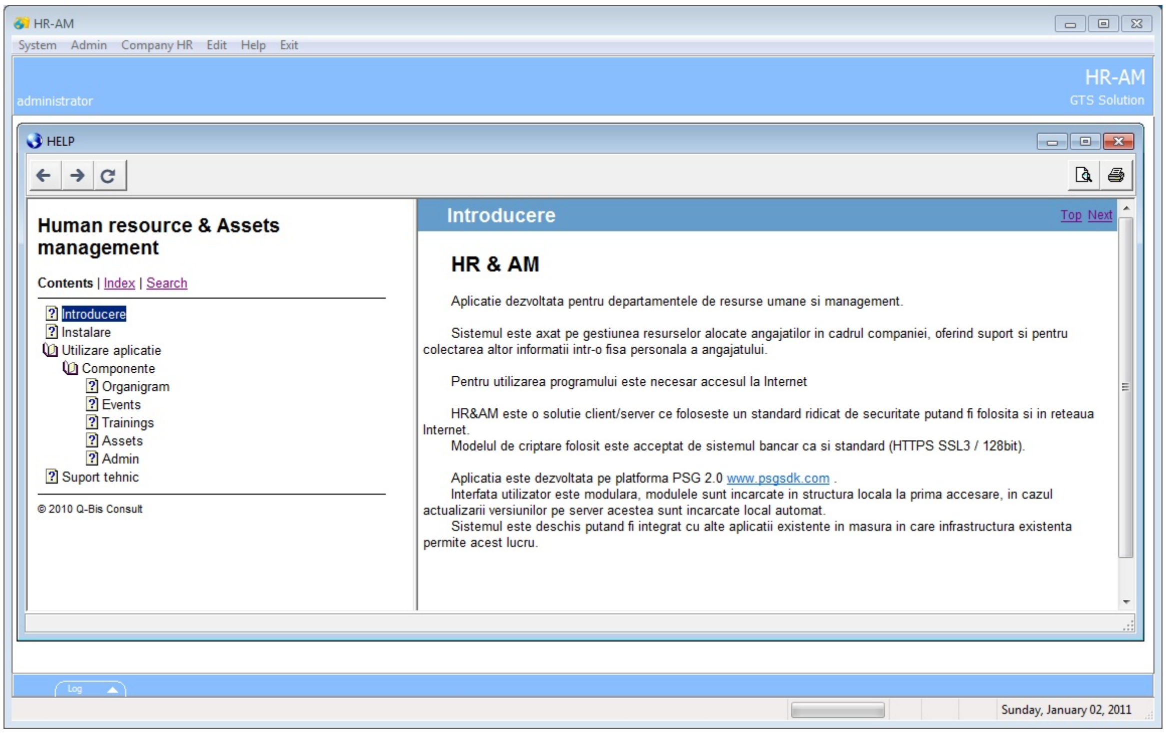
2.4 International

PSG platform can be used to build multi languages applications. To facilitate this a table with translations should be created and updated.
Please check UTILS in PSG server chapter.

2.5 PSG - application help

The PSG server opens two ports for listening.

The second opened port is used to serve HTML pages that can be used as help files.
Client side an installed WEB browser is opened or a small embedded WEB browser (like next picture).



Human resource & Assets management

Contents | Index | Search

- Introducere
- Instalare
- Utilizare aplicatie
  - Componente
    - Organigram
    - Events
    - Trainings
    - Assets
    - Admin
- Suport tehnic

© 2010 Q-Bis Consult

Introducere Top Next

HR & AM

Aplicatie dezvoltata pentru departamentele de resurse umane si management.

Sistemul este axat pe gestiunea resurselor alocate angajatilor in cadrul companiei, oferind suport si pentru colectarea altor informatii intr-o fisa personala a angajatului.

Pentru utilizarea programului este necesar accesul la Internet

HR&AM este o solutie client/server ce foloseste un standard ridicat de securitate putand fi folosita si in reteaua Internet.
Modelul de criptare folosit este acceptat de sistemul bancar ca si standard (HTTPS SSL3 / 128bit).

Aplicatia este dezvoltata pe platforma PSG 2.0 www.psgsdk.com .
Interfata utilizator este modulara, modulele sunt incarcate in structura locala la prima accesare, in cazul actualizarii versiunilor pe server acestea sunt incarcate local automat.
Sistemul este deschis putand fi integrat cu alte aplicatii existente in masura in care infrastructura existenta permite acest lucru.

The HELP pages can be created with any HTML designer. Help site is published into HTMLHELP folder under server folder.
Start page is index.htm, and client side a specific page can be set to be opened for specific help message into the application at one time.

One simple Internet browser is used with the help system, this is dependent on Internet Explorer. In case that Internet Explorer is not installed any other WEB browser could be set to be used in the "System" utility program (client side).

Part III

3 PSG Client programming

PSG client:

- login interface
- main user interface
    - client communication engine
        - communicate with the server
        - file download/upload
    - application modules

PSG is a WEB client/server optimized for data transfer to replace WEB pages. PSG uses REST web services model.

Data transfer protocol is HTTPS.
The connection between client and server is asynchronous.

Applications for PSG are developed with .NET C#, VFP or JAVA.

PSG client starts with a login module, and the list of available servers can be set here.
The login module loads user settings and opens the main application.

All elements excepting the login module can be customized or recreated to answer all possible requirements.

Security concept:

PSG is not as permissive as WEB environment. The PSG application model targets business applications.

- login module authenticates the user and launches the main application passing a session id key.
- all commands send later from client must use the session id key (login name and password checked)
- main module loads the user access rights to a variables array
- only one method deals with application module initialization and launch, which is done after verifying the MD5 checksum of the files with the registered one on server:
    - if module does not exist locally, it is downloaded from the server.
    - if exists but the checksum is not validated, it is downloaded from the server.
    - if the module validates the checksum is loaded.

The client sends 'KEEPALIVE#' messages if no activity occurs. In case that the clients stops sending keep alive messages (computer hibernates or sleep), the session is recovered when the computer is online if no other login occurs for the same user in the meantime.
If the server is restarted (for any reason) the client should login again starting with the login module (the application should close), the session's ID's are in the server memory for security reasons.

3.1 Overview

PSG client installation:

The installation kit can be downloaded from www.psgsdk.com. By default PSG is installed to "Program Files" directory, and the start application will create a working directory for each computer user under the user profile in "Application Data" directory.

After installation, server address should be set into available servers list on login module. A server address has two parameters; the server/router IP or domain name and the server port (listening port of the server or the router if the address is translated - recommended option for Internet).

The default main user interface will be downloaded from server at first login (for each server that will be used).
All PSG application modules will be downloaded from server at first use or when new releases are installed on server.
The client maintenance is close to zero. In case that the application requires new DLL,ACTIVEX controls they will be installed using an installation kit.

Each PSG session starts with user login when a session key is passed to the client. Each PSG command sent to the server should be preceded by the session key.
The application programmer does not deal with the communications objects as the specific commands are embedded into classes that should be used.

The PSG Client SDK contains all the necessary data to start programming a PSG based application (download from www.psgsdk.com):
- the installation Kit's and instructions files
- programming language specific manual
- sample applications based on well known databases like MS. Nortwind SQL database sample.
- templates and classes
- distributable files

3.2 Main User Interface

Main user interfaces:
- offer the main menu of the application
- deal with all client/server communications
- load and run application modules
  - verify the modules MD5 checksum with server values
  - download modules from the server

PSGCON object is instantiated in main interface and deals with all the communications with the server.

F1 starts a help web browser that opens the help site on the server.

More details on main user interface should be found on client programming SDK.

3.3 Programming

Programming PSG applications is easier by using already created classes and templates.

The programming model is object oriented and the created applications use a multi tier architecture.

The client side programming is not part of this manual. Please refer to the SDK programming language specific for programming manual.
Downloads are available on www.psgsdk.com.

Part IV

4 Distributing PSG application server

Please check the psgdistribution.txt file within the psgsdk kit. Small differences may vary from one revision to another.

The basic module offers an installation kit for a PSG Server management console which also facilitates installation for all the required components. The current PSG application server is distributed as ZIP file and installed within the management console.

For the server to be functional it is required a valid licence file.
For demo purposes one generic limited DEMO licence file is provided, and can be used on any computer without registration.

The license for the server uses a server id code. For more details please check the licensing agreement.

Free licenses could be provided in some particular cases for eligible beneficiaries such as:
- non profit charity organizations
- state or government institutions:
    - public hospitals offering free services in a significant percentage and any other similar establishments
    - public schools and universities
- college and university organizations - for internal use only (not SAAS services offerings).

To apply please send an email with all the details to orders@psgsdk.com.

4.1 Licensing

PSG clients do not require any licence to be installed or used, however licenses are required server side.

Several types of different license packages are available:
- per server users number
- per server users number and time period (rented)
- enterprise (site licence)

Generally licenses are sold only by software developers in conjunction with their solutions. For the end user the PSG licence could be assimilated with the final application licence. PSGSDK.COM uses a business to business model for licensing policy and does not sell licenses directly to end user.

A license management solution is provided to PSGSDK.COM partners. This is used to register and create client licenses for final users based on any PSGSDK licence type.
The partnership package is free, and issued licenses costs are linked to the committed volume.

PSGSDK.COM partners may be software developers (companies or individuals).

All licenses are related to the hardware, therefore new licence file should be issued when changes in hardware occur. Any new licence request should be carefully evaluate to correspond to a reasonable time period. Hardware service or accounting documents could be requested in case. The licenses issuer is allowed to stop providing a new licence file when provided explanation is not confirmed by the documents. For in-house software development departments the same model will be used, a partner agreement being required with the company as the end user.

The software is provided in the actual configuration. PSGSDK.COM cannot be hold responsible and it will not be involved in any legal disputes related to the content resulted from running the PSG platform. Please check the partner agreement for more details.

Development tools are free and for PSG server to be free some conditions should be met.

PSG server it is offered for free in the following circumstances:
- for development (up to 3 users/server only*), with no regard to the number of developers or server instances, when no licence file required, no registration with PSGSDK.COM .
- non profit charity organizations
- state or government:
    - public hospitals that offer free services in a significant percent and any other similar establishments
    - public schools and universities
- college and university organizations - for internal use (not SAAS services offerings or other commercial purposes).
(all previous cases are connected only with PSG, therefore some fees may apply depending on the protocol with applications' owners as PSG is the development platform only).

* if more users are required for extended testings like server stress testings a temporary developer licence could be requested for free at support@psgsdk.com .