

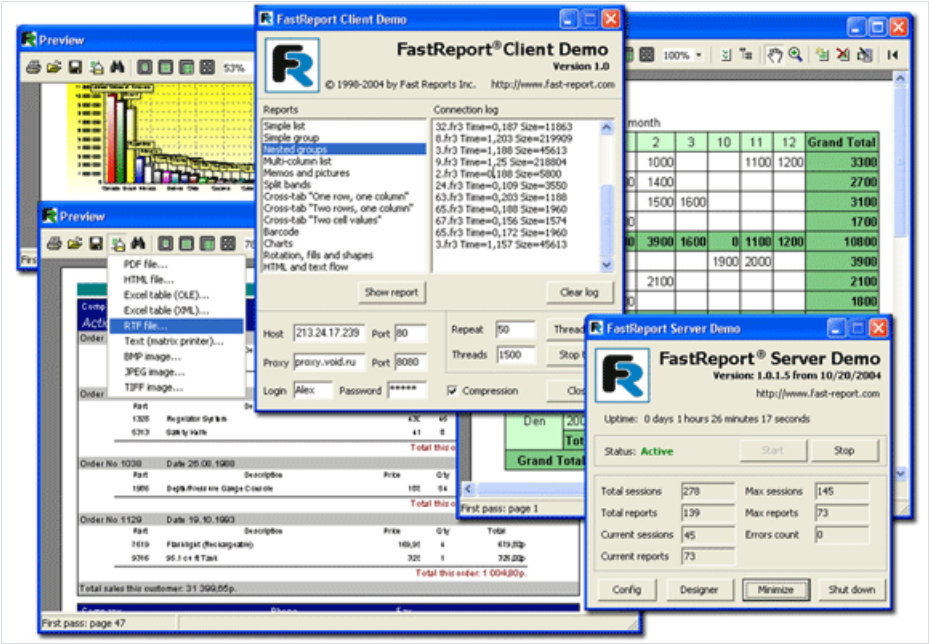
FastReport VCL Web Reporting Guide

Version 2026.2

© 1998-2026 Fast Reports Inc.

Introduction

This programming guide contains information about FastReport VCL library's extension. This extension allows to build reports on client-server technology with using of standard FastReport VCL components and additional components (that are intended for organization of interaction between client and server).



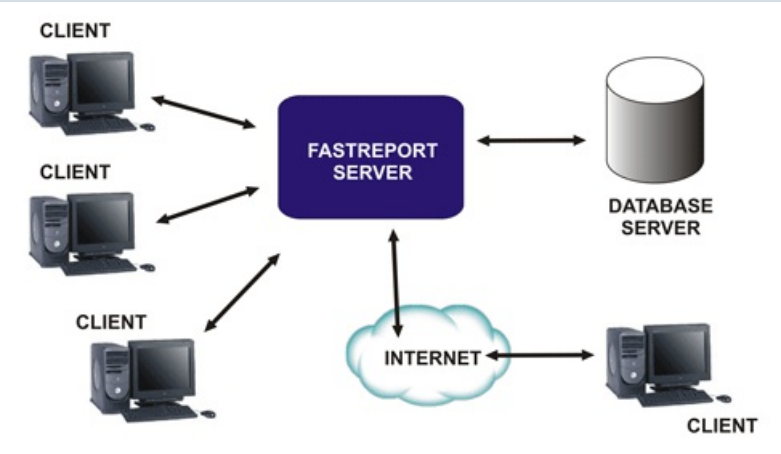
This guide describes the structure of client components and server components, their properties and methods, as well as architecture of a report server and the principles of its functioning. Furthermore, it gives recommendations concerning optimization and usage of new capabilities in already existing applications and those developed anew.

The experienced FastReport VCL users will be interested in recommendations about increasing server components' speed, optimization of reports for their correct export to various tabular formats, application of rules of information safety for application protection from non-authorized access.

We have been constantly improving the FastReport VCL Enterprise components. That is why there is a probability that some capabilities are not mentioned in this manual. Descriptions of all changes will be necessarily included to the next version of this manual.

FastReport VCL Enterprise - Client/Server reporting tool

The "client-server" technology is based on interaction between a client application (which inquires, analyzes, and displays requested information) and a server application that performs basic work related to various complex calculations.



There are several serious advantages of using client-server technology in your applications:

- low hardware requirements for client PCs;
- reducing of network traffic due to reducing the amount of information transferred between a client's application and a database server;
- simplicity of system management of the existing client-server;
- higher level of information protection.

However, the client-server technology has some considerable disadvantages:

- high hardware requirements for a PC used as a server;
- certain difficulties in development of client-server applications.

When developing FastReport VCL Enterprise, we take into account all major requirements for client-server applications. FastReport VCL Enterprise allows you to:

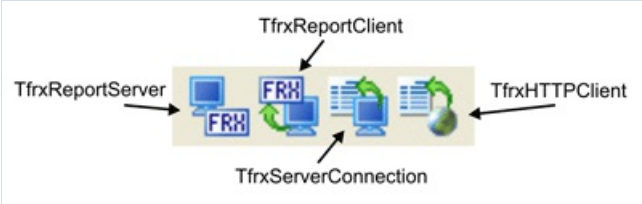
- run any reports on the server side on client request, without necessity to directly connect the client to the database server;
- manage several client requests simultaneously in separate threads; it minimizes response time of the server;
- since we use HyperText transfer protocol (HTTP, RFC 2068 [2]), you can use different existing applications, such as web-browsers (Edge, Mozilla Firefox, Safari, Opera etc), proxy-servers, web-servers (IIS, Apache, Nginx etc), together with FastReport VCL Enterprise without any additional requirements;
- use data compression algorithms (GZip, RFC 1952 [6]). This reduces network traffic and increases client-server processing power;
- use of MD5 algorithm for the MIC (Message Integrity Checksum, RFC 1321 [4], RFC 1864 [5]) increases data integrity;
- compatibility with FastReport VCL report files (with some restrictions) allows you to easily redesign your application to use client-server technology;
- standalone server application (without necessity to apply IIS, Apache or other web-server technologies) has a high processing power, short response time, and economical use of system resources (in comparison with solutions based on CGI technology);
- you can use the server as a simple HTTP server for storing and displaying any HTML documents;- application

of the Server Side Include (SSI) technology allows you to use the server as an engine for your web-site;

- managing the connection logs, error logs, and/or any additional system information allows you to keep record of the work, quickly track down the bugs and unauthorized access attempts;
- usage of authentications and "allow/deny" IP lists allows you to restrict access to the server;
- you can use several database connections in one report simultaneously;
- you can use FastReport client components for interaction between a client application and the server. You can use any web-browser as well;
- your reports may have a dialogue forms that will be used for entering some values before running a report;
- supported formats of the prepared reports are: HTML, PDF, RTF, XML, XLS, JPEG, and Text;
- you can use several modes of displaying the prepared report in your web-browser: single-page document, page-separated with page navigator.

Components of FastReport Enterprise Edition

After you install the packages of FastReport VCL Enterprise, a bookmark "FastReport Client/Server" in component palette IDE Delphi/C++Builder will be available.



Components of the "FastReport VCL Client/Server":

- TfrxReportServer - a server component, report server and HTTP server in a single whole;
- TfrxServerConnection - a client component, which contains information for connections with TfrxReportServer;
- TfrxReportClient - client component (a TfrxReport analogue) inquires the report on the server, and then displays the resulting report on client side;
- TfrxHTTPClient - client component, intended for requests of the files over HTTP protocol.

TfrxReportServer



TfrxReportServer component plays the role of a report server and HTTP server. This component does not require any additional components.

TfrxReportServer class inherited from TComponent

Properties

Name	Type	Description
Active	Boolean	the value, which indicates activity of the server. It may be used for starting the server by setting a value in "True"
Configuration	TfrxServerConfig	server configuration (TfrxServerConfig class is described below). Configuration changes become active only after you restart the server
AllowIP	TStrings	list of authorized IP addresses. The format of the list is as following: one line contains one IP address. In cases when the server does not find a client's address in this list, the client will be forbidden to connect; if the list is empty, all addresses are allowed to connect
DenyIP	TStrings	list of IP addresses forbidden to connect. The format of the list is as following: one line contains one IP address. In cases when the server does not find a client's address in this list, the client will be forbidden to connect; if the list is empty, all addresses are allowed to connect
PrintPDF	Boolean	pressing the "print" button in navigator's control panel (when viewing the resulting pages in a web browser) creates a PDF file, if this value is set to True. Otherwise, standard print action of the browser will be executed. Default setting is "True"

The following properties are inaccessible in object's inspector, but it is possible to access them from the code:

Name	Type	Description
Statistic	TfrxServerStatistic	server statistics (TfrxServerStatistic is described below)
Totals	TStrings	readable form of server statistics information
Variables	TfrxServerVariables	internal server variables (TfrxServerVariables is described below)

Methods

Method	Description
constructor Create(AOwner: TComponent)	creation of an object

Method	Description
procedure Open	startup of the server. At this moment all changes of configuration would be activated
procedure Close	server shutdown

Event handlers:

OnGetReport: TfrxServerGetReportEvent - may be used for loading the reports from any places (BLOB fields, files from any folders etc).

Type of the handler:

```
TfrxServerGetReportEvent = procedure (ReportName: String; Report: TfrxReport) of object;
```

ReportName - name of the requested report; it may be used for identification of a specific report;

Report - an instance of the TfrxReport object, to which the report should be loaded.

OnGetVariables: TfrxServerGetVariablesEvent - can be used for manual processing of the parameters received from the client, as well as execution of any operations directly on a server.

```
TfrxServerGetVariablesEvent = procedure(const ReportName: String; Variables: TfrxVariables) of object;
```

ReportName - The name of the report transferred in query. It can be used for filtering one or another parameter directly in the handler;

Variables - The list of the parameters received from the client.

TfrxServerConfig class inherited from TPersistent

Object of this class contains information about server configuration.

Properties

Name	Type	Description
Port	Integer	TCP/IP port number for client connection, default value is 80
IndexFileName	String	default filename, if the filename field in HTTP query is empty. Default value is 'index.html'
SessionTimeOut	Integer	time of storing report results on the server (in seconds). Default value is 300. As soon as the default time expires, the report results will be deleted. It is set depending on specificities of created reports and methods of client-server interaction; the waiting time of client activity after his connection in seconds, the default value is 60, after this time the client session will be deleted
SocketTimeOut	Integer	timeout of waiting for client's response (in seconds). Default value is 60. When time expires, the session will be terminated

Name	Type	Description
Logging	Boolean	log writing, "True" - enabled, "False" - disabled, "True" is set by default
LogPath	String	path to folder with logs; current folder by default
ReportPath	String	path to folder with reports; current folder by default
RootPath	String	path to folder with HTML files and reports results
Login	String	user name for authentication. If line is empty - authentication is not required. Empty line is a default setting
Password	String	password for authentication, empty line by default
Compression	Boolean	compression of transferred documents, client support required; "True" by default
MIC	Boolean	Message Integrity Checksum using MD5 algorithm. "True" by default
NoCacheHeader	Boolean	document is not cached by client, "True" by default
OutputFormats	TfrxServerOutputFormats	supported formats for requested reports, one or more from set (sfHTM, sfXML, sfXLS, sfRTF, sfTXT, sfPDF, sfJPG, sfFRP). By default, all elements of set are included
ReportCaching	Boolean	enable the reports cache on a server
ReportCachePath	String	path to a folder with reports cache
DefaultCacheLatency	Integer	reports in cache default storage time (in seconds)

Methods

Method	Description
procedure LoadFromFile(const FileName: String)	loads configuration from a file
procedure SaveToFile(const FileName: String)	saves configuration to a file

TfrxServerStatistic class inherited from TPersistent

Properties

Name	Type	Description
CurrentReportsCount	Integer	number of reports currently build
CurrentSessionsCount	Integer	number of sessions currently connected

Name	Type	Description
MaxReportsCount	Integer	maximum number of reports simultaneously built
MaxSessionsCount	Integer	maximum number of sessions simultaneously connected
TotalErrors	Integer	number of errors
TotalReportsCount	Integer	number of reports
TotalSessionsCount	Integer	number of sessions
UpTimeDays	Integer	Uptime days
UpTimeHours	Integer	Uptime hours
UpTimeMins	Integer	Uptime minutes
UpTimeSecs	Integer	Uptime seconds

TfrxServerVariables class inherited from TCollection

Contains server variables.

Used (reserved) names of the variables will describe in the part 3.4.

Methods

Method	Description
function GetValue(const Name: String): String	returns value of the variable with Name
procedure AddVariable(const Name: String; const Value: String)	adds a variable with Name and Value

TfrxServerConnection



TfrxServerConnection - client component keeps information for connection to report server TfrxReportServer. Object of this class is required for working of one or several TfrxReportClient components.

TfrxServerConnection class inherited from TComponent

Properties

Name	Type	Description
Host	String	server host name or server IP address, by default - 127.0.0.1
Port	Integer	server port; "80" by default
ProxyHost	String	HTTP-proxy name or IP address, blank by default
ProxyPort	Integer	HTTP-proxy port; "8080" by default
Login	String	username for authentication
Password	String	user password for authentication
Timeout	Integer	idle time (in seconds); "120" by default
RetryCount	Integer	retry count; "3" by default
RetryTimeout	Integer	delay between retries in seconds, "3" by default
Compression	Boolean	accept compressed files; "True" by default
MIC	Boolean	checking of the message integrity checksum; "True" by default

TfrxReportClient



TfrxReportClient is a client component for query. It receives and shows the reports from the server.

Required component: TfrxServerConnection.

TfrxReportClient is analogy of TfrxReport in previous versions of the applications based on traditional architecture.

TfrxReportClient class inherited from TfrxReport

Properties

Name	Type	Description
Connection	TfrxServerConnection	link to object of TfrxServerConnection
ReportName	String	name of the requested report, use method LoadFromFile for setting this property (see below)
Variables	TfrxVariables	contain report variables; can be used for variables transfer from client to server
Errors	TStrings	errors list

Methods

Method	Description
procedure LoadFromFile(FileName: String)	set the name of the requested report to property ReportName
function PrepareReport: Boolean	performs connection to the report server, requests a report, transfers report variables to server, and downloads a report result, which then is put to the "PreviewPages" Property. Result of the function is "True" if the task is successfully accomplished, otherwise it becomes "False"
procedure ShowPreparedReport	previews the received report
procedure ShowReport	requests and previews the report

TfrxHTTPClient



TfrxHTTPClient - client component for receiving any file via HTTP protocol.

TfrxHTTPClient class inherited from TComponet

Properties

Name	Type	Description
Active	Boolean	executes a query if "True" is set
Host	String	host name or host IP address; "127.0.0.1" by default
Port	Integer	host port, "80" by default
ProxyHost	String	HTTP-proxy name or proxy IP address
ProxyPort	Integer	proxy port
RetryCount	Integer	retry count, default - 3
RetryTimeOut	Integer	delay between retry in seconds, default - 5
TimeOut	Integer	idle time in seconds, default - 30
ClientFields	TfrxHTTPClientFields	fields the request's header, TfrxHTTPClientFields is described below
ServerFields	TfrxHTTPServerFields	parsed answer header, TfrxHTTPServerFields is described below
MIC	Boolean	checking the message's integrity checksum, "True" by default
Header	TStrings	raw request header; it is filled in automatically from ClientFields
Answer	TStrings	raw answer header, parsed fields will be stored in ServerFields
Stream	TMemoryStream	data received from server
Breaked	Boolean	sign of emergency disconnection
Errors	TStrings	errors list

Methods

Method	Description
procedure Connect	connect to remote server and get the file, after disconnect

Method	Description
procedure Disconnect	disconnect from server
procedure Open	same as "Connect"
procedure Close	same as "Disconnect"

TfrxHTTPClientFields class inherited from TPersistent

Properties

Name	Type	Description
AcceptEncoding	String	accepted compression formats, default -'gzip'
FileName	String	requested filename
Host	String	address or client's hostname; fills automatically if empty
HTTPVer	String	http protocol version, default - 'HTTP/1.1'
Login	String	user name for authentication
Password	String	password for authentication
QueryType	TfrxHTTPQueryType	query type, qtGet - GET query, qtPost - POST query; "qtGet" by default
Referer	String	referencing document name; blank by default
UserAgent	String	client program name, default - 'FastReport/3.0'

TfrxHTTPServerFields class inherited from TPersistent

Properties

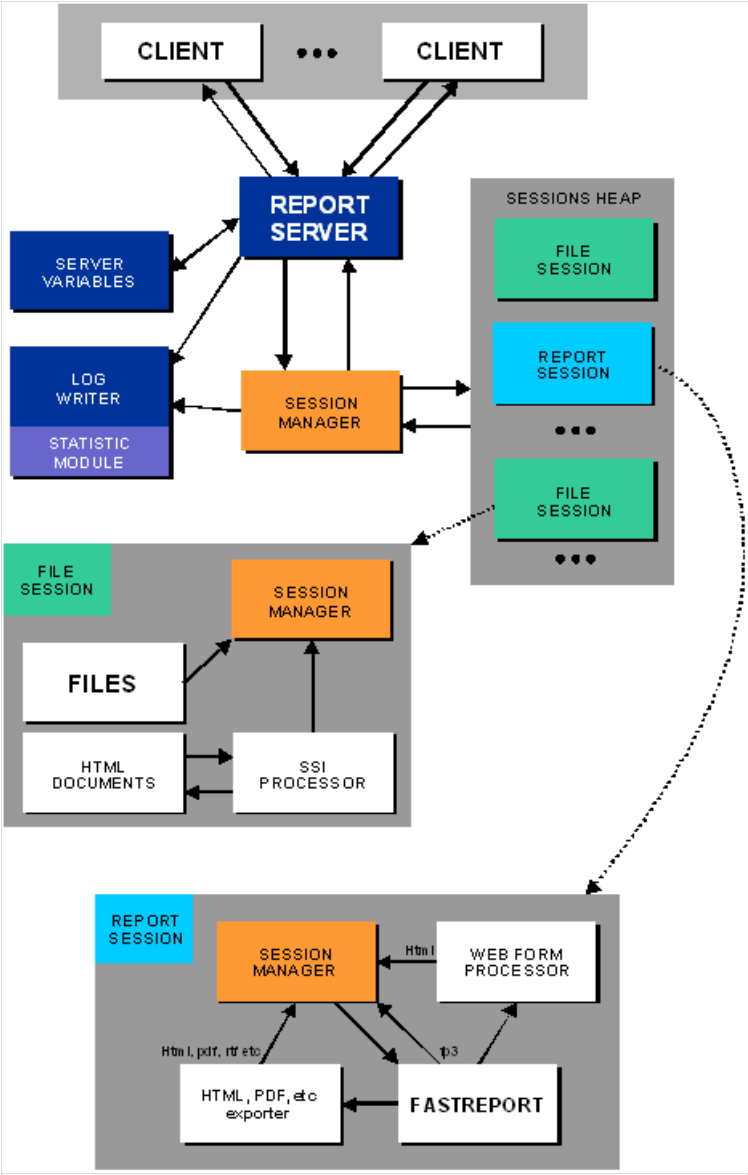
Name	Type	Description
AnswerCode	Integer	server response code
ContentEncoding	String	received data compression format
ContentMD5	String	MD5 checksum
ContentLength	Integer	received data length
Location	String	actual location of the document

The Report server

Server side (TfrxReportServer component) is represented as an autonomous HTTP server with a capability of report generating. The Report server is able to transact several reports simultaneously, logging any events, and collecting the statistics.

Internal architecture

The scheme displays the server's internal structure:



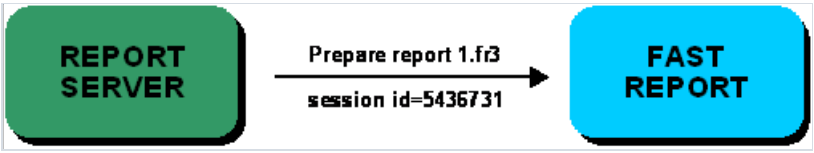
The sessions with unique identifier are created when a request from client comes. The line of the request is analyzed. If the requested file exists, then the server sends a positive response with the file to the client. Logs are updated with new record about this event. If the request contains the report query, then a special report session is created. After the report is built, the result is saved to folder with session number as a name. The server responses to the client, and reports a new file location. The client sends a new request to the new file location, and receives the file with the result. Session with the resulting file is stored by server until session time expires.

Below is a step-by-step graphical overview of the report query transaction with the web browser:

- client sends query; the report's title is "1.fr3"



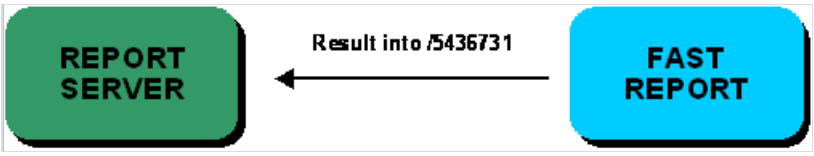
- the server creates a new FastReport instance and delivers parameters of the request



- FastReport prepares the report and exports results to a html file into the folder, the name of the folder is the same as the session's number



- server waits for the results from FastReport



- client receives redirection to the location of the resulting file



- client sends a new query with the request of results file



- server delivers the resulting file to the client



Step-by-step graphical overview of the report query transaction with the FastReport (TfrxReportClient):

- a client wants to show report "1.fr3":



- client component sends a query with the name of report "1.fr3" (native result format)



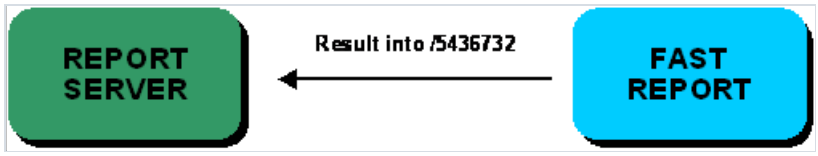
...

- FastReport prepares the report and saves the results to a native fp3; the name of the folder is the same as the

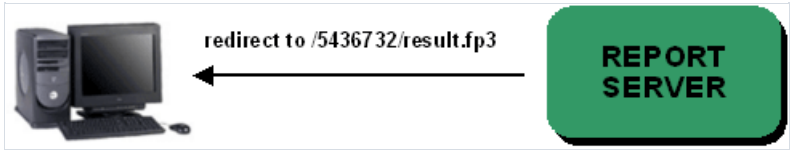
session's number



- server waits for the results from FastReport

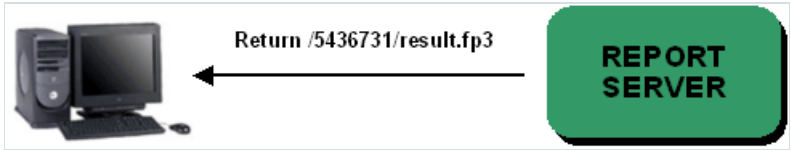


- client receives redirection to the location of the resulting file



...

- server sends the result file to the client

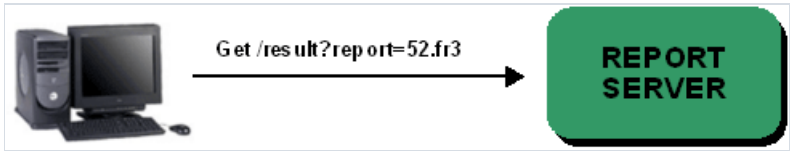


- client displays the report

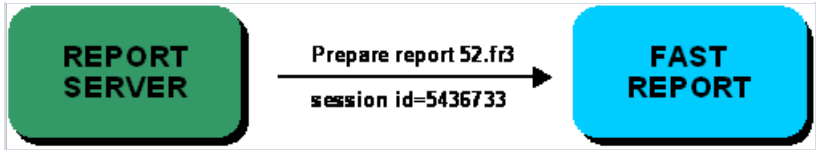


If the inquired report contains any forms, the process becomes more complicated:

- client client component sends a query with the name of report "1.fr3"



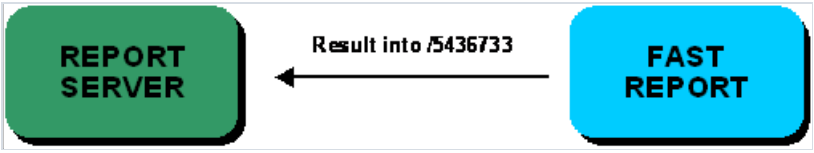
- the server creates a new FastReport instance and transfers parameters of the request



- FastReport prepares report and saves the web-form into the folder name according to the session number



- server wait the results from FastReport



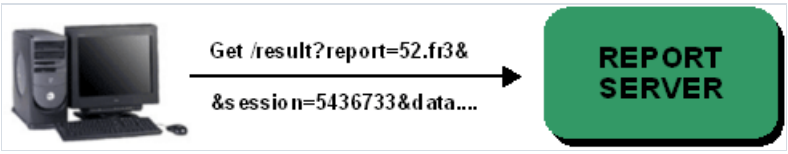
- server redirects the client to the web-form file



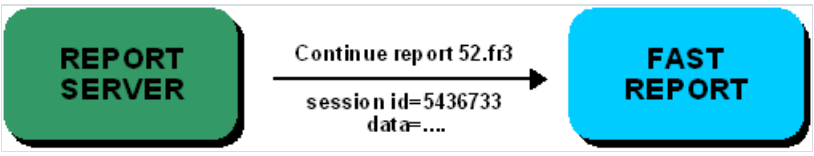
- client receives the web-form, while FastReport waits



- client sends of the web-form dialog controls states to the server



- the server transfers the values of the control elements to the server



- server delivers the received information to FastReport



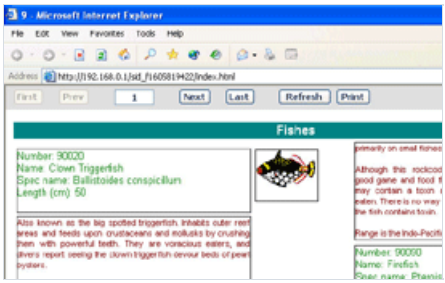
Format of the server request line, logging, authentication and other issues concerning server's functioning, are described below.

Supported formats of the report results

FR3 is a native FastReport VCL format. It is represented as a XML document. FR3 is used during transaction between TfrxReportServer and TfrxReportClient. This format is the most appropriate one for document printing. In most cases, use of this format reduces both transaction time and size of the transferred files (except reports containing high quality images).

It is undesirable to use additional compression components (TfrxGZipCompressor) on server side, since it reduces overall server performance, especially when the traffic compression option is activated (TfrxReportServer.Configuration.Compression := True; TfrxReportClient.Connection.Compression := True).

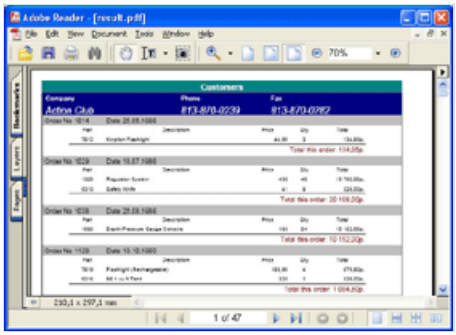
HTML format used by most web-sites in the Internet network is intended for previewing document in low resolution. It is quite difficult to perform high-quality printing of document using this format. HTML format is convenient for most web-browsers. If you use a web-browser as a client, then this format is appropriate for you (see details in topic 5.2). FastReport server creates web pages with a report navigator, with the help of which you can scroll the pages.



PDF format by Adobe is designed specially for documents intended for printing.

FastReport makes high-quality export to this format. For viewing and printing PDF documents, you should install Adobe Acrobat Reader program on your computer.

If property "TfrxReportServer.PrintPDF := True" is set, then, during previewing HTML pages with report results, a file in this format is generated (by pressing the "Print" button on report navigator panel).



The Server also supports the following formats:

- RTF format. A RichText document can be opened in most text processors;
- XLS and XML. These are the Excel spreadsheets formats;
- text file (required for dot-matrix printing);
- graphic file jpeg.

The set of the formats allowed to use in queries is configured by the "TfrxReportServer.Configuration.OutputFormats" property, which may contain one or several values from the following set: sfHTM - HTML format, sfXML - XML format, sfXLS - Excel format, sfRTF - RichText format, sfTXT - text

file, sfPDF - Adobe Acrobat format, sfJPG - jpeg picture, sfFRP - native FastReport 3 (FR3) format.

If type of a returned format is not specified during request, then the server generates the result in HTML format.

Query syntax

When using an ordinary web-browser as a client, you can use parameters of the query line:

report=*name*

name - name of the report available on server

Example: */report=1.fr3* (query report 1.fr3, resulting format - HTML).

format=*name*

name - format of the required file, available formats: **HTM** (HTML), **XML** (xml table), **XLS** (Excel table), **RTF** (rich-text document), **TXT** (text file), **PDF**(Adobe Acrobat file), **JPG** (jpeg image), **FRP** (internal FastReport prepared report format).

By default format is **HTM** (HTML).

Example: */report=1.fr3&format=TXT* (query report 1.fr3, resulting format - text file).

pagerange=*value*

value - result page range (for FRP this option is inaccessible).

Example of the page range: 1,3,5-12.

Example of the query line: */report=3.fr3&pagerange=20-25* (query report 3.fr3, pages from 20 to 25, resulting format - HTML).

multipage=param

Only for HTM format. If param value is "1", then the resulting report will be presented as several pages (one file on each page). If param set as "0", then a single resulting page will be generated that will contain all report pages. Default parameter value setting is "1".

Example: */report=3.fr3&multipage=0* (query report 3.fr3, resulting format - HTML, all pages on one HTML page).

pagenav=*param*

Only for HTM format. To enable page navigator, set param value as "1". If param value is "0", then page navigator is off. For correct page displaying, use web-browser with javascript and frames support. Default setting of this parameter is "1".

Example: */report=9.fr3&multipage=0&pagenav=0* (query report 9.fr3, resulting format - HTML, all result pages on one HTML page, page navigator is off).

Transferring parameters to the report

If other parameters are presented in the request line (not listed above), then server interprets them as parameters for building a report as internal FastReport variable.

Example:

`/report=myreport.fr3&param1=Hello%20World!` (query report "myreport.fr3", FastReport variable "param1" setting is "Hello World!")

Below are some restrictions concerning parameters transferred:

- all strings can be converted in Unicode UTF-8 format and can be compatible with HTTP query standard (use of standard function Untf8Encode and function HTMLCodeStr declared in frxServerUtils.pas file);
- all parameters are transferred to report as strings. Please keep in mind this when you use these parameters in the report script;
- all variables contained in TfrxReportClient.Variables are automatically sent to the server.

Internal server variables

During server's working, the "TfrxReportServer.Variables" property contains the following automatically created and updated variables:

`SERVER_NAME` - server name;

`SERVER_COPYRIGHT` - copyright;

`SERVER_SOFTWARE` - server version;

`SERVER_LAST_UPDATE` -last update date;

`SERVER_UPTIME` - up time of the server;

`SERVER_TOTAL_SESSIONS` - sum total of sessions;

`SERVER_TOTAL_REPORTS` - sum total of reports;

`SERVER_TOTAL_ERRORS` - sum total of errors;

`SERVER_MAX_SESSIONS` - maximal number of simultaneous sessions;

`SERVER_MAX_REPORTS` - maximal number of simultaneous report generations.

Example of getting a variable SERVER_TOTAL_REPORTS

```
Totals := frRepotServer1.Variables.GetValue('SERVER_TOTAL_REPORTS');
```

Using HTML documents

The Server can be used as a simple HTTP server for viewing any HTML documents or any other files.

Place the HTML documents to any folder, and then correctly set the property TfrxReportServer.Configuration.RootPath.

Name of the default document must be specified in the TfrxReportServer.Configuration.IndexFileName property (default index.html). Correspondingly, a document with this name must exist in the root folder.

SSI (Server Side Include) commands description.

Include any file in document.

```
<!--#include virtual="filename.html" -->
```

Include the file with name filename.html in current document position. Path to file is specified from RootPath.

Example:

```
<!--#include virtual="header.html" --> Command line help
<!--#include virtual="top.html" -->
<font face="Tahoma" size="3"><a href="index.html"><b>Back to main page</b></a><b><br>
</b></font><hr>
...
```

Insert value of server variable.

```
<!--#echo var="VARIABLE"-->
```

Insert the value of variable with the "VARIABLE" name in current document position.

Example:

```
...

<tr> <td align="right" width="200"><b>Uptime:</b></td>
<td width="300"><!--#echo var="SERVER_UPTIME"--></td></tr>
<tr> <td align="right"><b>Total sessions:</b></td>
<td><!--#echo var="SERVER_TOTAL_SESSIONS"--></td></tr>
<tr> <td align="right"><b>Total reports:</b></td>
<td><!--#echo var="SERVER_TOTAL_REPORTS"--></td></tr>
<tr> <td align="right"><b>Max sessions:</b></td>
<td><!--#echo var="SERVER_MAX_SESSIONS"--></td></tr>
<tr> <td align="right"><b>Max reports:</b></td>
<td><!--#echo var="SERVER_MAX_REPORTS"--></td></tr>
...
```

Use of SSI commands optimizes website development.

Example of the site with SSI you can see in the "\FastReport\Demos\ClientServer\Server\htdocs" folder.

Logs

If the TfrxReportServer.Configuration.Logging property setting is "True," then the server writes logs to folder described in the "TfrxReportServer.Configuration.LogPath" property.

The server supports 5 logs:\

- log of the accessed clients "access.log" - contains information about date, time, session id, IP and query line. Log fragment:

```
10/26/2004 23:56:19 sid_f1672494035    192.168.0.2  result?report=3.fr3
10/26/2004 23:56:23 sid_f1340767011    192.168.0.2  sid_f1672494035/index.html
10/26/2004 23:56:23 sid_f1949776310    192.168.0.2  sid_f1672494035/index.nav.html
10/26/2004 23:56:23 sid_f1150188690    192.168.0.2  sid_f1672494035/index.1.html
```

- log of the connected program type "agent.log", contains information about date, time, IP, and program name. Log fragment:

```
10/26/2004 23:56:19 192.168.0.2 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
10/26/2004 23:56:23 192.168.0.2 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
10/26/2004 23:56:23 192.168.0.2 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

- log of the referencing URLs "referer.log", contains information about date, time, IP and referencing URL. Log fragment:

```
10/26/2004 23:56:19 192.168.0.2  http://192.168.0.1/
10/26/2004 23:56:23 192.168.0.2  http://192.168.0.1/
10/26/2004 23:56:23 192.168.0.2  http://192.168.0.1/sid_f1672494035/index.html
```

- errors log "error.log", contains information about errors:

```
10/25/2004 13:30:52 192.168.0.2  588864044016/index.1.html document not found
10/26/2004 0:03:11  192.168.0.2  Software caused connection abort.(10053)
10/26/2004 0:43:42  192.168.0.2  Connection reset by peer.(10054)
```

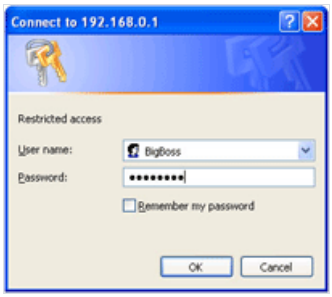
- server log "server.log", contains summary server information:

```
10/25/2004 19:38:15 Started
10/25/2004 19:38:15 HTTP server created
10/25/2004 19:58:57 HTTP server closed
10/25/2004 19:58:57 Stopped
Uptime: 0 days 0 hours 20 minutes 42 seconds
Total sessions: 654
Total reports: 327
Total errors: 0
Max sessions: 84
Max reports: 42
```

Do not forget to archive the log files.

Authentication

The server supports basic HTTP authentication. To activate authentication, set property TfrxReportServer.Configuration.Login and TfrxReportServer.Configuration.Password. If you set this properties, then request header must contain authentication info (RFC 2068 [2]). If client receives answer from server with 401 "Unauthorized" error coded, then the client must retry sending the query with correct authentication data. At that, web browser simply shows dialog window with login and password request:



Access restriction by IP address

The Server supports the restriction by client IP address.

Property TfrxReportServer.DenyIP can contain list of the restricted client IPs'.

Property TfrxReportServer.AllowIP can contain list of the allowed client IPs'.Each list must contain one IP address in one line.Here is an example of such list:

```
192.168.0.10
192.168.0.12
192.168.0.54
```

If the "DenyIP" and "AllowIP" lists are empty, then all clients are allowed to connect to the server.

If the "DenyIP" list is empty, while the "AllowIP" list contains an IP address, then only one client with this IP address can connect to the server.

If the IP address of a connected client is not included in the "DenyIP" list, then the server checks if this address is included in the "AllowIP" list.

IP addresses' masks are not supported.

Examples:

1. Only local host can connect to the server:

AllowIP:

```
127.0.0.1
```

DenyIP is empty.

2. IP addresses 192.168.0.2 - 192.168.0.6 can connect to the server.

AllowIP:

```
192.168.0.2
192.168.0.3
192.168.0.4
192.168.0.5
192.168.0.6
```

DenyIP is empty.

2. IP addresses from range 192.168.0.8 - 192.168.0.10 cannot be connected to the server.

AllowP is empty.

DenyIP:

```
192.168.0.8
192.168.0.9
192.168.0.10
```

Database connections

Most of reports use data from databases. To connect to a database, you should:

- specify the database connection component (for example, TADOConnection) in one of your application forms;
- use internal data access components (such as TfrxADOTable, TfrxADOQuery) in your report. To connect to the database, these components should use the application connection.

In this case, a report will be thread-safe, using single connection to the database. In case some data access components do not support simultaneous work with database through a single connection, you should use the way described below.

The other way is to use the database connection component (such as TfrxADODatabase) in each report. In this case, you will be able to connect to different databases at one time. We do not recommend thisway if you do not need this functionality because each time when report starts, it will attempt to connect to the database (in some DB servers the connection may take a long time).

Read the "FastReport 3 - User manual" [9] to learn more about creating reports with internal data access components (page 134).

It is not recommended to use BDE to connect to a database. BDE has a great amount of problems when working with several threads.

Using the reports cache

Caching of reports allows achieving high efficiency because of saving prepared reports in temporary files of the server. Depending on server configuration, after preparation the result can be placed in cache.

After specified time, the result of the report will be removed from the cache.

If during this time a query with the *same name of the report and the same values of parameters* is received from a client, the response will be immediately returned to it. The reply will be based on the result saved in cache, and will be represented in the format requested by the client.

In that case, the server will waste time only on conversion of the prepared report in the requested format without building a report. It considerably increases the productivity.

Depending on tasks performed by a server, it is possible to assign an individual storage time in cache for each particular report.

Time value is set by the administrator of a server, according to actuality of a report, after certain period of time.

For example, the annual report about activity of an enterprise can be stored in cache long enough, since the information will be relevan for a long period of time, and it would not become outdated very soon. On the contrary, a report about a large commercial organization warehouse would be relevant during a small period, and therefore it consequently should be stored in cache not too long.

Reports cache properties:

`TfrxServer.Configuration.ReportCaching` - enable the cache (True/False);

`TfrxServer.Configuration.ReportCachePath` - path to the cache folder;

`TfrxServer.Configuration.DefaultCacheLatency` - latency timeout, default setting is 300 seconds.

Server configurations file parameters:

```
[ReportsCache]
; enable caching of the reports with same params
Enabled=1
; path to chache folder
CachePath=.\cache\
; dafault delay for cache of the report results in seconds
DefaultLatency=300
```

The additional section of a configuration file of a server [ReportsLatency] is for customization of a storage time in cache results of one or another report:

```
[ReportsLatency]
; cache delay for the 1.fr3 report in seconds
1.fr3=10
; cache delay for the 2.fr3 report in seconds
2.fr3=20
; add below the any reports for the custom cache delay setup
```

Correction of parameters configuration will minimize time of working clients and will reduce total traffic on the

server.

Increasing server's processing power

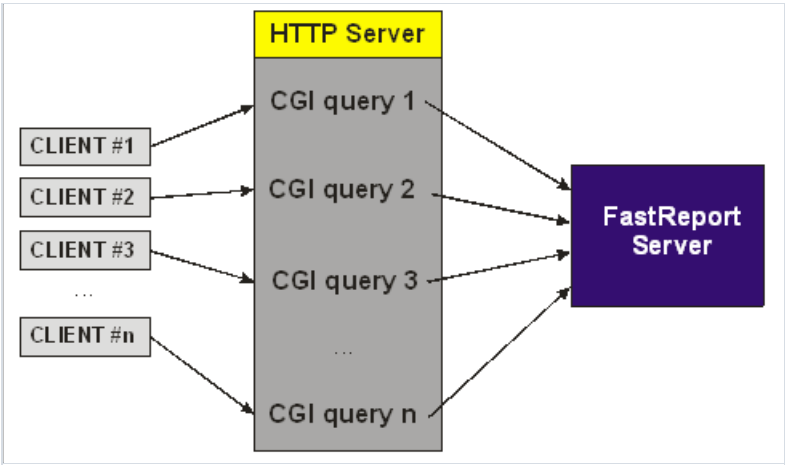
Use the following recommendations to increase the report server performance:

- do not use compression component (TfrxGZipCompressor). It considerably slows down the server;
- optimize your SQL queries. In some cases running the SQL query may take a longer time than the report execution;
- do not use high-resolution bitmaps in your reports - it will increase the report execution time and network traffic;
- do not use complex scripts in your reports;
- use TfrxReportClient component in your client application. It works with FP3 native format and allows reducing the server response time (server does not perform the export to HTML or other formats) and the network traffic.
- when developing a report, keep in mind recommendations from [4.2](../Developing%20the%20reports/Some%20advices%20concerning%20the%20design%20of%20a%20report.md);
- turn off the checksum, TfrxReportServer.Configuration.MIC := False;
- increase the memory size, use the faster CPU on PC used as a report server.

Using the FastReport server together with other HTTP servers (Apache, IIS, etc)

To use already existing solutions based on other HTTP servers, their integration with the FastReport server is possible by means of the "CGI" mechanism. It gives an advantage in comparison with using a built-in HTTP server FastReport. Reports can be built in an already-working system (site). HTTP server and a server of reports can work on different computers. Usage "SSL" encoding for operation with HTTP a server is possible (this possibility is unavailable in HTTP server FastReport yet).

Applying such method, CGI becomes an intermediate for tranfering a query to the "FastReport" server, obtaining results from a server of reports, and return of the results to the client.



You can found example of CGI wrapper in the "Demos\ClientServer\CGI" folder.

To us the CGI wrapper:

- compile and copy file fastreport.exe to the folder /cgi-bin of the HTTP server;
- configure the HTTP server (Apache, IIS or other) to execute the CGI application. Read more about this in HTTP server user manual;

If HTTP and reports servers work on same computer:

- if TCP/IP port 80 is used by HTTP server configure the FastReport server on other port 8097 (this port is used by CGI application by default if configuration file is missed), if you want to use other TCP/IP port, read below about using the configuration file of the CGI application;

If HTTP and FastReport servers work on separate computers:

- create the configuration file of the CGI application in folder /cgi-bin with name fastreport.ini:

```
[REPORTSERVER]
; IP address of the FastReport server
Host=192.168.0.34
; IP port of the FastReport server
Port=80
```

- launch the FastReport server and check work of the CGI application.

Report query example with using of CGI application:<http://127.0.0.1/cgi-bin/fastreport.exe?

report=67.fr3&multipage=0&pagenav=0>

Read more about query line syntax in 3.3 topic. Replace the "result" keyword in this point at "cgi-bin/fastreport.exe" construction.

Attention: to restrict direct access to the report server from clients, it is necessary to specify an IP address of the HTTP server, on which CGI application works (127.0.0.1 or other).

Developing the reports

Developing of the FastReport reports was described in the "FastReport VCL - user manual."

Some client/server restrictions

When developing reports for client-server application, please remember that:

- you cannot use script event handlers for dialogue forms' controls since dialogue forms are displayed as web forms in the browser;
- you cannot use event handlers of the TfrxReportClient component (for example, OnGetValue, OnUserFunction). All such handlers should be on the server side;
- you cannot use common data access components, such as TfrxDBDataSet (common components cannot be simultaneously used by several reports). Each report should have internal data access components, such as TfrxIBXTable, Query, and so on.

Some advices concerning the design of a report

Many of the document formats use table-style data representation. For representing of resulting reports, the server uses such formats as HTML, XLS, and RTF.

Table-style documents cannot have intersected cells, while FastReport document can. FastReport uses free-form data layout - there is no "lines", "table cells" like in Word, Excel or other such formats. FastReport export filters for table-style formats (RTF, HTML, and XLS) uses special algorithm to convert intersected cells into table cells and optimally arranges them. In places where FastReport objects intersect with each other, export filter may generate additional table rows and columns. It is necessary for better WYSIWYG, but may result in increased number of rows and columns in a resulting layout, which makes the table layout unusable for further analysis and slows down the export process.

Keep in mind these export limitations when developing a report, if you intend to export your report into such table-style formats. To avoid the objects' intersection, use alignment tools of the FastReport designer. Turn on the "grid align" option.

When creating tables in a report, put the table cells side-by-side, if possible, and avoid cells' intersection. If cells are intersected, the export algorithm would make clipping, and the export result may differ from the original report.

If possible, place objects along the horizontal and vertical guide lines. Use designer's guide lines to do this.

Following these instructions would help your reports to look perfect during exporting to any of the supported formats.

Report client

There are two kinds of clients of the FastReport server:

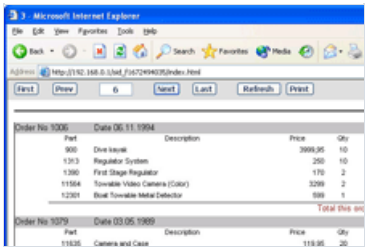
- applications that use TfrxReportClient component;
- any stand-alone HTTP-clients, such as web-browsers.

TfrxReportClient-based report client

TfrxReportClient component is designed specially for client applications. This component allows querying a report from the server, passing some report parameters (variables) to the server. It receives prepared report in the FP3 format (native FastReport format). The prepared report can be displayed and printed on the client side. You can also export the prepared report to any of the supported formats, using export filter components. In most cases, this solution is optimal for client applications. Clients that use the TfrxReportClient component make low network traffic and use less server system resources.

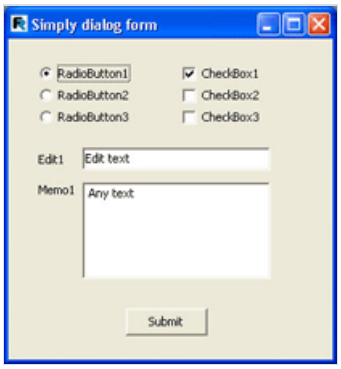
Other clients

The FastReport server gives you wide opportunities of choosing the client program due to using standard HTTP protocol. You can use any HTTP-compatible client such as web-browser that supports JavaScript, tables, and frames.

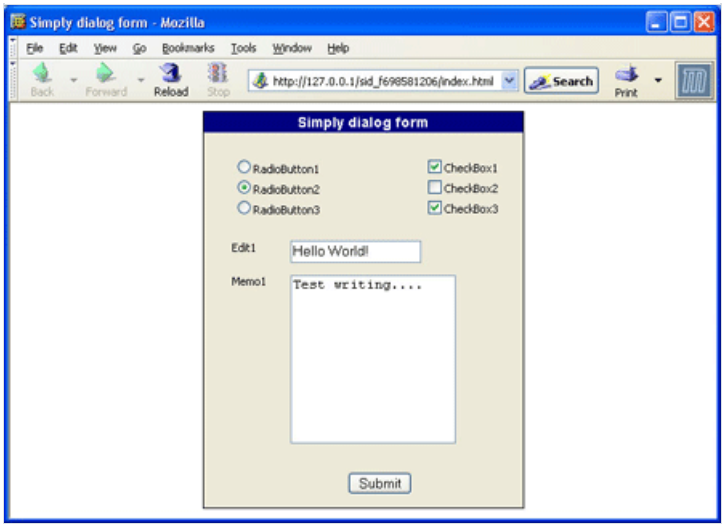


When using dialogue forms in your reports, the server will convert them to web-forms and pass them to a client. Client should fill in the form and return it to the server.

This is how the dialogue form looks when running a report in a simple (non-client-server) application:



The same form appears in the Mozilla web-browser, when running a report in the client-server application.



Adapting your applications for client-server technology

When adapting previously developed applications to the client-server technology, use the following recommendations:

- Clearly define the interaction between client and server sides;
- Take into account the recommendations from topics 4.1 and 4.2 of this manual;
- When working with databases, take into account the recommendations from topic 3.10 of this manual;
- To improve the level information security, take into account the recommendations from topic 8 of this manual.

Example of a simple client-server application

For familiarization with methods of using component FastReport Enterprise, see demonstration examples stored in the "\FastReport\Demos\ClientServer" folder.

Server side

You can find all source files of this example in the \FastReport\Demos\ClientServer\Server folder.

Components used in this demo: server component TfrxReportServer (Serv), database connection component TADOConnection and TfrxADOComponents, along with other add-on FastReport components.



For the convenience of clients, data about configuration of the server is stored in a file, which is editable by the built-in editor.

File server.conf:

```
[Server]
; TCP/IP port for HTTP server
Port=80
; report session timeout in seconds
SessionTimeOut=600
; client connection timeout in seconds
SocketTimeOut=600
; index page filename
IndexFileName=index.html
; path to folder with logs
LogPath=.\logs\
; enable of log writing
WriteLogs=1
; maximum log files in history
MaxLogFles=5
; maximum log file size
MaxLogSize=1024
; path to folder with the reports (*.fr3)
ReportPath=.\reports\
; public document folder for documents and results
RootPath=.\htdocs\
; disable of the caching document by the web browser
NoCacheHeader=1
; GZIP compression enable
Compression=1
; MD5 message integrity check
MIC=1
; user login
Login=
; user password
Password=

[ReportsCache]
; enable caching of the reports with same params
Enabled=1
; path to chache folder
CachePath=.\cache\
; dafault delay for cache of the report results in seconds
DefaultLatency=300

[ReportsLatency]
; cache delay for the 1.fr3 report in seconds
1.fr3=10
; cache delay for the 1.fr3 report in seconds
2.fr3=20
; add below the any reports for the custom cache delay setup
```

Fields of the configuration file correspond to field names of the TfrxReportServer.Configuration property.

The "allow.conf" and "deny.conf" files contain lines with allowed and restricted addresses respectively.

Database file is stored in the "\database" folder.

In the main module, the constants with names of configuration files are defined:

```
const
  CONFIG_FILE = 'server.conf';
  ALLOW_FILE = 'allow.conf';
  DENY_FILE = 'deny.conf';
```

After program starts, the database is connected via the MicrosoftJet OLE DB interface.

In variables ConfFile, AllowFile, DenyFile we a store path to configuration files:

```
AppPath := ExtractFilePath(Application.ExeName);
ConfFile := AppPath + CONFIG_FILE;
AllowFile := AppPath + ALLOW_FILE;
DenyFile := AppPath + DENY_FILE;
```

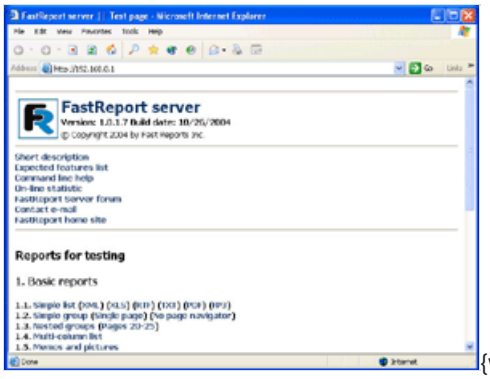
Load config files to the Serv component:

```
Serv.Configuration.LoadFromFile(ConfFile);
Serv.AllowIP.LoadFromFile(AllowFile);
Serv.DenyIP.LoadFromFile(DenyFile);
```

Execute the server:

```
Serv.Open;
```

After all work is done, you are ready to use a powerful report server. Launch the any web browser and type http://127.0.0.1 in address line



{width="296" height="227"}

You can design reports with the help of the internal FastReport designer:

```
OpenDialog1.InitialDir := Serv.Configuration.ReportPath;
if OpenDialog1.Execute then
begin
  frReport1 := TfrxReport.Create(nil);
  frReport1.LoadFromFile(OpenDialog1.FileName);
  frReport1.DesignReport;
  frReport1.Free;
end;
```

Client side

You can find all source files of this example in the \FastReport\Demos\ClientServer\Client\Simple folder.

This is an example of using the TfrxReportClient component and transferring report variables to the server.

Before starting this program, launch the server application described above (topic 7.1.1.)



Press the "Show Report" button and type "1.fr3" in the "Report Name" field when running this example so that the program would execute the code below:

```
frxServerConnection1.Host := Host.Text;
frxServerConnection1.Port := StrToInt(Port.Text);
frxServerConnection1.Login := Login.Text;
frxServerConnection1.Password := Password.Text;
frxReportClient1.LoadFromFile(RepName.Text);

if Length(Param1Value.Text) \> 0 then
  with frxReportClient1.Variables.Add do
  begin
    Name := Param1.Text;
    Value := Param1Value.Text;
  end;

if Length(Param2Value.Text) \> 0 then
  with frxReportClient1.Variables.Add do
  begin
    Name := Param2.Text;
    Value := Param2Value.Text;
  end;

if frxReportClient1.PrepareReport then
  frxReportClient1.ShowPreparedReport;

Memo1.Lines.AddStrings(frxReportClient1.Errors);
```

After successful report request, you see the preview of the result.

Client side with threads

You can find all source files of this example in the "FastReport\Demos\ClientServer\Client\Advanced" folder.

This example shows how you can use the "TfrxReportClient" component in the threads.



Thread class:

```
TfrxClientTestThread = class (TThread)
  protected
    procedure Execute; override;
  private
    CountRep: Integer;
    ErrorsCount: Integer;
    Log: TMemo;
    ThreadID: Integer;
    procedure AppendLog;
    procedure FinishLog;
  public
    Report: TfrxReportClient;
    constructor Create(C: TfrxServerConnection; RepName: String; Id: Integer; Rep: Integer; L: TMemo);
    destructor Destroy; override;
end;
```

Constructor of the TfrxClientTestThread class:

```
constructor TfrxClientTestThread.Create(C: TfrxServerConnection; RepName: String; Id: Integer; Rep:
Integer; L: TMemo);
begin
  inherited Create(True);
  FreeOnTerminate := False;
  ErrorsCount := 0;
  ThreadId := Id;
  CountRep := Rep;
  Log := L;
  Report := TfrxReportClient.Create(nil);
  Report.EngineOptions.ReportThread := Self;
  Report.Connection := C;
  Report.ReportName := RepName;
  Resume;
end;
```

The method `TfrxClientTestThread.Execute` sends a request to the CountRep server. All resulting information is displayed in Memo1 by the "AppendLog" and "FinishLog" methods:

```
procedure TfrxClientTestThread.Execute;
var
  i: Integer;
begin
  inherited;
  for i := 1 to CountRep do
  begin
    if Terminated then break;
    Report.PrepareReport;
    if not Terminated then
    begin
      Synchronize(AppendLog);
      ErrorsCount := ErrorsCount + Report.Errors.Count;
    end;
  end;
  Synchronize(FinishLog);
end;
```

Before starting this program, launch the server application described above.

On press button "Thread test" execute the code below:

```
procedure TMainForm.TestBtnClick(Sender: TObject);
var
  i, j, k: Integer;
  Thread: TfrxClientTestThread;
begin
  frxServerConnection1.Host := Host.Text;
  frxServerConnection1.Port := StrToInt(Port.Text);
  frxServerConnection1.Login := Login.Text;
  frxServerConnection1.Password := Password.Text;
  frxServerConnection1.Compression := Compression.Checked;
  if (Length(ProxyHost.Text) > 0) then
  begin
    frxServerConnection1.PrxoyHost := ProxyHost.Text;
    frxServerConnection1.ProxyPort := StrToInt(ProxyPort.Text);
  end;
  ClearThreads;
  Memo1.Lines.Add('Start test');
  j := StrToInt(Threads.Text);
  k := StrToInt(Rep.Text);
  for i := 1 to j do
  begin
    Thread := TfrxClientTestThread.Create(frxServerConnection1, ReportsList[ListBox1.ItemIndex], i, k,
Memo1);
    ThreadList.Add(Thread);
  end;
end;
```

Important security issues

1. When using a report server on Microsoft Windows platform over the Internet, it is recommended to use a firewall between server and internet network.
2. It is obligatory to use the authentication of the client program (section 3.8).
3. Use the "allow/deny IP" function in local network (section 3.9).
4. If you have any gateways to Internet in local network, then include IP addresses of these gateways to the "deny" list of the report server (section 3.9).
5. Do not pass parameters to database connection from client if you use reports with internal database components.
6. In reports folder, store only those reports, which you use in your application.
7. Do not store any private documents in the HTTP root folder.
8. If you find any bugs in security system of the FastReport Enterprise, send a note to the developers of the product.

References

1. Braden, R., "Requirements for Internet hosts - application and support", STD 3, RFC 1123, IETF, October 1989.
2. Berners-Lee, T., Fielding, R., and H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.1" RFC 2068, January 1997.
3. Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., and L. Stewart, "An Extension to HTTP: Digest Access Authentication", RFC 2069, January 1997.
4. Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
5. Meyers, J., and M. Rose "The Content-MD5 Header Field", RFC 1864, Carnegie Mellon, Dover Beach Consulting, October, 1995.
6. Deutsch, P., "GZIP file format specification version 4.3." RFC 1952, Aladdin Enterprises, May 1996.

Developers' contact information

If you have any suggestions concerning improvement and development of FastReport VCL Enterprise, please contact us:

e-mail: support@fast-report.com

web site: https://www.fast-report.com