uniGUI

Evaluation / Migration

OCTOBER 22, 2015

IQMS

uniGUI (unified Graphical User Interface)

Evaluation / Migration

Contents

Introduction

uniGUI's web page lists the most important features, but I'll add a few comments to what is said.

uniGUI is a framework for developing AJAX Web Applications in classical WYSIWYG Delphi RAD way. Its unique set of components allows developing web applications at lightning speed.

One of the unique features of uniGUI is its ability to use a single code base / resource base for creating a standalone Web Server application or a hosted ISAPI Module.

uniGUI itself is not a single library. It relies on a few other libraries with the uniGUI core at the center.

- For Web front-end, the well-known Sencha Ext JS JavaScript library is used.
- For touch based web devices Sencha Touch is used.

uniGUI enables developers to create, design and debug their Delphi projects as regular desktop applications and then chose one of the available options for Web Server deployment such as Standalone, ISAPI, or Windows Service.

Features:

- Based on industry standard Ext JS cross-browser JavaScript library.
    - Currently supporting version 4.2.4 (latest version is 6)
- Creates stateful Web applications.
- Complete IDE support for creating projects, designing forms and handling data modules.
    - It requires using TUni* visual components
    - Data access is not affected
- Delphi style event handling.
    - Events belong to TUni* visual components
- Advanced client-side event scripting
- Automatic handling of Delphi data modules per session.
- Deployment options:
    - ISAPI Web application,
    - Standalone Web application,
    - Windows Service Web application.
- Supported Delphi versions: Turbo Delphi Pro, Delphi 2006, Delphi 2007, Delphi 2009, Delphi 2010, Delphi XE, Delphi XE2, XE3, XE4, X5, XE6, XE7, XE8 and Delphi 10 (Win32 & Win64).
- Supported C++ Builder versions: C++ Builder 2006 - XE7.
- Available in two editions (volume pricing available):
    - uniGUI – Professional Edition ($680) for creating desktop Web Applications. Includes OEM license for Sencha Ext JS
    - uniGUI Complete – Professional Edition ($890) for creating desktop and mobile Web Applications (touch enabled). Includes OEM licenses for Sencha Ext JS and Sencha Touch.

uniGUI is been for sale, in beta state, for more than a year. Customer support, both for users trying the product or registered users is provided in forums.

FMSoft provides almost 100 hundred demo projects, in addition to third-party projects created by uniGUI's users.

uniGUI installation

Installation is partially automated, everything gets installed, except for the Delphi packages which are created for the selected Delphi versions and just need to be compiled / installed. Instructions are provided in document uniGUI.PDF.

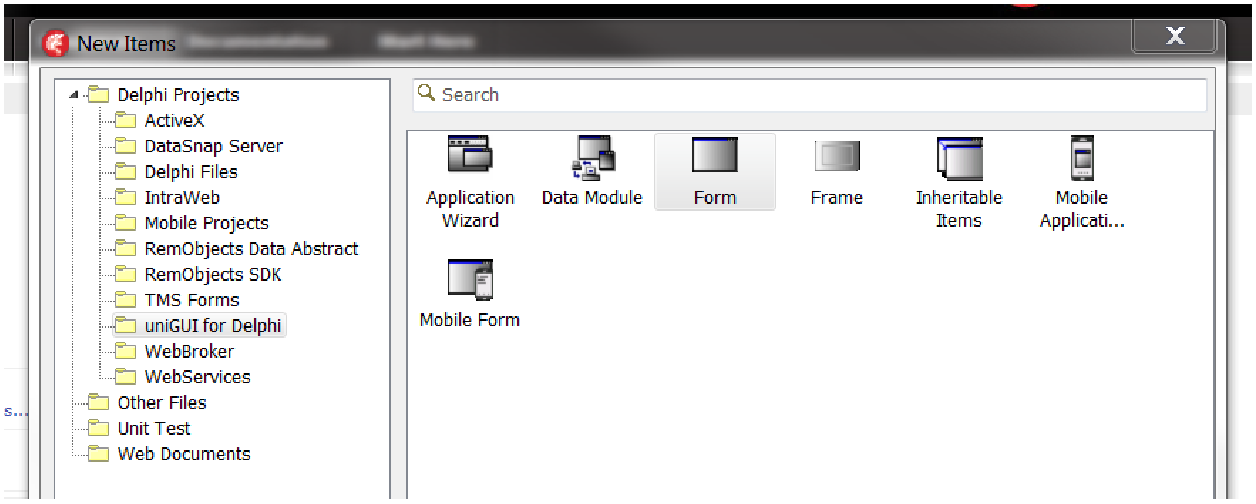
Once installed, it is not necessary to download and install Sencha Ext JS products. A set of templates is added to Delphi Projects\uniGUI for Delphi with wizards for a desktop / Web application, data module, form, frame, inheritable items, and mobile application and forms.

First uniGUI application

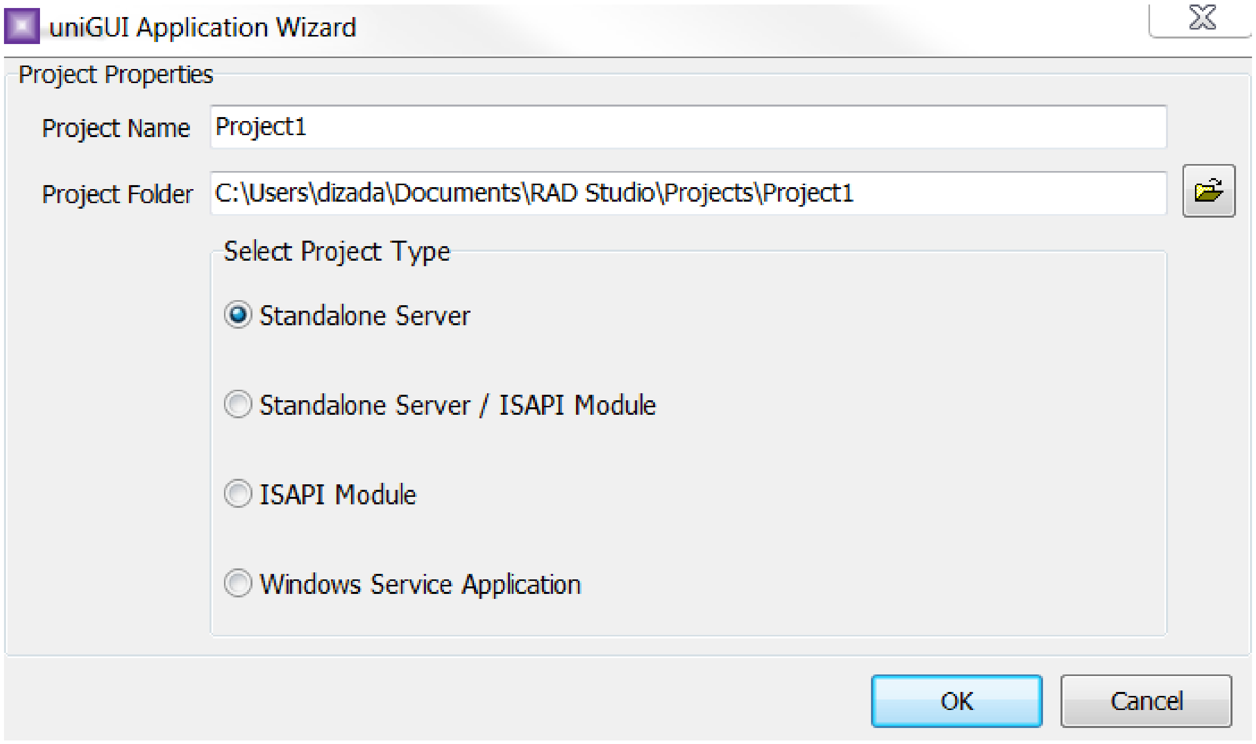
uniGUI projects are created as desktop / Web applications or Mobile applications (touch-enabled Web applications).

Creating a Web Application project

A new application is created by opening Delphi and selecting File | New | Other and selecting "uniGUI for Delphi".



Select "Application Wizard" to start this wizard:


uniGUI Application Wizard
Project Properties
Select Project Type
Standalone Server / ISAPI Module
ISAPI Module
Windows Service Application
OK
Cancel

Project Name: Project1

Project Folder: C:\Users\dizada\Documents\RAD Studio\Projects\Project1

Select Project Type: [✓ Standalone Server]

A **Standalone Server** is an executable which will run in a server session, listening to a port (default is 8077), serving pages to multiple clients. If writing more than one application, each one of them will need its own port.

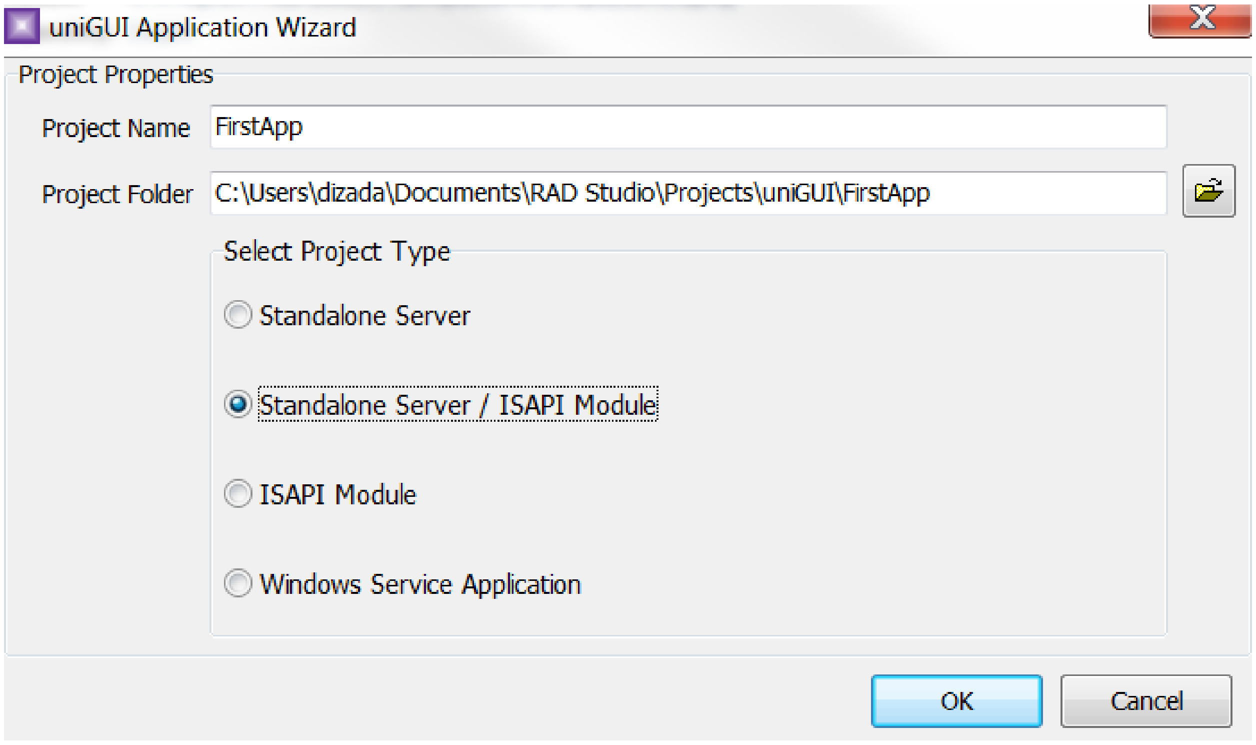
An **ISAPI Module** is a standard Microsoft IIS module which can be run under IIS as any other ISAPI module. This is the preferred deployment option.

Asking for **Standalone Server / ISAPI Module** will create a combo application capable of being deployed as both previous options.

A **Windows Service Application** is a Standalone Server running as a Windows Service (not requiring a server session).

The best deployment option is the **ISAPI Module**, but the best way to start could be the **Standalone Server** which allows to run the application as a locally, self-hosted, web application, easier to test in the web because it won't require any IIS configuration.

Probably, the best option to start developing the application will be the combo **Standalone Server / ISAPI Module**.



uniGUI Application Wizard

Project Properties

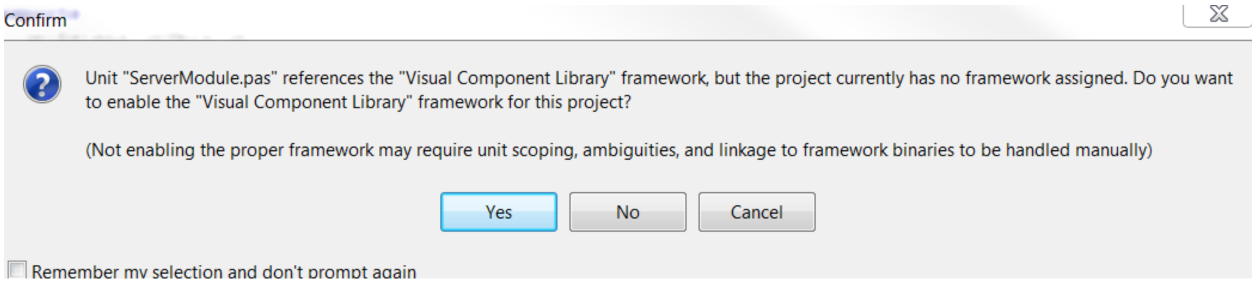
Project Name: FirstApp

Project Folder: C:\Users\dizada\Documents\RAD Studio\Projects\uniGUI\FirstApp

Select Project Type

- Standalone Server
- [✓ Standalone Server / ISAPI Module]
- ISAPI Module
- Windows Service Application

After selecting our preferred project type, the following prompt is shown:



Confirm

Unit "ServerModule.pas" references the "Visual Component Library" framework, but the project currently has no framework assigned. Do you want to enable the "Visual Component Library" framework for this project?

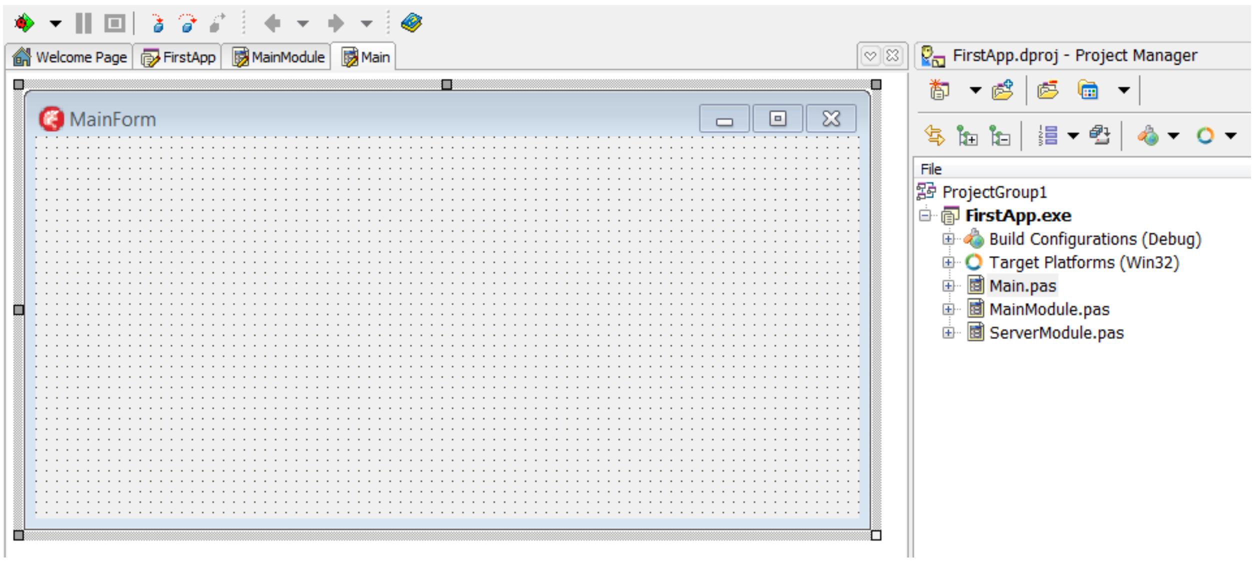
(Not enabling the proper framework may require unit scoping, ambiguities, and linkage to framework binaries to be handled manually)

Remember my selection and don't prompt again

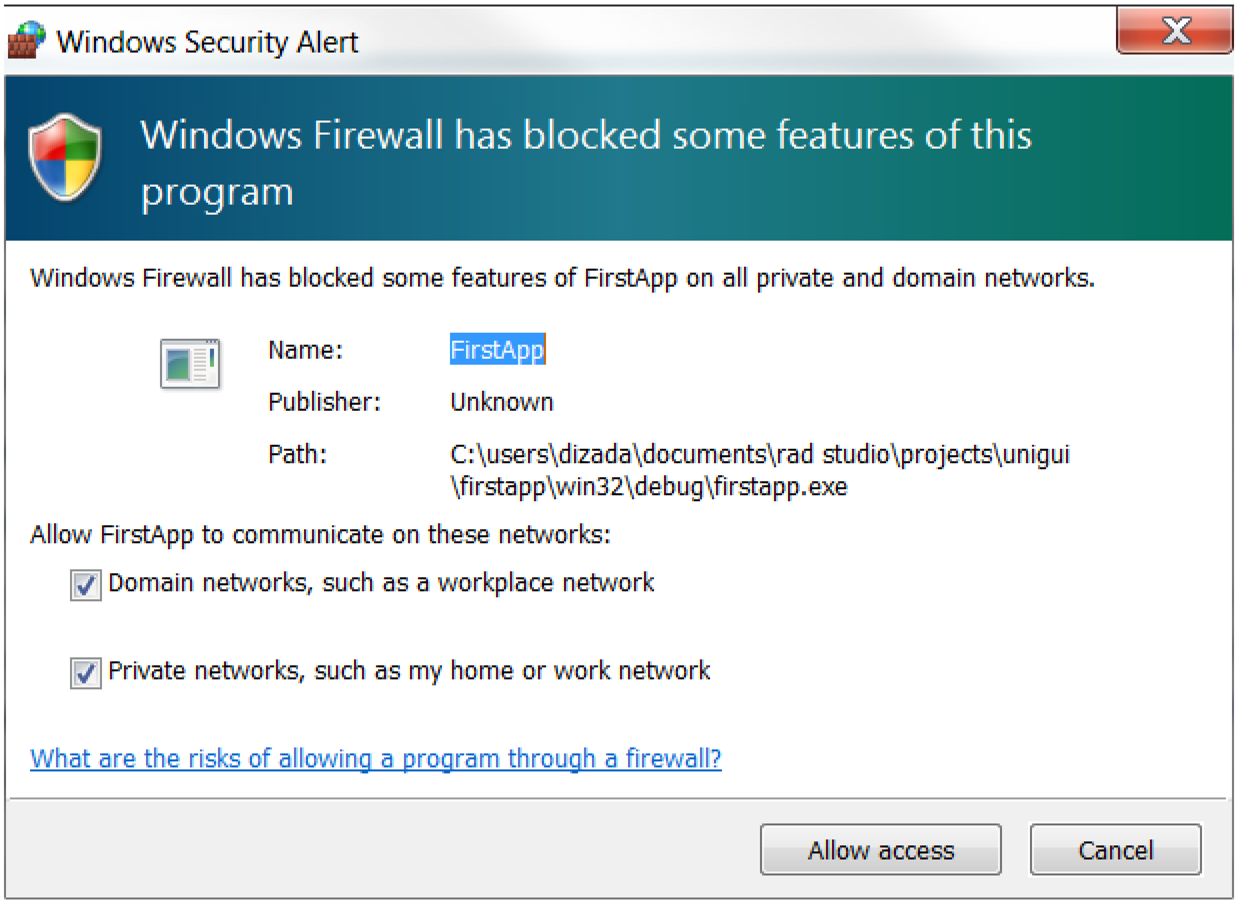
As uniGUI only supports VCL, the correct answer is "Yes".

Running the application

The new application is ready:



Running the application will require allowing permission to open the configured port (default 8077) and Windows Firewall pop ups this dialog:



Windows Security Alert

Windows Firewall has blocked some features of this program

Windows Firewall has blocked some features of FirstApp on all private and domain networks.

Name: FirstApp

Publisher: Unknown

Path: C:\users\dizada\documents\rad studio\projects\unigui \firstapp\win32\debug\firstapp.exe

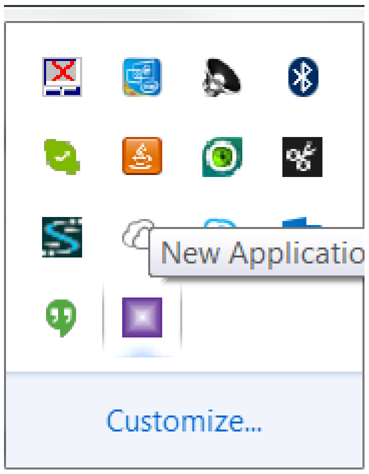
Allow FirstApp to communicate on these networks:

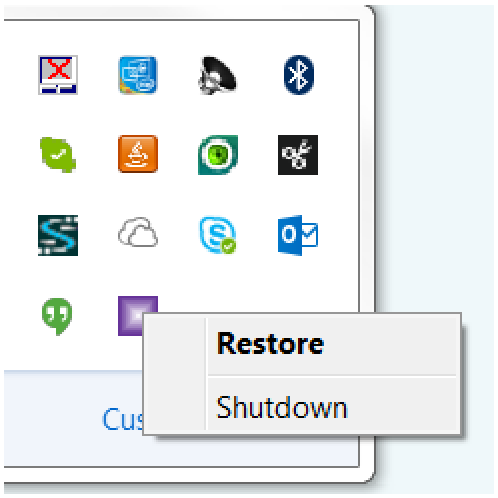
[✓] Domain networks, such as a workplace network

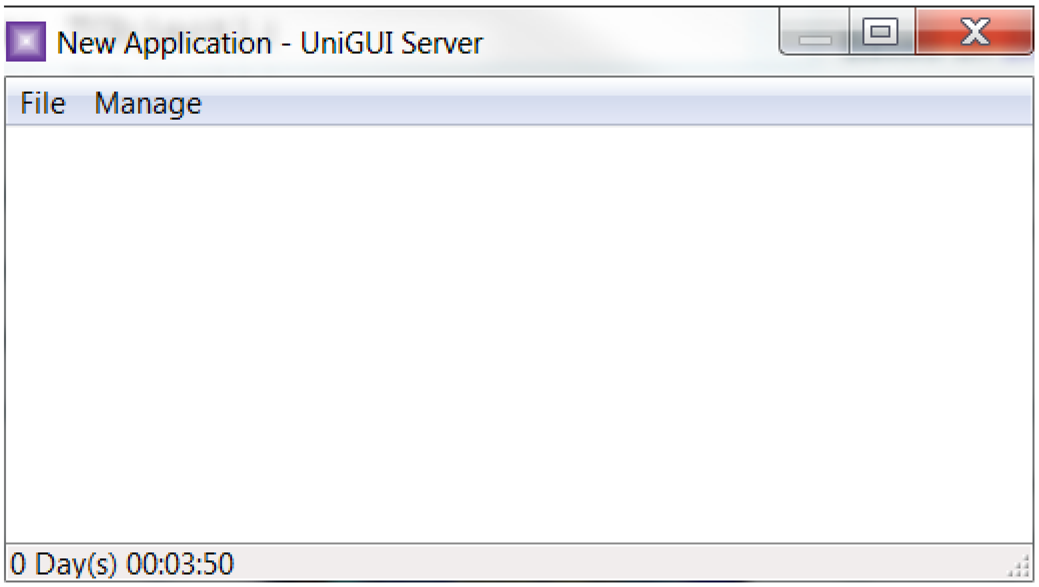
[✓] Private networks, such as my home or work network

What are the risks of allowing a program through a firewall?

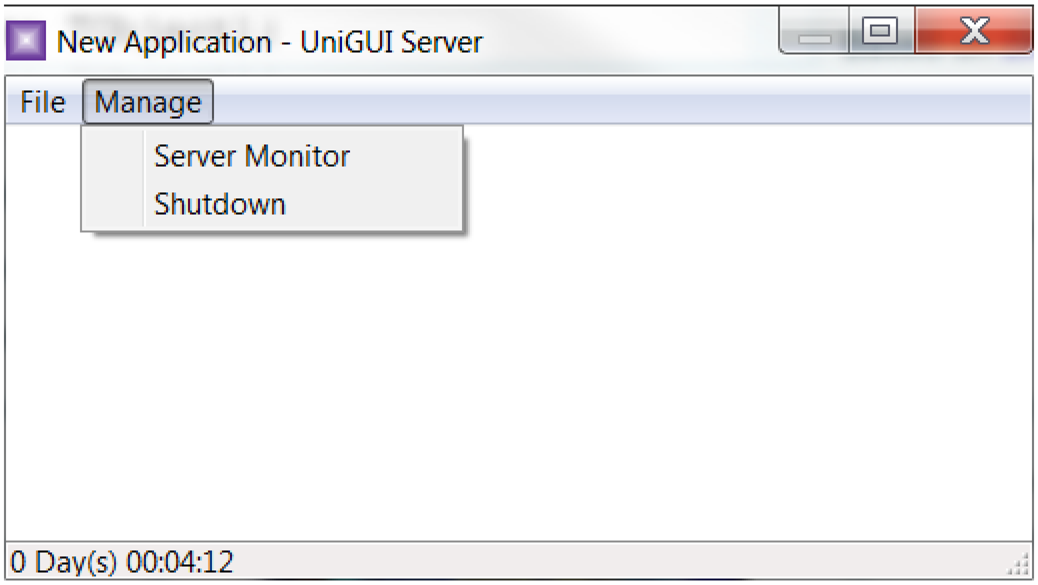
After allowing access, the application will be running in debug mode, and it will be possible to monitor and manage it from the system tray.





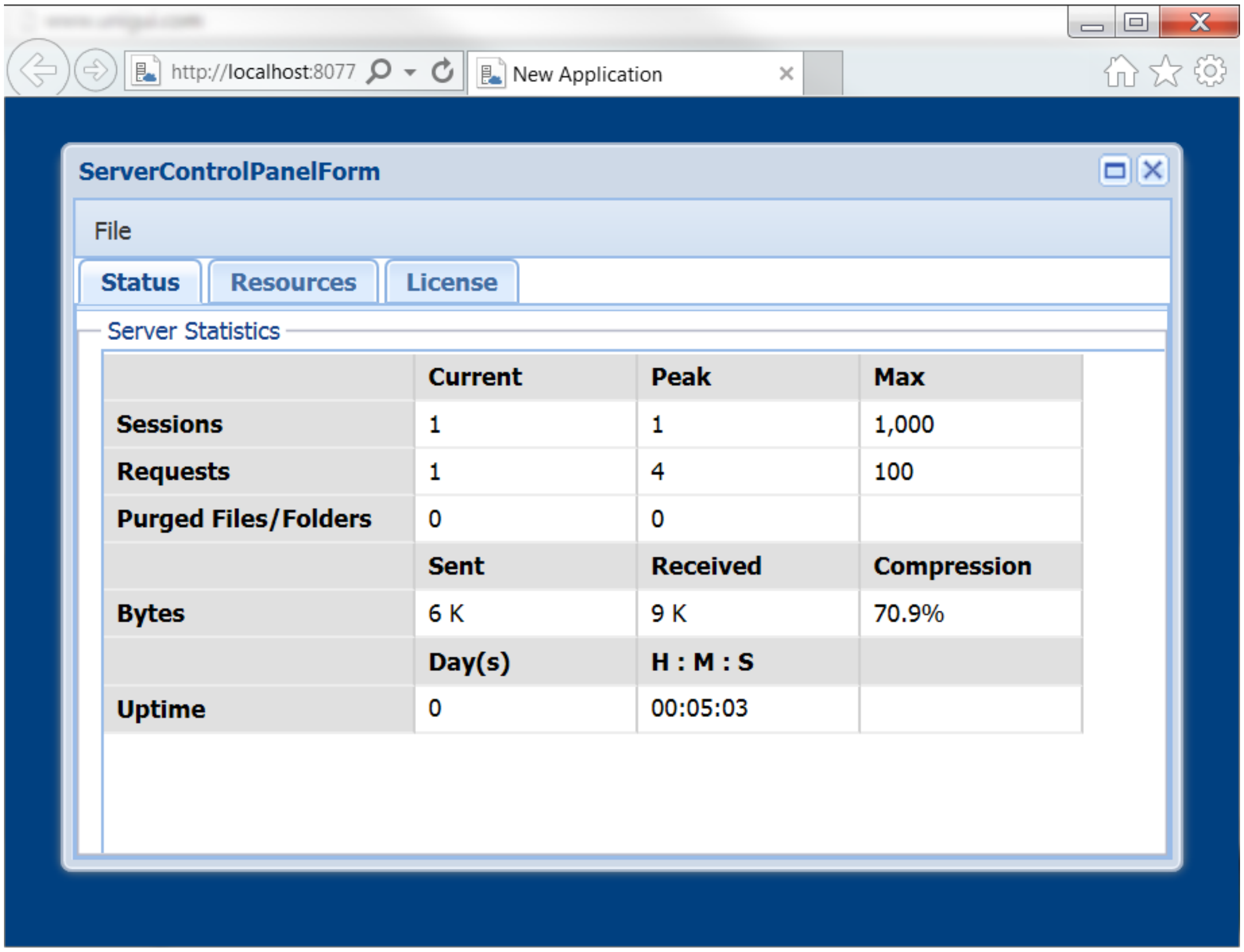


0 Day(s) 00:03:50



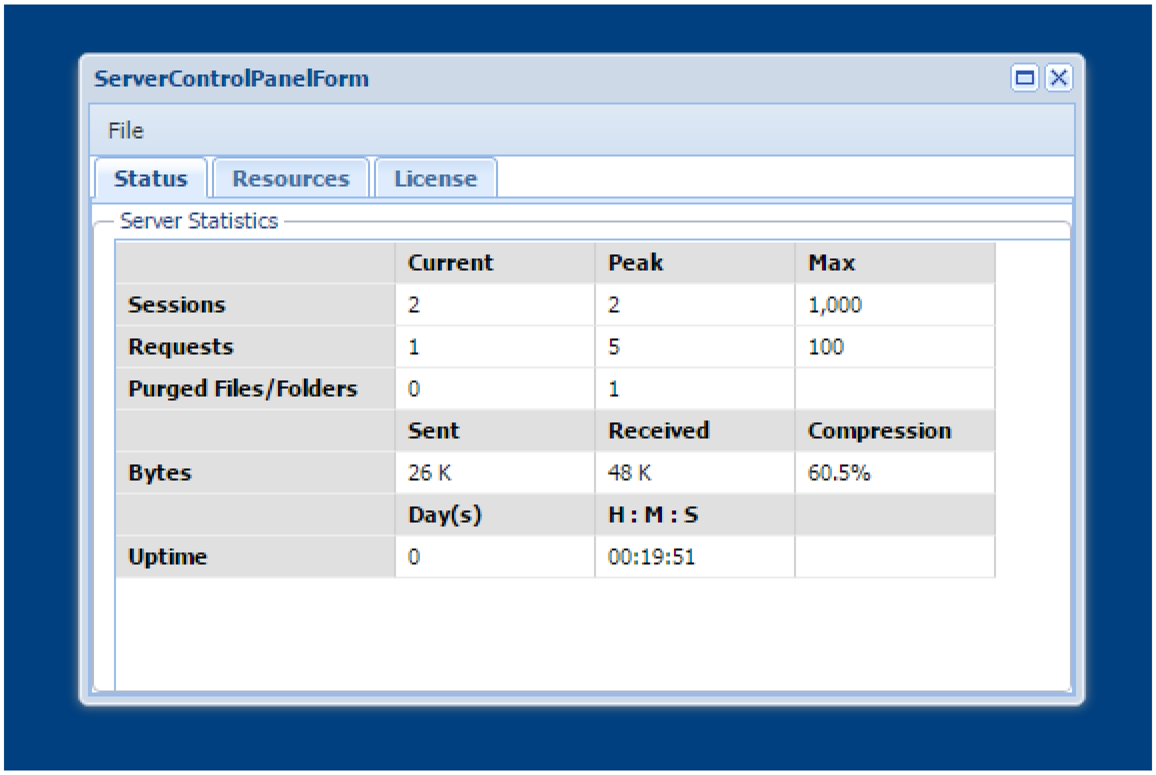
0 Day(s) 00:04:12

Server Monitor opens this web page (a uniGUI application itself):



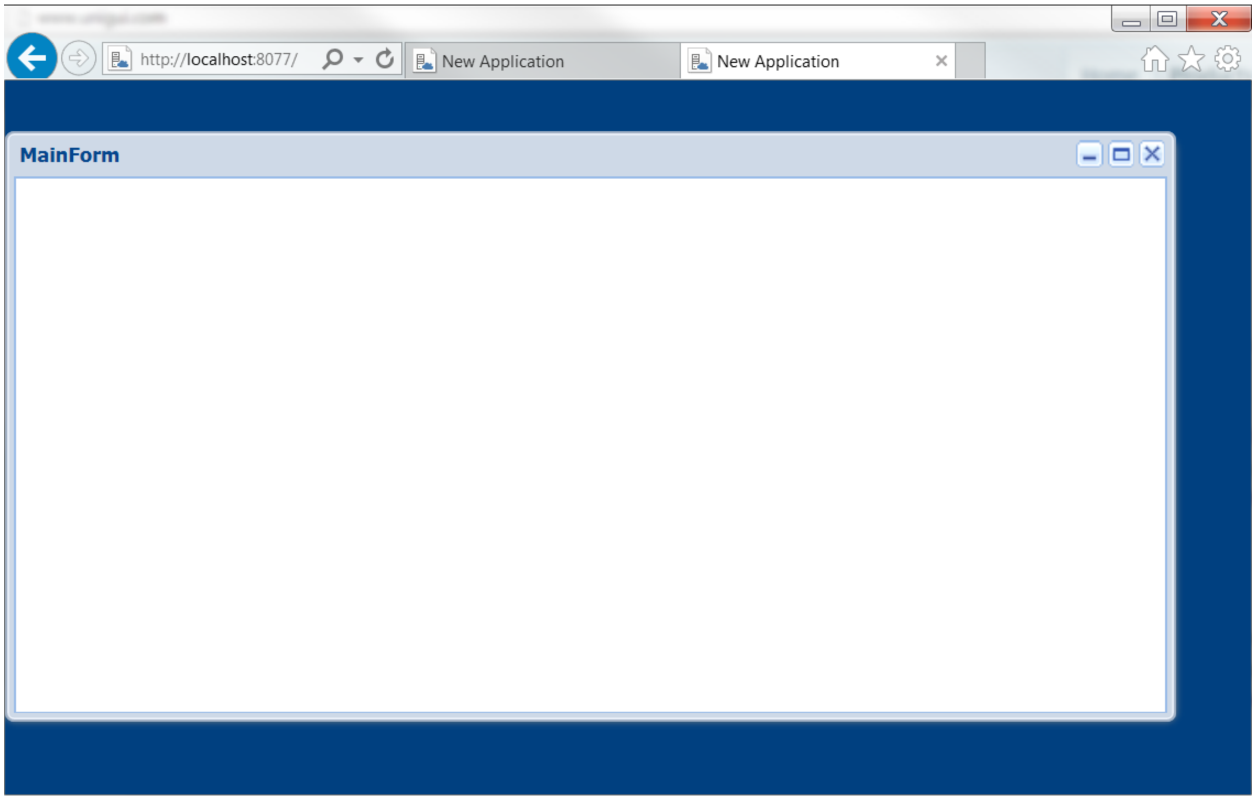
	Current	Peak	Max
Sessions	1	1	1,000
Requests	1	4	100
Purged Files/Folders	0	0	
	Sent	**Received**	**Compression**
Bytes	6 K	9 K	70.9%
	Day(s)	**H : M : S**	
Uptime	0	00:05:03	

The same screen as rendered by Google Chrome:

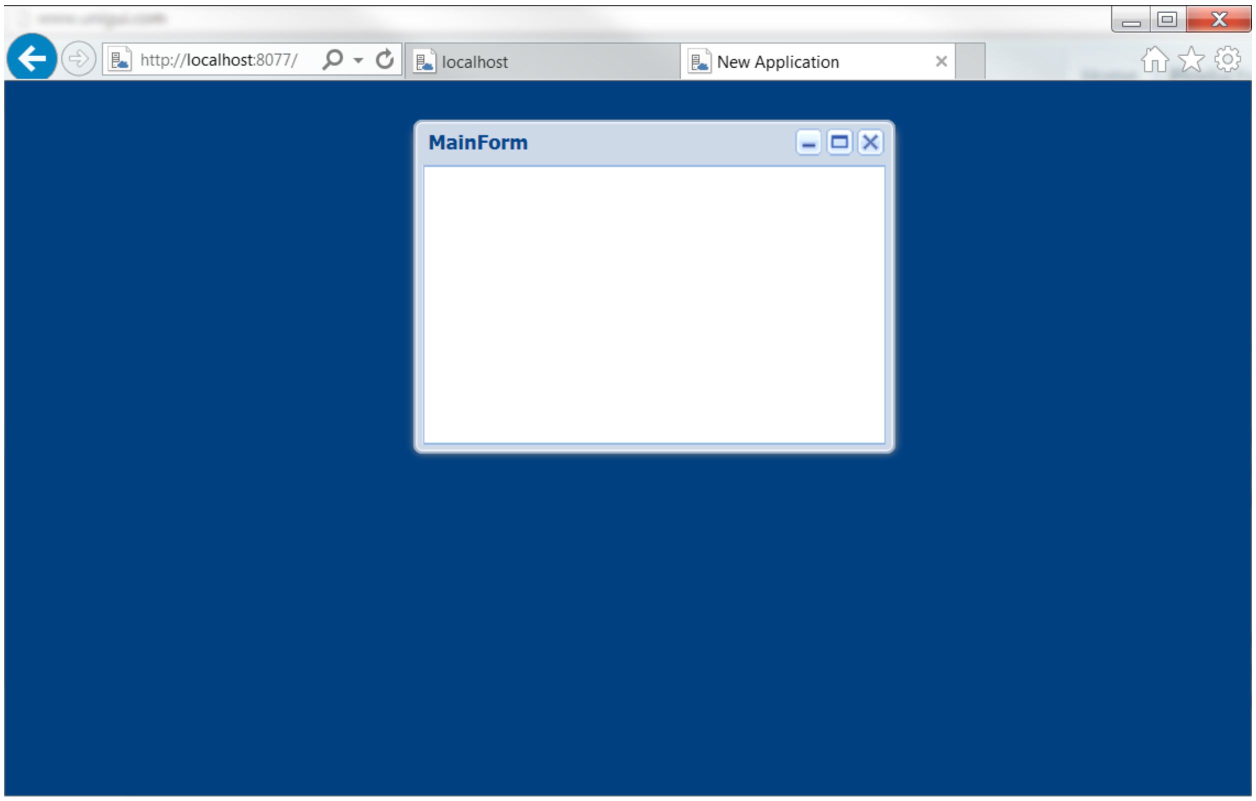


	Current	Peak	Max
Sessions	2	2	1,000
Requests	1	5	100
Purged Files/Folders	0	1	
	Sent	Received	Compression
Bytes	26 K	48 K	60.5%
	Day(s)	H : M : S	
Uptime	0	00:19:51	

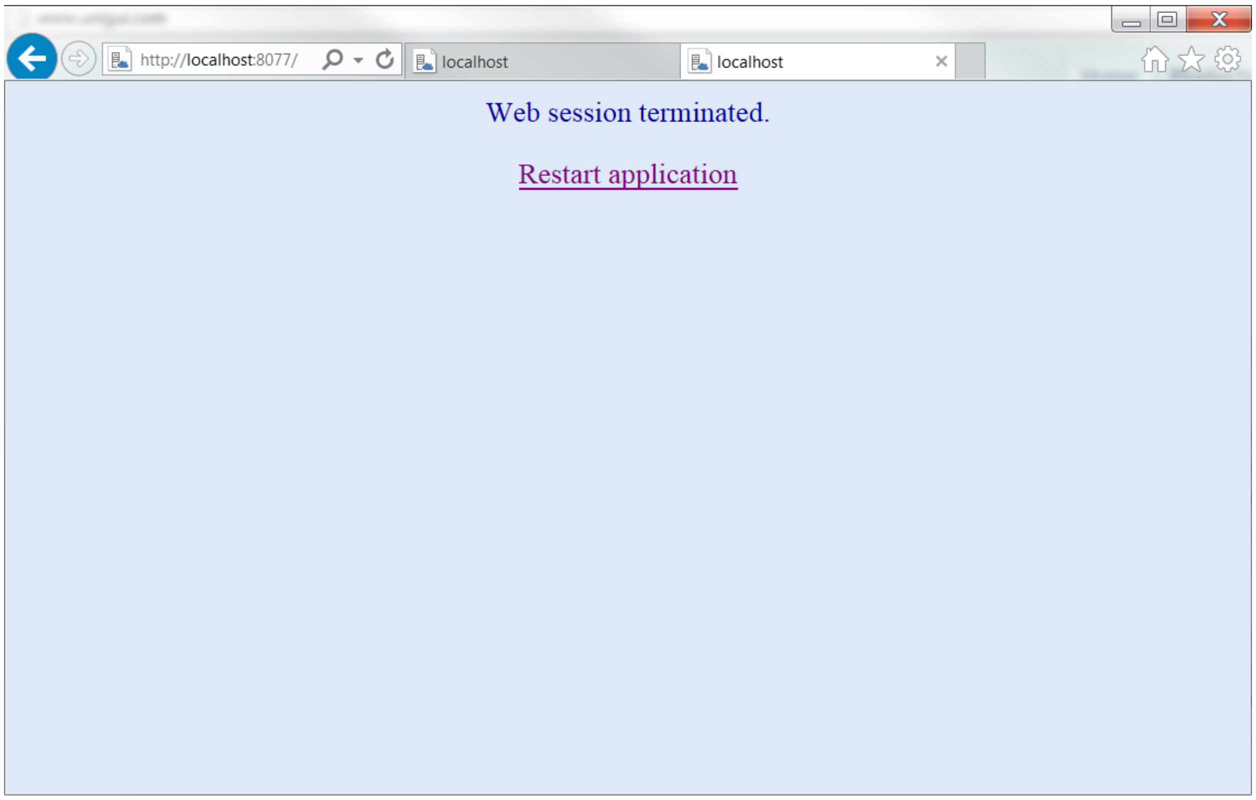
Our first application will run in the browser using URL http://localhost:8077



It won’t do anything useful, but it can be resized and dragged inside the browser window.



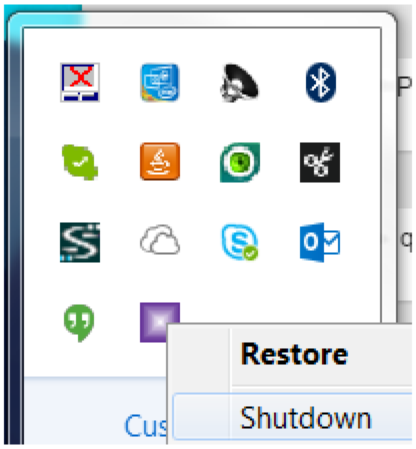
Of course, it is possible to close it. It will close this client's session; the server is still running.



Web session terminated.

Restart application

Stopping the server requires selecting Shutdown in the tray application



Anatomy of the simplest application

FirstApp.dpr

```
{$define UNIGUI_VCL} // Comment out this line to turn this project into an
ISAPI module'

{$ifndef UNIGUI_VCL}
library
{$else}
program
{$endif}
  FirstApp;

uses
  uniGUIISAPI,
  Forms,
  ServerModule in 'ServerModule.pas' {UniServerModule: TUniGUIServerModule},
  MainModule in 'MainModule.pas' {UniMainModule: TUniGUIMainModule},
  Main in 'Main.pas' {MainForm: TUniForm};

{$R *.res}

{$ifndef UNIGUI_VCL}
exports
  GetExtensionVersion,
  HttpExtensionProc,
  TerminateExtension;
{$endif}

begin
{$ifdef UNIGUI_VCL}
  Application.Initialize;
  TUniServerModule.Create(Application);
  Application.Run;
{$endif}
end.
```

Conditional compilation allows to create an executable .EXE for running the application as a Standalone Server or a dynamic link library .DLL to be used as an ISAPI Module with Microsoft Internet Information Server.

MainModule.pas

```
unit MainModule;

interface

uses
  uniGUIMainModule, SysUtils, Classes;

type
  TUniMainModule = class(TUniGUIMainModule)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

function UniMainModule: TUniMainModule;

implementation

{$R *.dfm}

uses
  UniGUIVars, ServerModule, uniGUIApplication;

function UniMainModule: TUniMainModule;
begin
  Result := TUniMainModule(UniApplication.UniMainModule)
end;

initialization
  RegisterMainModuleClass(TUniMainModule);
end.
```

This module exposes UniMainModule which provides services like creating TUniForm at runtime.

Main.pas

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, uniGUITypes, uniGUIAbstractClasses,
  uniGUIClasses, uniGUIRegClasses, uniGUIForm;

type
  TMainForm = class(TUniForm)
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;

function MainForm: TMainForm;

implementation

{$R *.dfm}

uses
  uniGUIVars, MainModule, uniGUIApplication;

function MainForm: TMainForm;
begin
  Result := TMainForm(UniMainModule.GetFormInstance(TMainForm));
end;

initialization
  RegisterAppFormClass(TMainForm);

end.
```

This is the usual Main form created in Delphi, uniGUI-style.

Notice that it is a TUniForm (as any other visual component in a uniGUI application) and that, instead of using a VAR declaration for the form, it is a function which uses UniMainModule service GetFormInstance for creating every requested instance.

ServerModule.pas

```
uses
  Classes, SysUtils, uniGUIServer, uniGUIMainModule, uniGUIApplication,
uIdCustomHTTPServer,
  uniGUITypes;

type
  TUniServerModule = class(TUniGUIServerModule)
  private
    { Private declarations }
  protected
    procedure FirstInit; override;
  public
    { Public declarations }
  end;

function UniServerModule: TUniServerModule;

implementation

{$R *.dfm}

uses
  UniGUIVars;

function UniServerModule: TUniServerModule;
```

```
begin
  Result:=TUniServerModule(UniGUIServerInstance);
end;

procedure TUniServerModule.FirstInit;
begin
  InitServerModule(Self);
end;

initialization
  RegisterServerModuleClass(TUniServerModule);
end.
```

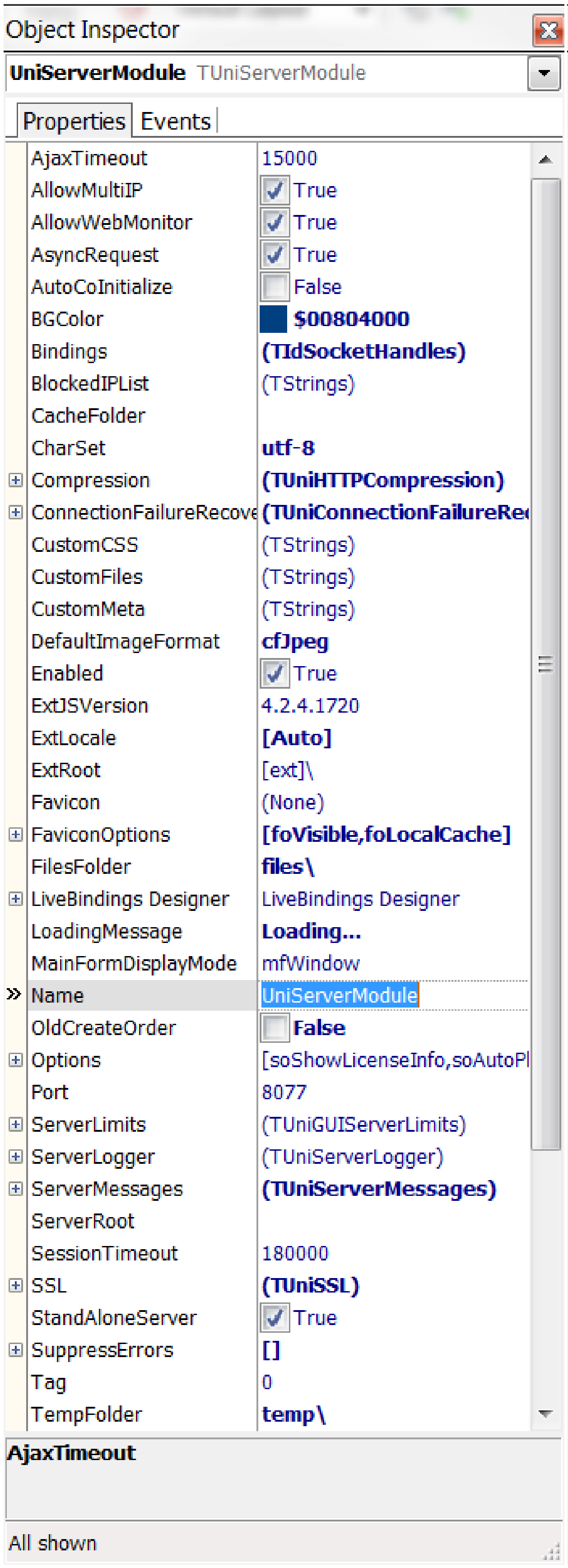
For this module, the source code says almost nothing. Everything important is in the module's properties:

ServerModule.dfm

```
object UniServerModule: TUniServerModule
  OldCreateOrder = False
  FilesFolder = 'files\'
  TempFolder = 'temp\'
  Title = 'New Application'
  BGColor = 8404992
  CharSet = 'utf-8'
  FaviconOptions = [foVisible, foLocalCache]
  DefaultImageFormat = cfJpeg
  SuppressErrors = []
  UnavailableErrMsg = 'Communication Error'
  LoadingMessage = 'Loading...'
  Bindings = <>
  ServerMessages.ExceptionTemplate.Strings = (
    '<html>'
    '<body bgcolor="#dfe8f6">'

      '<p style="text-align:center;color:#A05050">An Exception has occured in 
application:</p>'
      '<p style="text-align:center;color:#0000A0">[###message###]</p>'

      '<p style="text-align:center;color:#A05050"><a 
href="[###url###]">Restart application</a></p>'
    '</body>'
    '</html>')
  ServerMessages.InvalidSessionTemplate.Strings = (
    '<html>'
    '<body bgcolor="#dfe8f6">'
      '<p style="text-align:center;color:#0000A0">[###message###]</p>'

      '<p style="text-align:center;color:#A05050"><a 
href="[###url###]">Restart application</a></p>'
    '</body>'
    '</html>')
  ServerMessages.TerminateTemplate.Strings = (
    '<html>'
    '<body bgcolor="#dfe8f6">'
    '<p style="text-align:center;color:#0000A0">[###message###]</p>'
```

```
      '<p style="text-align:center;color:#A05050"><a
href="[###url###]">Restart application</a></p>'
    '</body>'
    '</html>')
  ServerMessages.InvalidSessionMessage = 'Invalid session or session
Timeout.'
  ServerMessages.TerminateMessage = 'Web session terminated.'
  ExtLocale = '[Auto]'
  Compression.MinTextSize = 512
  SSL.SSLOptions.RootCertFile = 'root.pem'
  SSL.SSLOptions.CertFile = 'cert.pem'
  SSL.SSLOptions.KeyFile = 'key.pem'
  SSL.SSLOptions.Method = sslvTLSv1_1
  SSL.SSLOptions.SSLVersions = [sslvTLSv1_1]
  SSL.SSLOptions.Mode = sslmUnassigned
  SSL.SSLOptions.VerifyMode = []
  SSL.SSLOptions.VerifyDepth = 0
  ConnectionFailureRecovery.ErrorMessage = 'Connection Error'
  ConnectionFailureRecovery.RetryMessage = 'Retrying...'
  Height = 150
  Width = 215
end
```

A better view will be:



Object Inspector

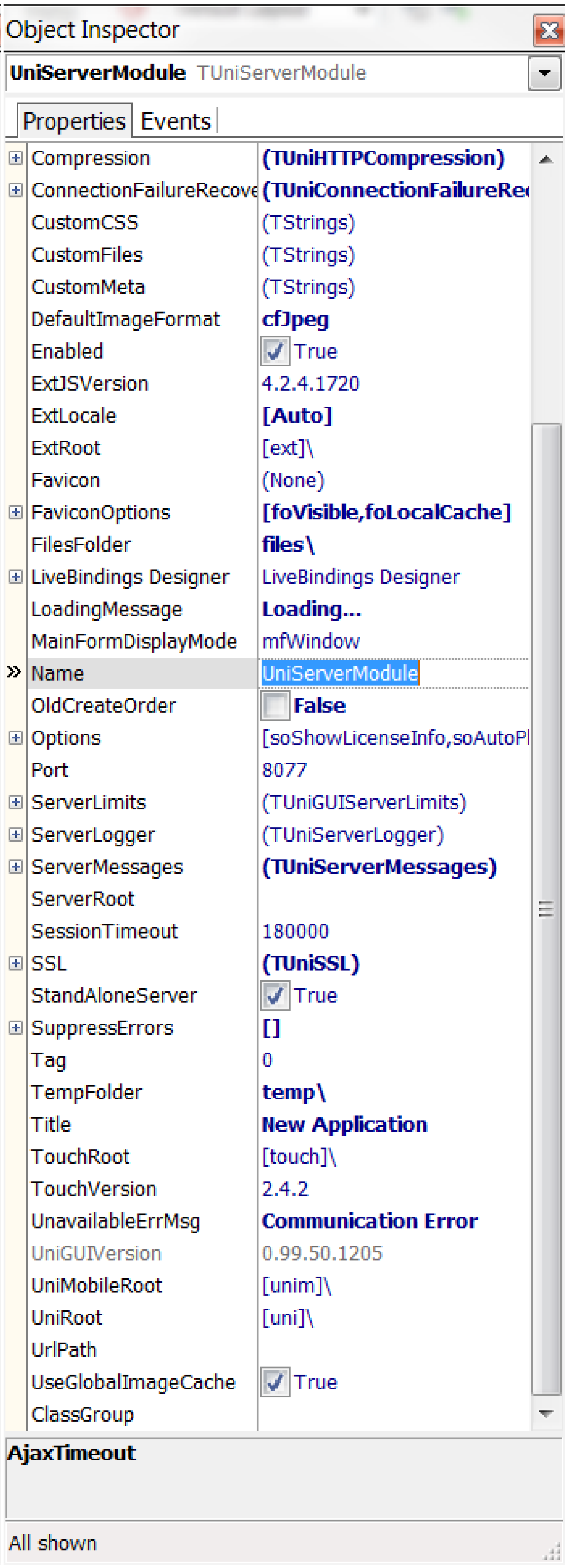
UniServerModule TUniServerModule

Properties | Events

AjaxTimeout	15000
AllowMultiIP	✓ True
AllowWebMonitor	✓ True
AsyncRequest	✓ True
AutoCoInitialize	☐ False
BGColor	🟦 **$00804000**
Bindings	**(TIdSocketHandles)**
BlockedIPList	(TStrings)
CacheFolder	
CharSet	**utf-8**
⊞ Compression	**(TUniHTTPCompression)**
⊞ ConnectionFailureRecove	**(TUniConnectionFailureRe**
CustomCSS	(TStrings)
CustomFiles	(TStrings)
CustomMeta	(TStrings)
DefaultImageFormat	**cfJpeg**
Enabled	✓ True
ExtJSVersion	4.2.4.1720
ExtLocale	**[Auto]**
ExtRoot	[ext]\
Favicon	(None)
⊞ FaviconOptions	**[foVisible,foLocalCache]**
FilesFolder	**files**
⊞ LiveBindings Designer	LiveBindings Designer
LoadingMessage	**Loading...**
MainFormDisplayMode	mfWindow
» Name	UniServerModule
OldCreateOrder	☐ **False**
⊞ Options	[soShowLicenseInfo,soAutoPl
Port	8077
⊞ ServerLimits	(TUniGUIServerLimits)
⊞ ServerLogger	(TUniServerLogger)
⊞ ServerMessages	**(TUniServerMessages)**
ServerRoot	
SessionTimeout	180000
⊞ SSL	**(TUniSSL)**
StandAloneServer	✓ True
⊞ SuppressErrors	**[]**
Tag	0
TempFolder	**temp**

AjaxTimeout

All shown



Object Inspector

UniServerModule TUniServerModule

Properties | Events

⊞ Compression	**(TUniHTTPCompression)**
⊞ ConnectionFailureRecove	**(TUniConnectionFailureRe**
CustomCSS	(TStrings)
CustomFiles	(TStrings)
CustomMeta	(TStrings)
DefaultImageFormat	**cfJpeg**
Enabled	✓ True
ExtJSVersion	4.2.4.1720
ExtLocale	**[Auto]**
ExtRoot	[ext]\
Favicon	(None)
⊞ FaviconOptions	**[foVisible,foLocalCache]**
FilesFolder	**files**
⊞ LiveBindings Designer	LiveBindings Designer
LoadingMessage	**Loading...**
MainFormDisplayMode	mfWindow
» Name	UniServerModule
OldCreateOrder	☐ **False**
⊞ Options	[soShowLicenseInfo,soAutoPl
Port	8077
⊞ ServerLimits	(TUniGUIServerLimits)
⊞ ServerLogger	(TUniServerLogger)
⊞ ServerMessages	**(TUniServerMessages)**
ServerRoot	
SessionTimeout	180000
⊞ SSL	**(TUniSSL)**
StandAloneServer	✓ True
⊞ SuppressErrors	**[]**
Tag	0
TempFolder	**temp**
Title	**New Application**
TouchRoot	[touch]\
TouchVersion	2.4.2
UnavailableErrMsg	**Communication Error**
UniGUIVersion	0.99.50.1205
UniMobileRoot	[unim]\
UniRoot	[uni]\
UrlPath	
UseGlobalImageCache	✓ True
ClassGroup	

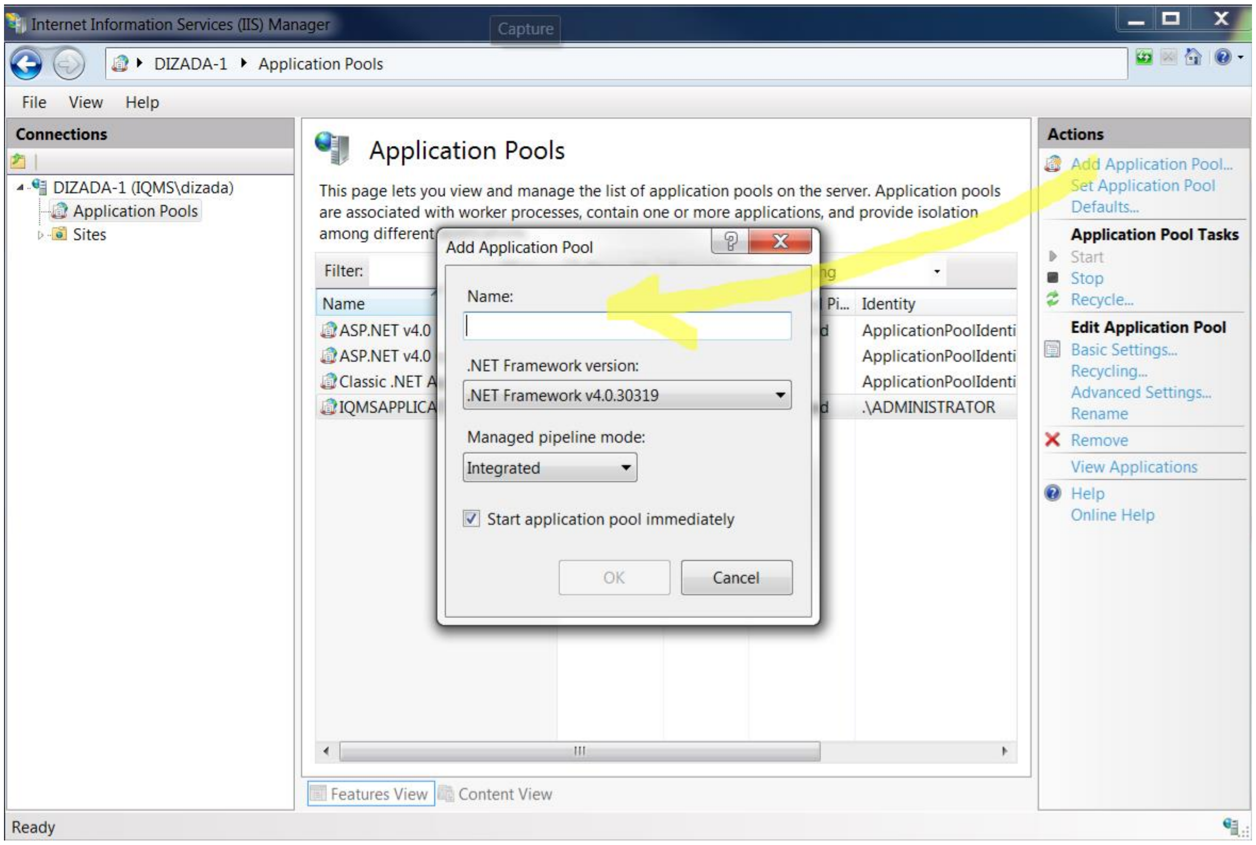
AjaxTimeout

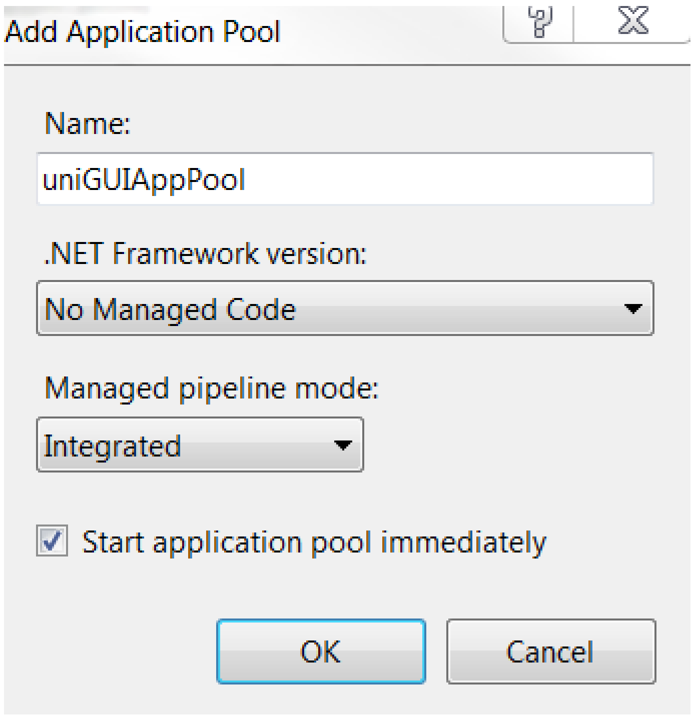
All shown

ServerModule is the place where the Web Application is configured (things like SessionTimeout, StandaloneServer, etc.)

Deploying the simplest application

Create and configure a new Application Pool



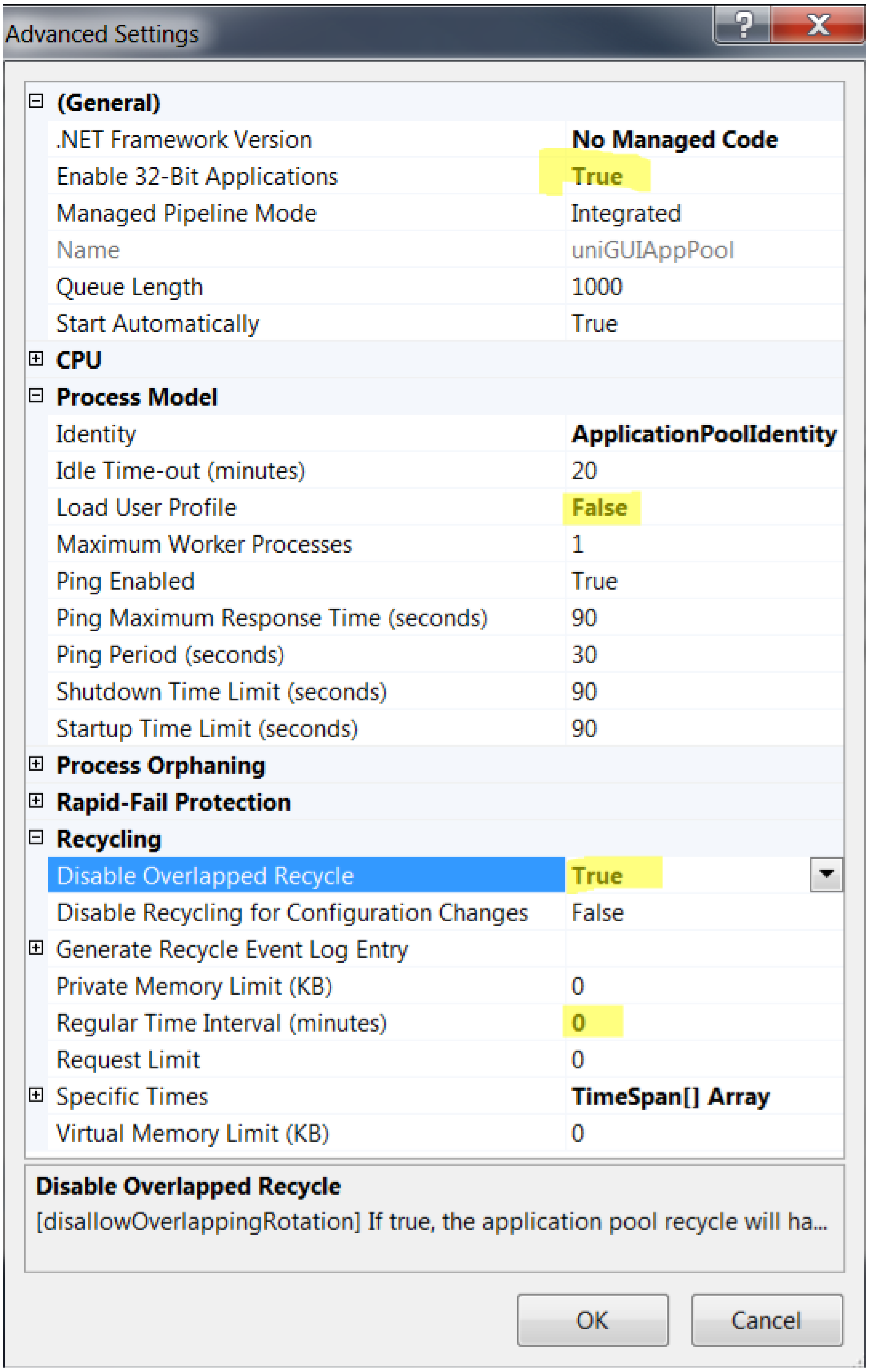


Name: uniGUIAppPool

.NET Framework version: No Managed Code

Managed pipeline mode: Integrated

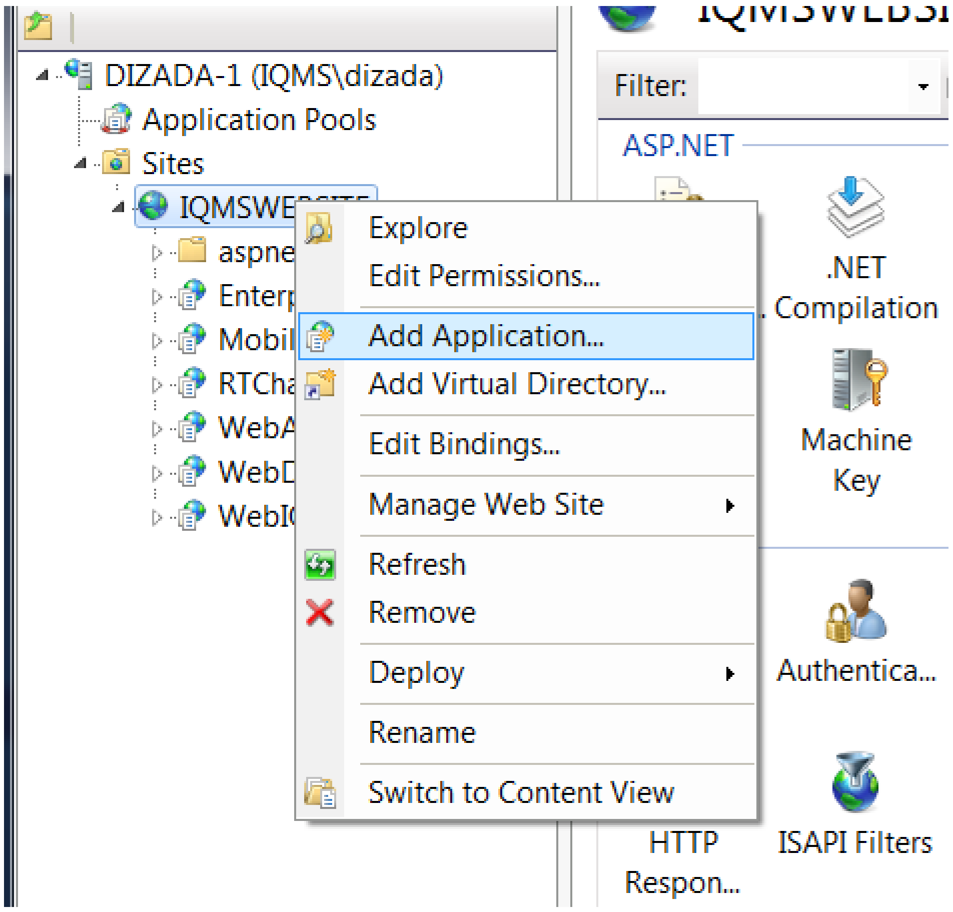
✓ Start application pool immediately

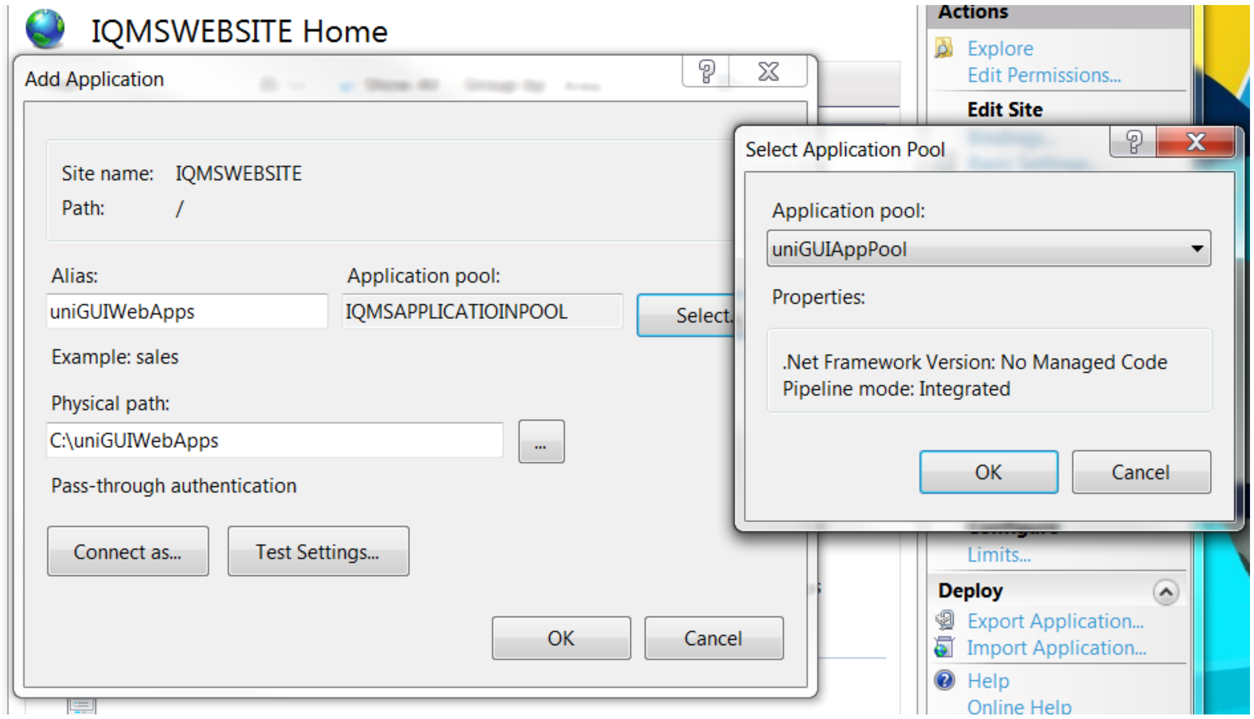

Advanced Settings

(General)	
.NET Framework Version	**No Managed Code**
Enable 32-Bit Applications	**True**
Managed Pipeline Mode	Integrated
Name	uniGUIAppPool
Queue Length	1000
Start Automatically	True
CPU	
Process Model	
Identity	**ApplicationPoolIdentity**
Idle Time-out (minutes)	20
Load User Profile	**False**
Maximum Worker Processes	1
Ping Enabled	True
Ping Maximum Response Time (seconds)	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90
Process Orphaning	
Rapid-Fail Protection	
Recycling	
Disable Overlapped Recycle	**True**
Disable Recycling for Configuration Changes	False
Generate Recycle Event Log Entry	
Private Memory Limit (KB)	0
Regular Time Interval (minutes)	**0**
Request Limit	0
Specific Times	**TimeSpan[] Array**
Virtual Memory Limit (KB)	0

Disable Overlapped Recycle
[disallowOverlappingRotation] If true, the application pool recycle will ha...

Create and configure a new Application




IQMSWEBSITE Home

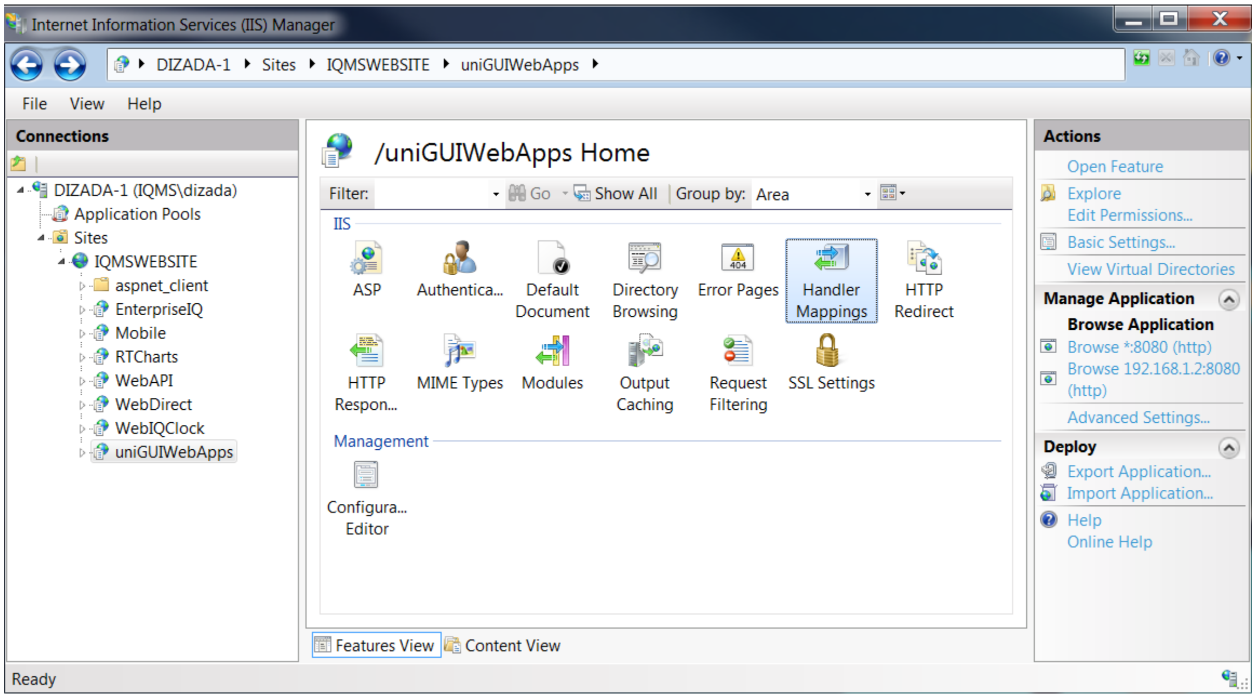
Add Application

Site name: IQMSWEBSITE
Path: /

Alias: uniGUIWebApps
Application pool: IQMSAPPLICATIOINPOOL
Example: sales
Physical path: C:\uniGUIWebApps
Pass-through authentication

Select Application Pool
Application pool: uniGUIAppPool
Properties:
.Net Framework Version: No Managed Code
Pipeline mode: Integrated

Modify Handler Mappings to enable ISAPI DLLs

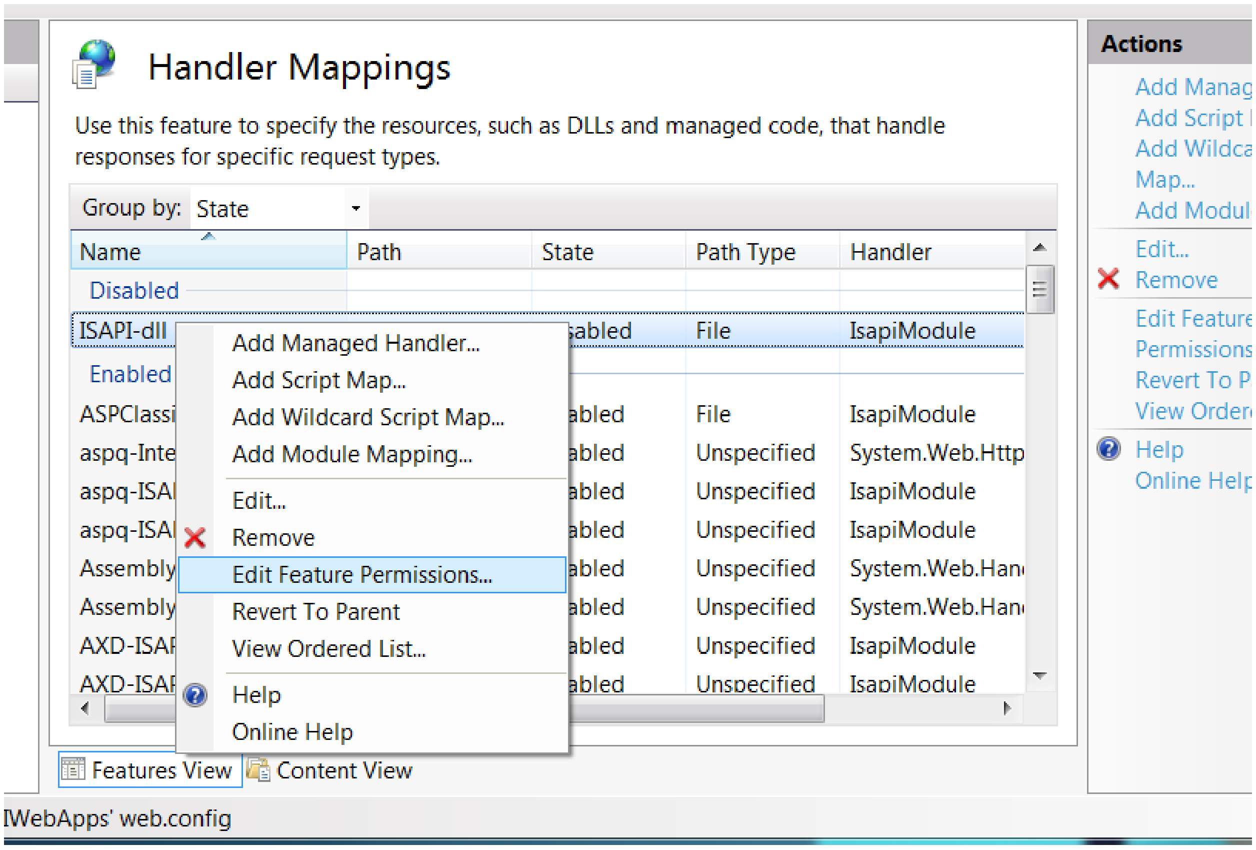

Internet Information Services (IIS) Manager

DIZADA-1 ▸ Sites ▸ IQMSWEBSITE ▸ uniGUIWebApps ▸

Connections
- DIZADA-1 (IQMS\dizada)
  - Application Pools
  - Sites
    - IQMSWEBSITE
      - aspnet_client
      - EnterpriseIQ
      - Mobile
      - RTCharts
      - WebAPI
      - WebDirect
      - WebIQClock
      - uniGUIWebApps

/uniGUIWebApps Home

Filter: Go Show All Group by: Area



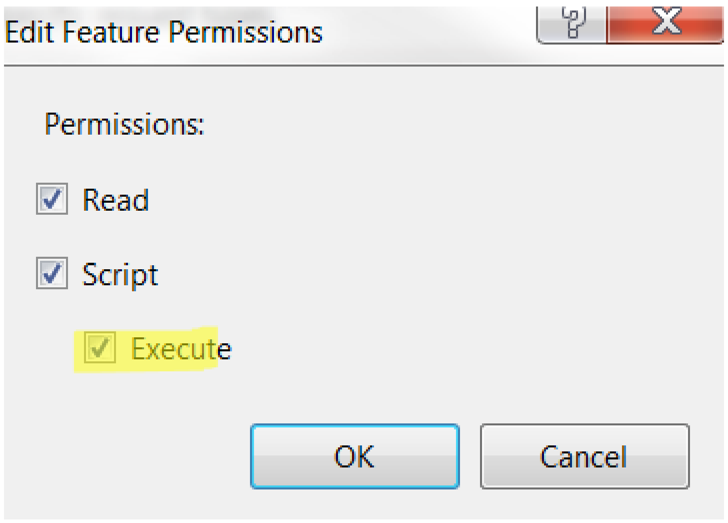
Handler Mappings

Use this feature to specify the resources, such as DLLs and managed code, that handle responses for specific request types.

Group by: State

Name	Path	State	Path Type	Handler
Disabled				
ISAPI-dll		abled	File	IsapiModule
Enabled				
ASPClassi		abled	File	IsapiModule
aspq-Inte		abled	Unspecified	System.Web.Http
aspq-ISA		abled	Unspecified	IsapiModule
aspq-ISA		abled	Unspecified	IsapiModule
Assembly		abled	Unspecified	System.Web.Han
Assembly		abled	Unspecified	System.Web.Han
AXD-ISAP		abled	Unspecified	IsapiModule
AXD-ISAP		abled	Unspecified	IsapiModule

IWebApps' web.config



Edit Feature Permissions

Permissions:

[✓] Read

[✓] Script

[✓] Execute

Create and give permissions to a few directories

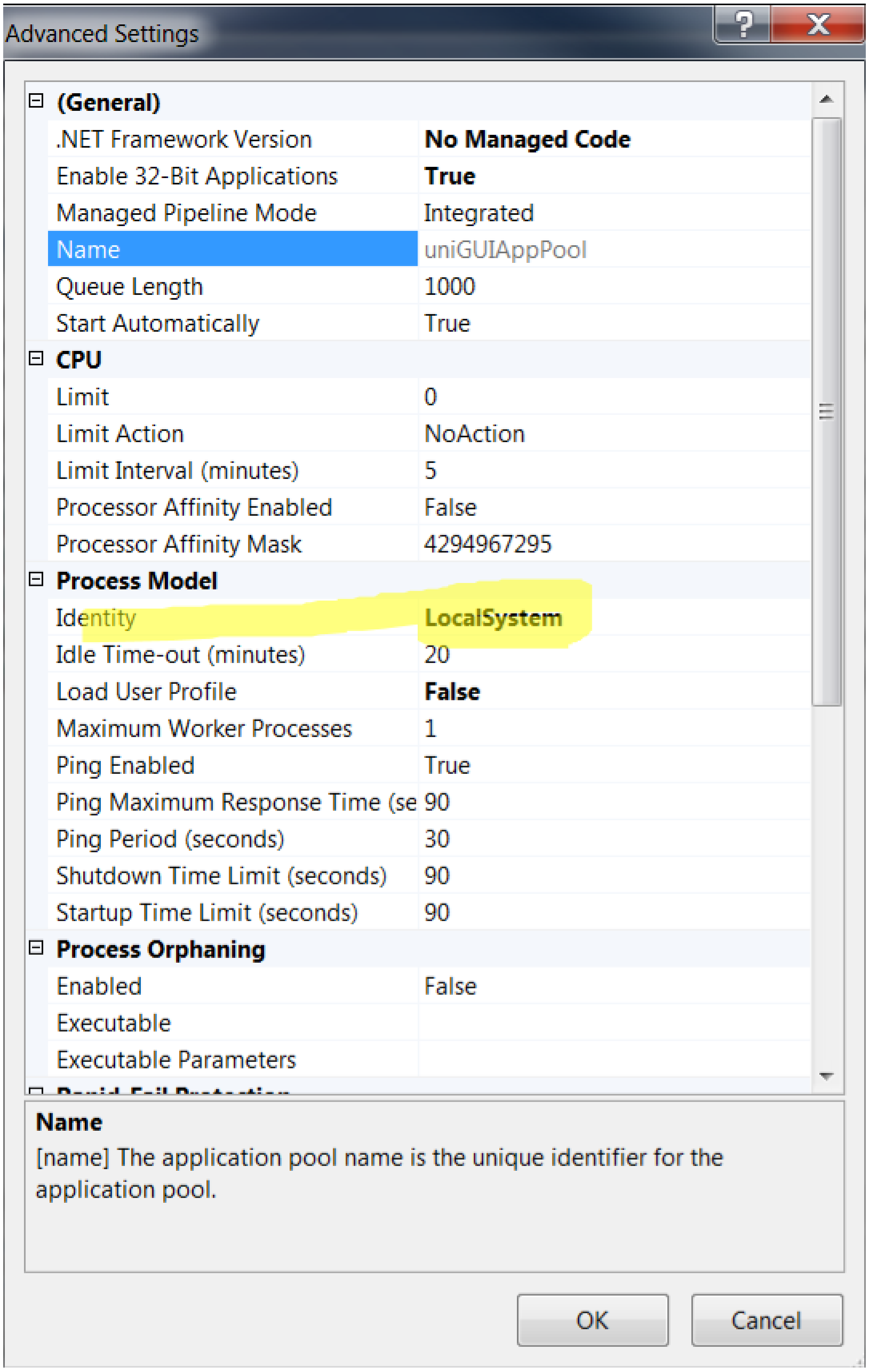
Finally, confirm that Sencha Ext Js OEM is located in the same directory defined in the ServerModule. For example, if all the Web Modules created with uniGUI are installed in a directory like **C:\IQMS\WEBIIS\uniGUIWebApps** we could copy the OEM Sencha files from <InstallFolder>\FMSoft\Framework\uniGUI\ext<version> to C:\IQMS\WEBIIS\uniGUIWebApps\ext.

The last step is to create several folders in the location specified in the ServerModule, which by default will be where the application is deployed, C:\IQMS\WEBIIS\uniGUIWebApps, and give enough rights for the application to access them:

- cache – temporary cache folder
- files
- log
- temp

The standard permissions will not allow the modules to load the Oracle OCI library.

The correct configuration will assign some specific user to the uniGUIAppPool with enough rights to execute the ISAPI modules which will be deployed to C:\IQMS\WEBIIS\uniGUIWebApps. The simplest solution, for developers, is just to assign **LocalSystem** as the identity for that application pool.



Advanced Settings

(General)	
.NET Framework Version	**No Managed Code**
Enable 32-Bit Applications	**True**
Managed Pipeline Mode	Integrated
Name	uniGUIAppPool
Queue Length	1000
Start Automatically	True
CPU	
Limit	0
Limit Action	NoAction
Limit Interval (minutes)	5
Processor Affinity Enabled	False
Processor Affinity Mask	4294967295
Process Model	
Identity	**LocalSystem**
Idle Time-out (minutes)	20
Load User Profile	**False**
Maximum Worker Processes	1
Ping Enabled	True
Ping Maximum Response Time (se	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90
Process Orphaning	
Enabled	False
Executable	
Executable Parameters	

Name
[name] The application pool name is the unique identifier for the application pool.

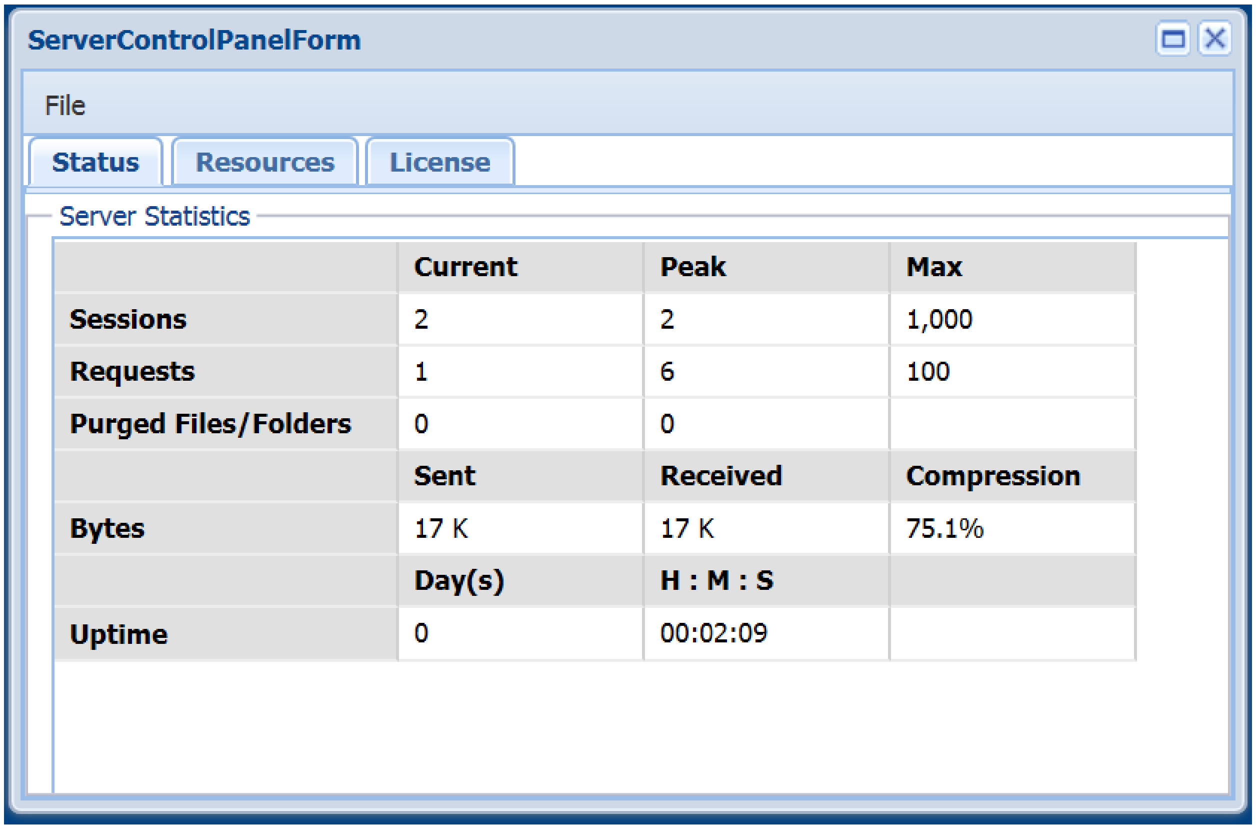
How to use the log files

The information written to the log is useful for detecting errors in the configuration and reasons for application failure. For example, this is part of that log:

```
FirstApp: 00000D14: 14:26:41 []:Starting Server. Module Handle: 0000000001CF0000

FirstApp: 00000D14: 14:26:41 [TUniServerModule]:Server First Init.

FirstApp: 00000D14: 14:26:41 [TUniServerModule]:Erasing Cache Folder...

FirstApp: 00000D14: 14:26:41 [TUniServerModule]:Cache Folder Erased. <0> Files deleted.

FirstApp: 00000D14: 14:26:41 [TUniServerModule]:HTTP Server Started.

FirstApp: 00000D14: 14:26:41 []:Server Started. Module Handle: 0000000001CF0000
```

Advantages of deploying the Web Application as an ISAPI Module

1. Current Standalone Server requires a session in the server, which is perfect for developers, but not for production.
2. Windows Service Web Application project type is the same as a Standalone Server, but running as a service. It is the best solution for small deployments (not too many remote clients).
3. An ISAPI module runs under Internet Information Server, which itself runs as a Windows Service.
4. Performance is higher using Internet Information Server hosting the ISAPI Modules where there are many simultaneous clients.
5. Applications pools, identities, permissions, management of the Web Application is done with IIS.
6. We keep the ability to monitor the application using a URL like
   a. http://localhost:8080/uniGUIWebApps/PoC1.dll/server
7. There is a new project in development which adds Load Balancing to the Standalone Server / Windows Service. It will add redundancy and better performance without requiring IIS.



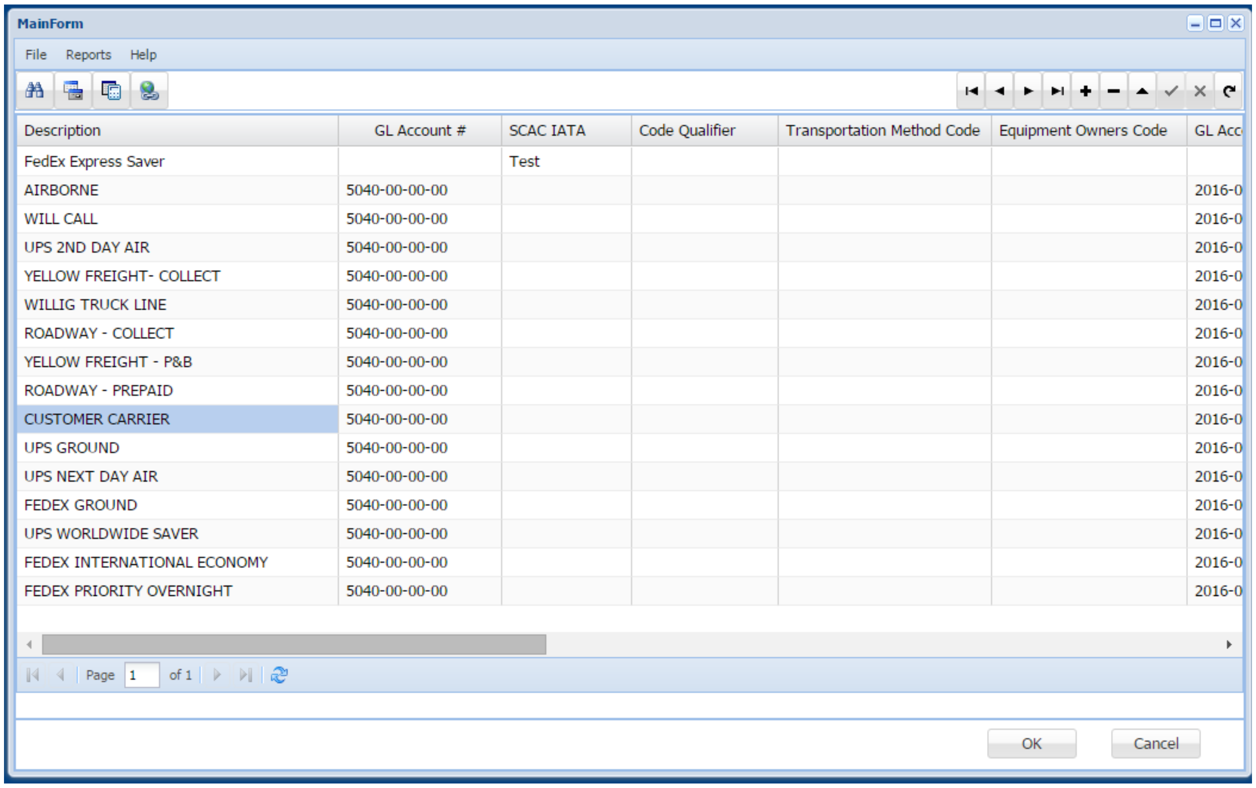
	Current	Peak	Max
Sessions	2	2	1,000
Requests	1	6	100
Purged Files/Folders	0	0	
	Sent	Received	Compression
Bytes	17 K	17 K	75.1%
	Day(s)	H : M : S	
Uptime	0	00:02:09	

Ship Via Maintenance

This proof-of-concept will look for options to migrate Ship Via Maintenance (FREIGHT) from Delphi VCL desktop and BDE to Delphi ISAPI Web Module using uniGUI.

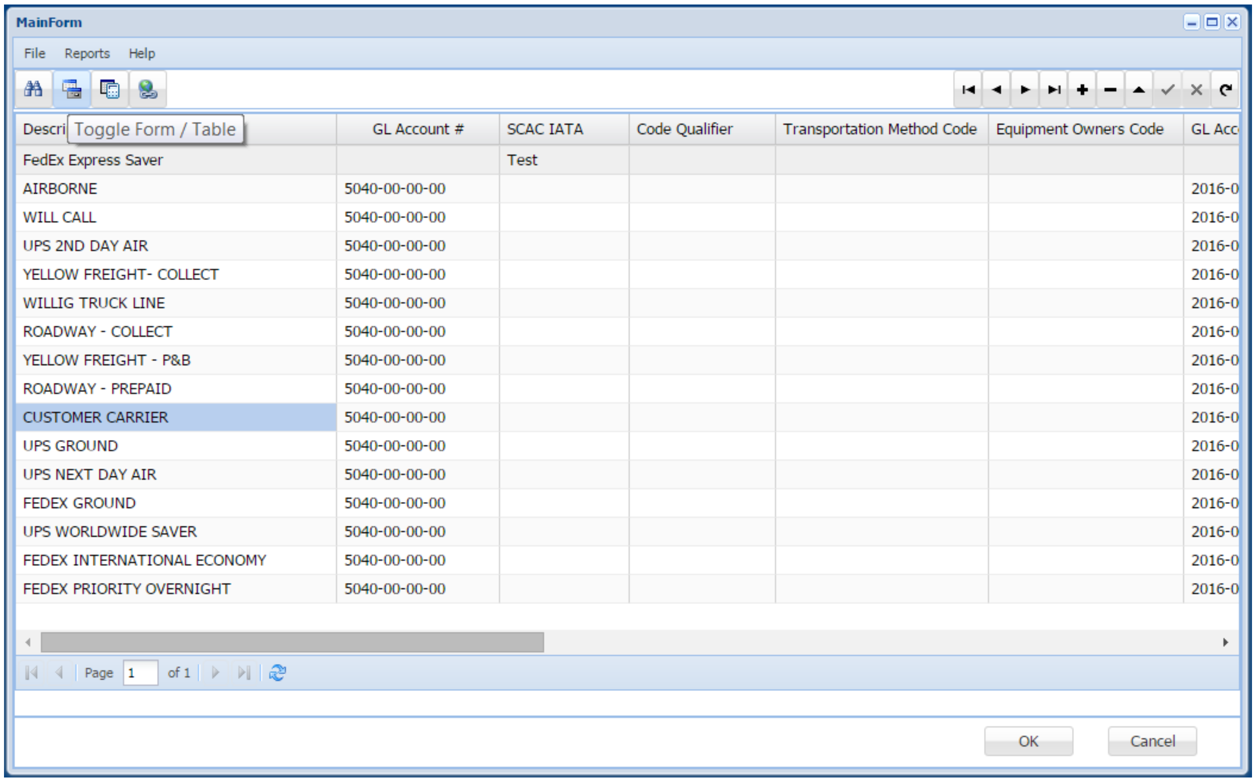
In many ways, this is a typical module in EnterpriseIQ:

- Main menu
- A top bar with speed buttons and DB navigator
- A bottom panel with OK & Cancel buttons
- A Page Control with two tabs
    - Form view using DB components and some dialogs
    - Table view using a DB Grid
- Some of the speed buttons and form components could pop up another form, sometimes modal.



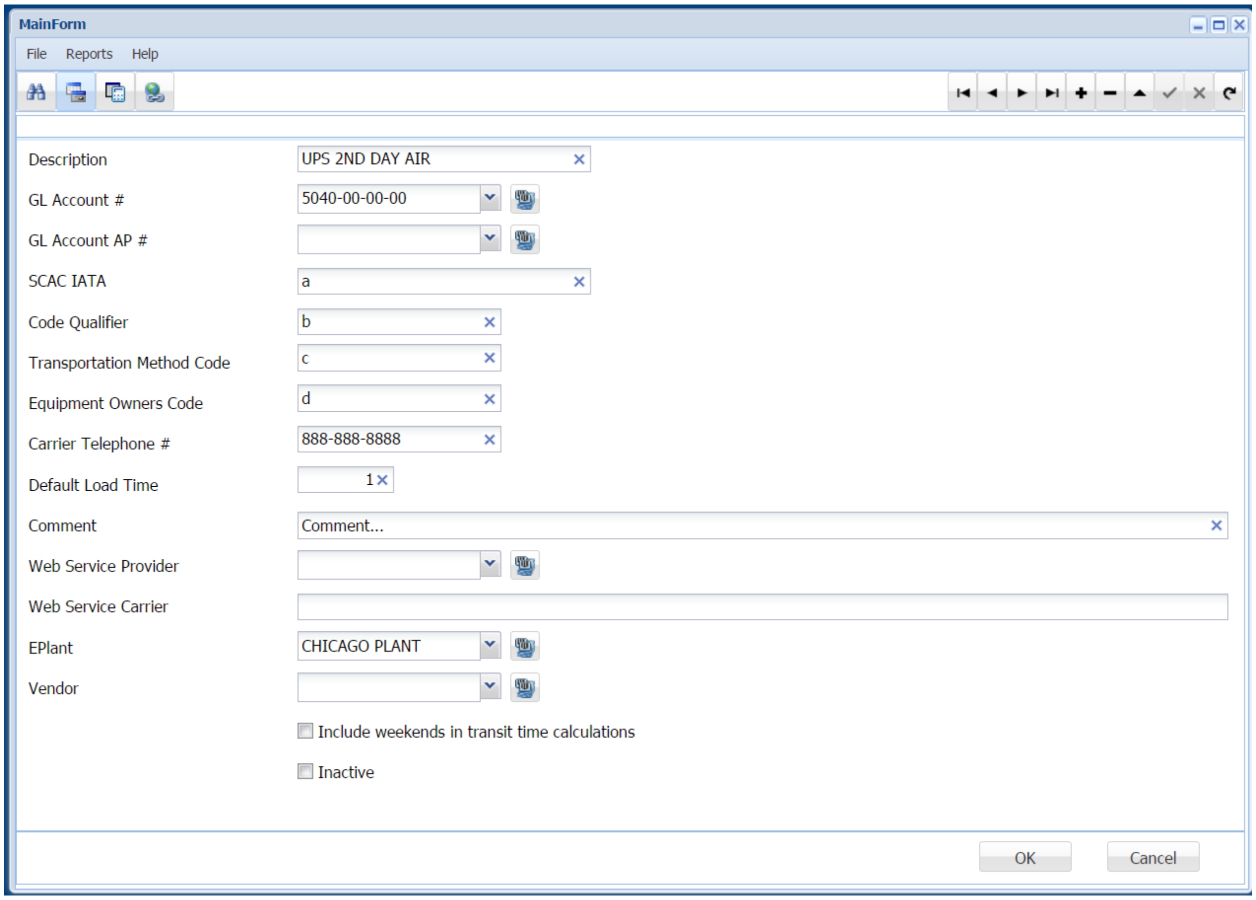
Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipment Owners Code	GL Acc
FedEx Express Saver		Test				
AIRBORNE	5040-00-00-00					2016-0
WILL CALL	5040-00-00-00					2016-0
UPS 2ND DAY AIR	5040-00-00-00					2016-0
YELLOW FREIGHT- COLLECT	5040-00-00-00					2016-0
WILLIG TRUCK LINE	5040-00-00-00					2016-0
ROADWAY - COLLECT	5040-00-00-00					2016-0
YELLOW FREIGHT - P&B	5040-00-00-00					2016-0
ROADWAY - PREPAID	5040-00-00-00					2016-0
CUSTOMER CARRIER	5040-00-00-00					2016-0
UPS GROUND	5040-00-00-00					2016-0
UPS NEXT DAY AIR	5040-00-00-00					2016-0
FEDEX GROUND	5040-00-00-00					2016-0
UPS WORLDWIDE SAVER	5040-00-00-00					2016-0
FEDEX INTERNATIONAL ECONOMY	5040-00-00-00					2016-0
FEDEX PRIORITY OVERNIGHT	5040-00-00-00					2016-0

After using the speed button Toggle Form / Table



Descri	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipment Owners Code	GL Acc
FedEx Express Saver		Test				
AIRBORNE	5040-00-00-00					2016-0
WILL CALL	5040-00-00-00					2016-0
UPS 2ND DAY AIR	5040-00-00-00					2016-0
YELLOW FREIGHT- COLLECT	5040-00-00-00					2016-0
WILLIG TRUCK LINE	5040-00-00-00					2016-0
ROADWAY - COLLECT	5040-00-00-00					2016-0
YELLOW FREIGHT - P&B	5040-00-00-00					2016-0
ROADWAY - PREPAID	5040-00-00-00					2016-0
CUSTOMER CARRIER	5040-00-00-00					2016-0
UPS GROUND	5040-00-00-00					2016-0
UPS NEXT DAY AIR	5040-00-00-00					2016-0
FEDEX GROUND	5040-00-00-00					2016-0
UPS WORLDWIDE SAVER	5040-00-00-00					2016-0
FEDEX INTERNATIONAL ECONOMY	5040-00-00-00					2016-0
FEDEX PRIORITY OVERNIGHT	5040-00-00-00					2016-0

Page Control shows the tab TabForm.

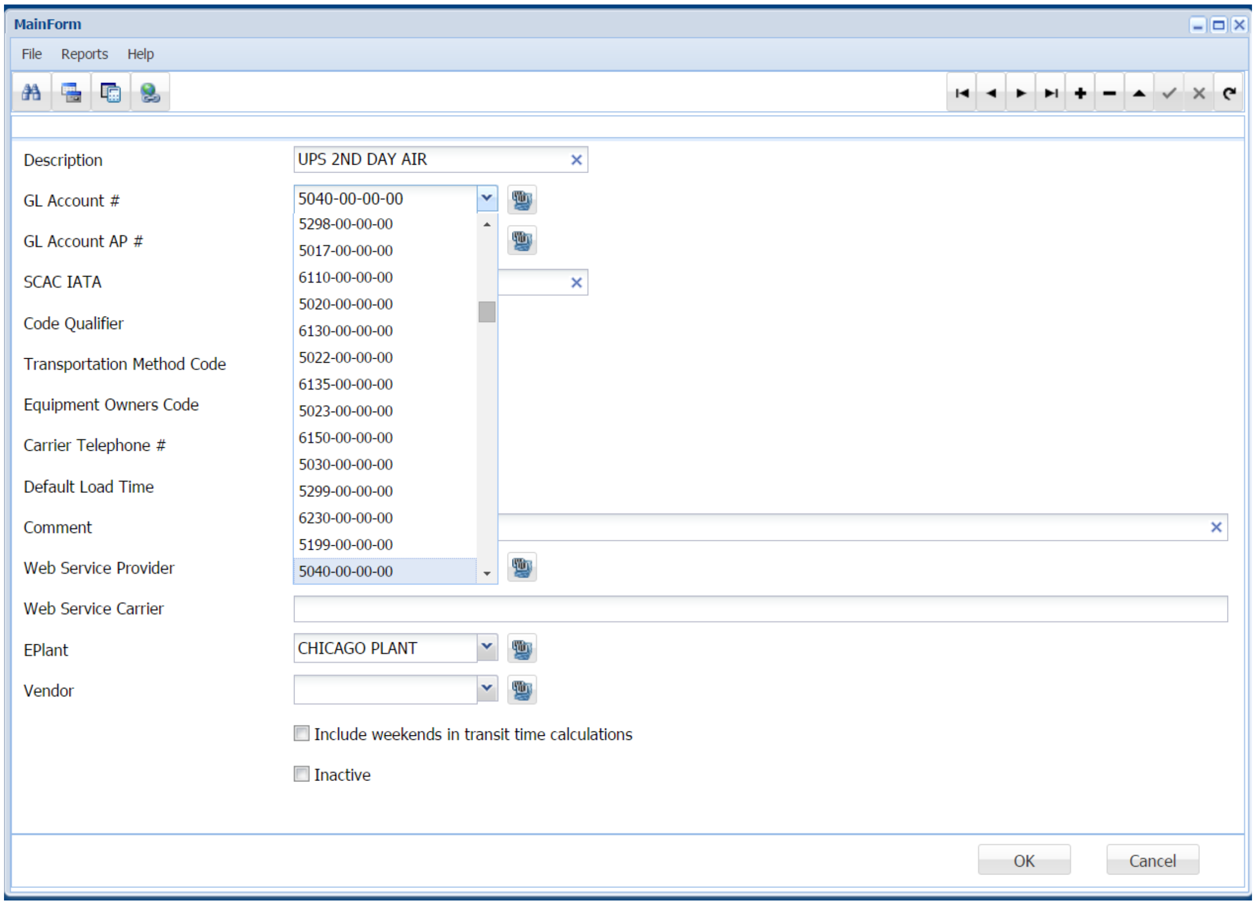


Description	UPS 2ND DAY AIR
GL Account #	5040-00-00-00
GL Account AP #	
SCAC IATA	a
Code Qualifier	b
Transportation Method Code	c
Equipment Owners Code	d
Carrier Telephone #	888-888-8888
Default Load Time	1
Comment	Comment...
Web Service Provider	
Web Service Carrier	
EPlant	CHICAGO PLANT
Vendor	

☐ Include weekends in transit time calculations

☐ Inactive

GL Accounts and other fields are using a standard DBLookupComboBox in addition to a button which will launch a Pick List. Any current selection can be cleared by pressing the DELETE key.



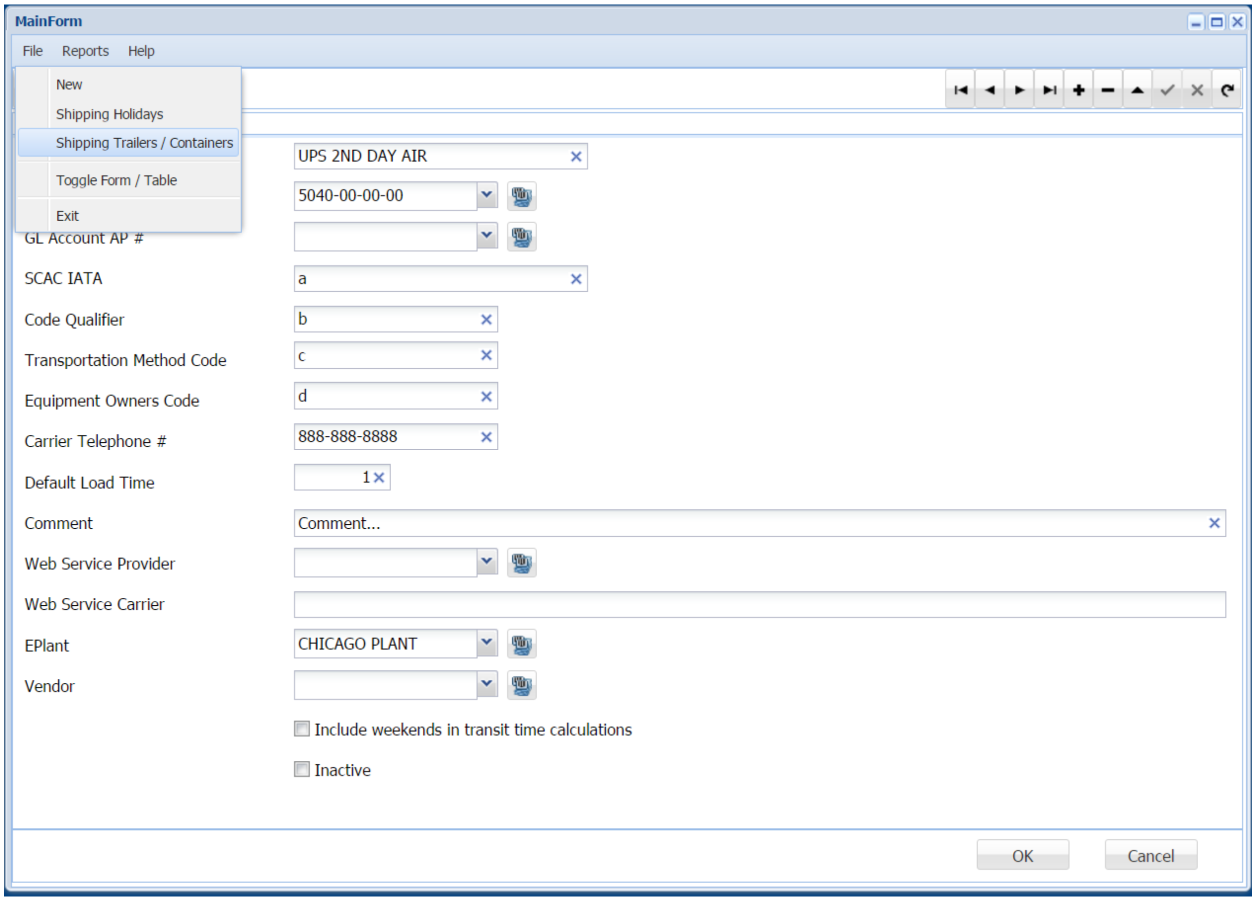
Description	UPS 2ND DAY AIR
GL Account #	5040-00-00-00
GL Account AP #	
SCAC IATA	
Code Qualifier	
Transportation Method Code	
Equipment Owners Code	
Carrier Telephone #	
Default Load Time	
Comment	
Web Service Provider	
Web Service Carrier	
EPlant	CHICAGO PLANT
Vendor	

```
5298-00-00-00
5017-00-00-00
6110-00-00-00
5020-00-00-00
6130-00-00-00
5022-00-00-00
6135-00-00-00
5023-00-00-00
6150-00-00-00
5030-00-00-00
5299-00-00-00
6230-00-00-00
5199-00-00-00
5040-00-00-00
```

☐ Include weekends in transit time calculations

☐ Inactive

The main menu works exactly like the standard Delphi component.

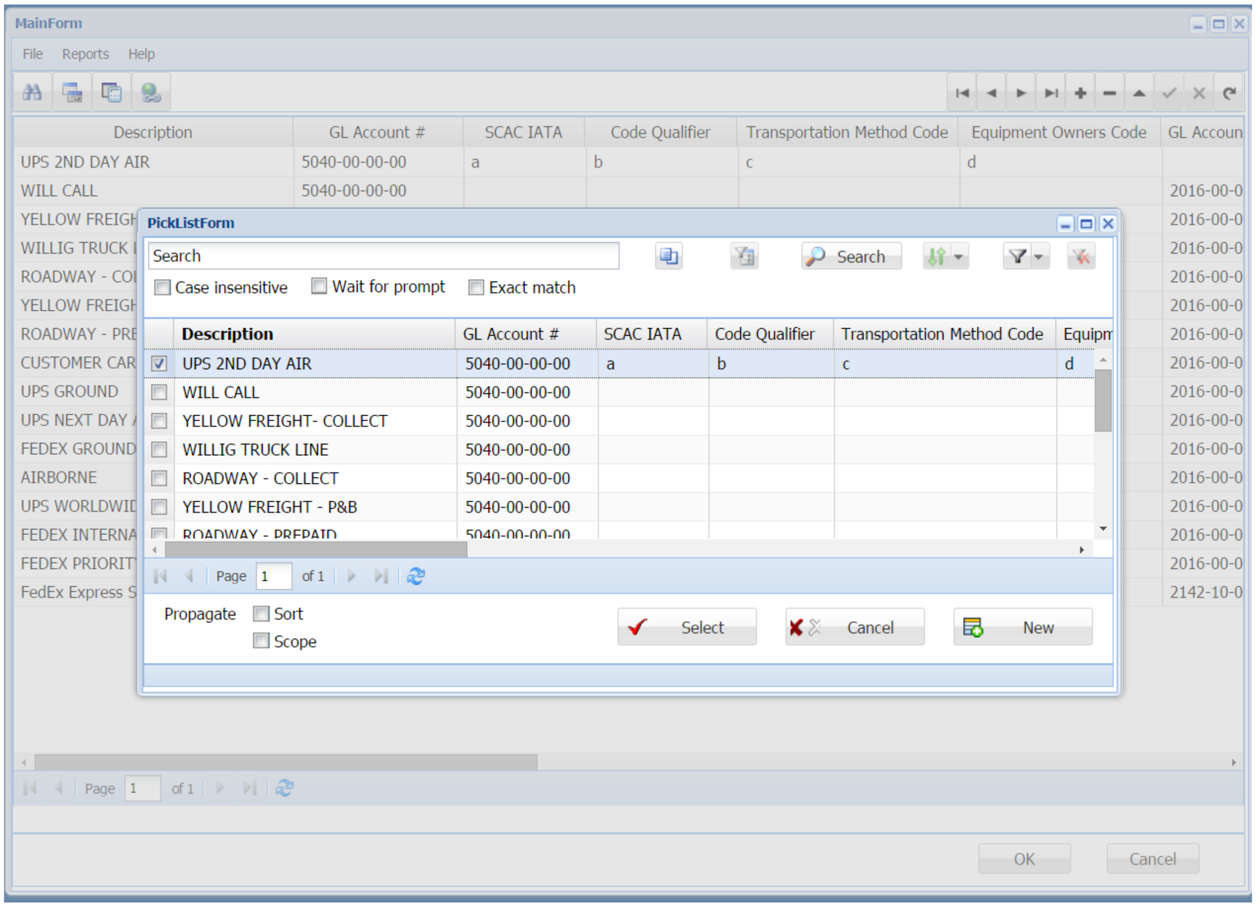


	UPS 2ND DAY AIR
	5040-00-00-00
GL Account AP #	
SCAC IATA	a
Code Qualifier	b
Transportation Method Code	c
Equipment Owners Code	d
Carrier Telephone #	888-888-8888
Default Load Time	1
Comment	Comment...
Web Service Provider	
Web Service Carrier	
EPlant	CHICAGO PLANT
Vendor	

☐ Include weekends in transit time calculations

☐ Inactive

After clicking the speed button Search, a Pick List is shown.


MainForm

Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipment Owners Code	GL Accoun
UPS 2ND DAY AIR	5040-00-00-00	a	b	c	d	
WILL CALL	5040-00-00-00					2016-00-0
YELLOW FREIGH						2016-00-0
WILLIG TRUCK I						2016-00-0
ROADWAY - COI						2016-00-0
YELLOW FREIGH						2016-00-0
ROADWAY - PRE						2016-00-0
CUSTOMER CAR						2016-00-0
UPS GROUND						2016-00-0
UPS NEXT DAY /						2016-00-0
FEDEX GROUND						2016-00-0
AIRBORNE						2016-00-0
UPS WORLDWIC						2016-00-0
FEDEX INTERNA						2016-00-0
FEDEX PRIORIT						2016-00-0
FedEx Express S						2142-10-0

PickListForm

Search

☐ Case insensitive ☐ Wait for prompt ☐ Exact match

	Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipm
☑	UPS 2ND DAY AIR	5040-00-00-00	a	b	c	d
☐	WILL CALL	5040-00-00-00				
☐	YELLOW FREIGHT- COLLECT	5040-00-00-00				
☐	WILLIG TRUCK LINE	5040-00-00-00				
☐	ROADWAY - COLLECT	5040-00-00-00				
☐	YELLOW FREIGHT - P&B	5040-00-00-00				
☐	ROADWAY - PREPAID	5040-00-00-00				

Page 1 of 1

Propagate ☐ Sort ☐ Scope

Exiting the application with File | Exit or Cancel will close the session. No validation is implemented for button OK.

A few advanced techniques supporting a migration

In addition to using plain Delphi for migrating our application to the Web, we need to migrate first our core framework, our set of non-visual and visual components used extensively throughout all EnterpriseIQ.

Here we will show how to build a component package IQWebVCL with non-visual and visual components (a basic Pick List and the first draft of a User Defined Form).

Non-Visual component TIQWebHPick (Pick Lists generic implementation)

Pick Lists are used everywhere in EnterpriseIQ, but the initial implementation was based on BDE.

Basically, a pick list is a TQuery with some additional properties, and the Execute method for activating the form which will allow the operator to filter the query and select one or multiple rows.

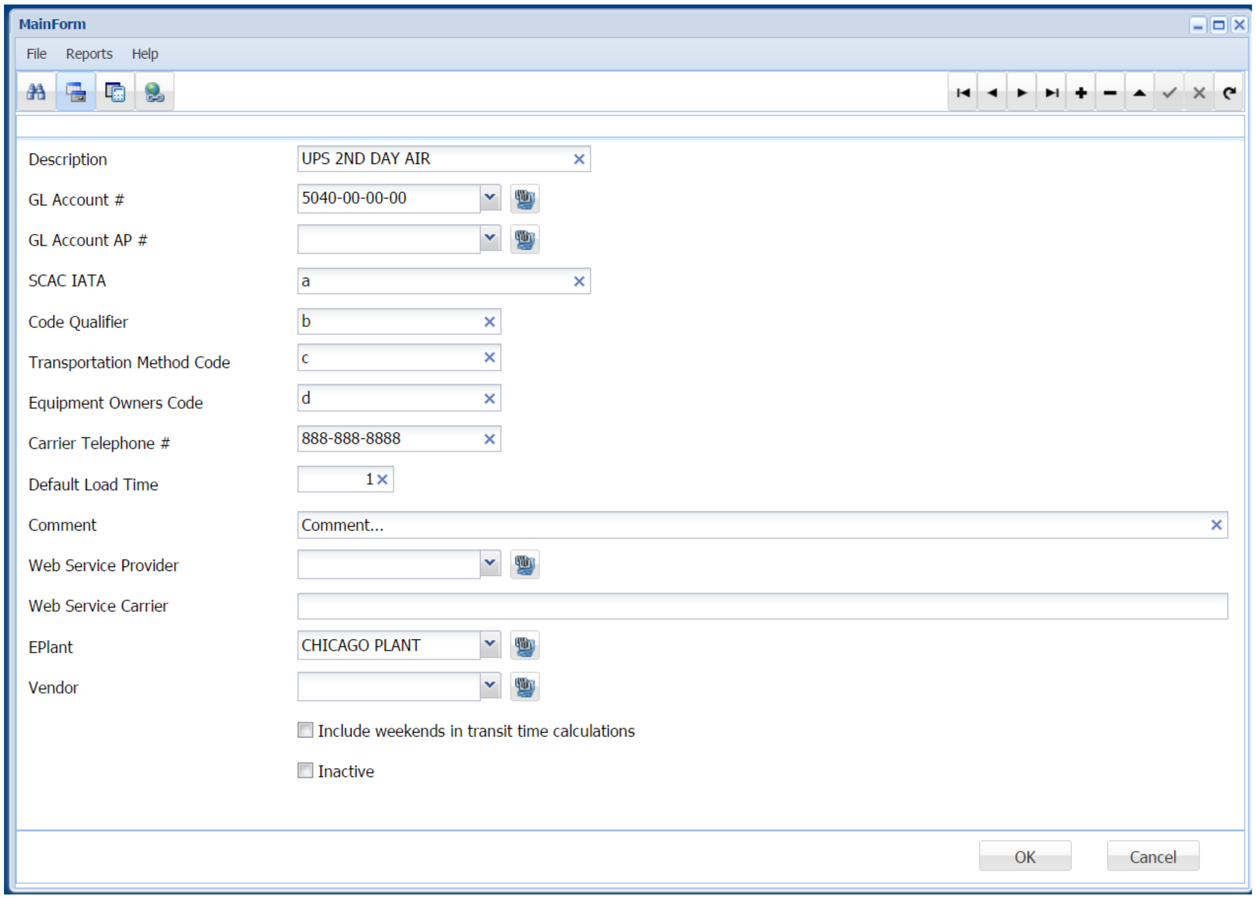
First, we need to implement the Pick List Form (PickListDlg) which will be activated by button OnClick event by the user for picking one or several rows from the dataset.

Instead of using a TQuery, I'll try to use a TDataset, ignoring the specific technology used for connecting to the database. Internally, however, a memory dataset will be used (TFDMemTable).

Main Form

TUniDBGrid allows using row editors, but we are providing the Toggle Grid / Form button so that any modification should happen in Form view.

Switching to Form view will show the common data inspector panel. Current combo boxes will be replaced with some button which will use the same picklist dialog associated with the binoculars (Search).



Description	UPS 2ND DAY AIR
GL Account #	5040-00-00-00
GL Account AP #	
SCAC IATA	a
Code Qualifier	b
Transportation Method Code	c
Equipment Owners Code	d
Carrier Telephone #	888-888-8888
Default Load Time	1
Comment	Comment...
Web Service Provider	
Web Service Carrier	
EPlant	CHICAGO PLANT
Vendor	

☐ Include weekends in transit time calculations

☐ Inactive

Pick List Dialog

This is a much simpler version of the Pick List Dialog used in EnterpriseIQ.

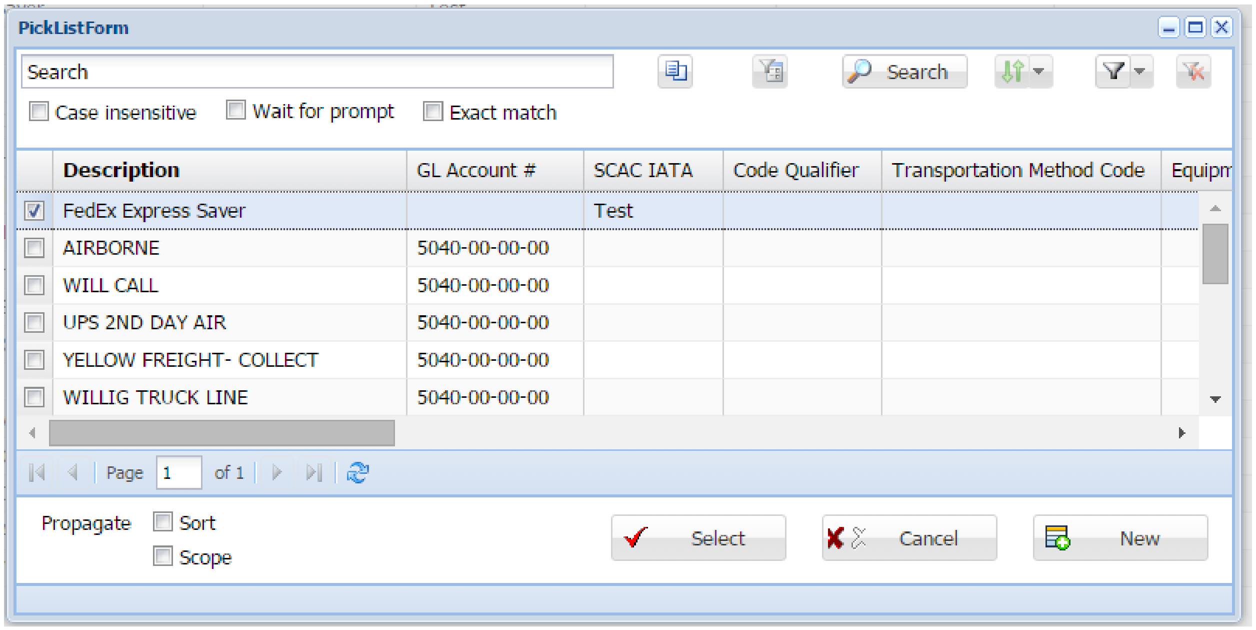
As a proof-of-concept, the only implemented features are:

1. Select active column (will have the title in bold)
2. Search
   a. In the selected column (bold)
   b. Takes into account “Case insensitive” and “Exact match” check boxes
3. Filter
   a. In the selected column (bold)
   b. Takes into account “Case insensitive” and “Exact match” check boxes
   c. Enables “Remove Filter” button
4. Remove Filter
   a. Clear the filter and disables itself
5. Sorting
   a. Using the column drop down menu allows to sort any column in ascending or descending order
   b. Takes into account “Case insensitive”

6. Selection checkboxes
    a. Enabled for selecting only one row (it is designed to be used for multiple selections)

User Interface

All the information for naming the columns, including the width, is taken from the original data source (meaning that it must be defined to get a correct render). In this case, the pick list is using the same query used in the main form, but using a cloned cursor (sorting and filtering will not affect what is already shown in the main form grid). Most of the time a pick list will use a subset of the original query (a different query).

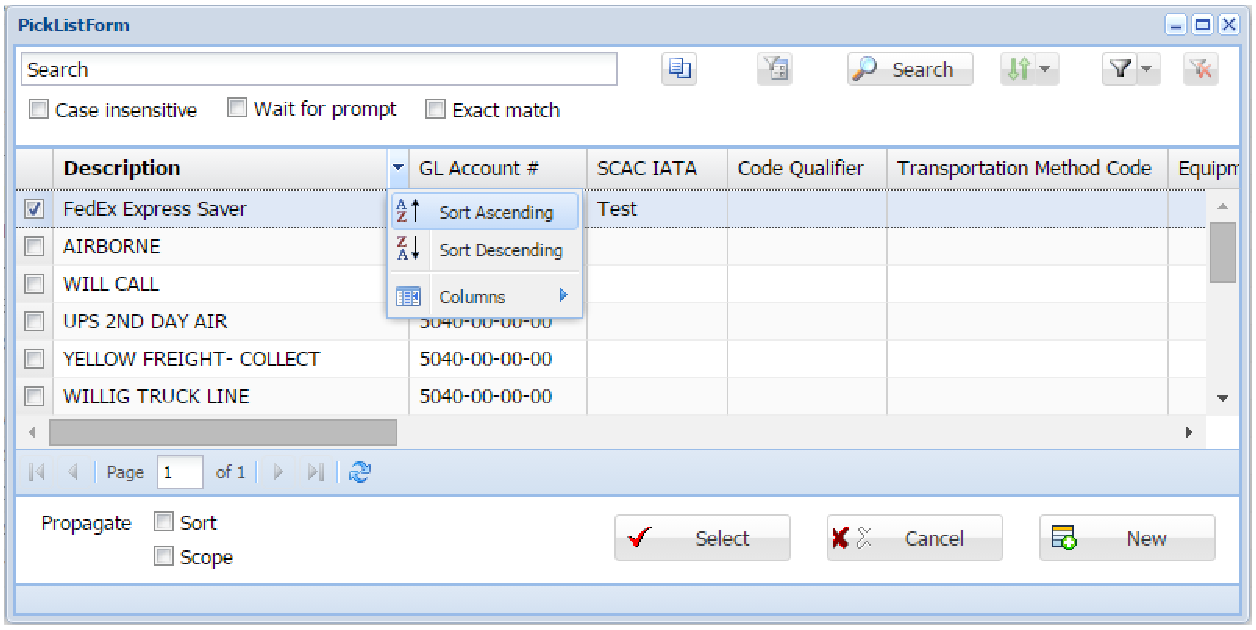


	Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipm
✓	FedEx Express Saver		Test			
☐	AIRBORNE	5040-00-00-00				
☐	WILL CALL	5040-00-00-00				
☐	UPS 2ND DAY AIR	5040-00-00-00				
☐	YELLOW FREIGHT- COLLECT	5040-00-00-00				
☐	WILLIG TRUCK LINE	5040-00-00-00				

Sorting

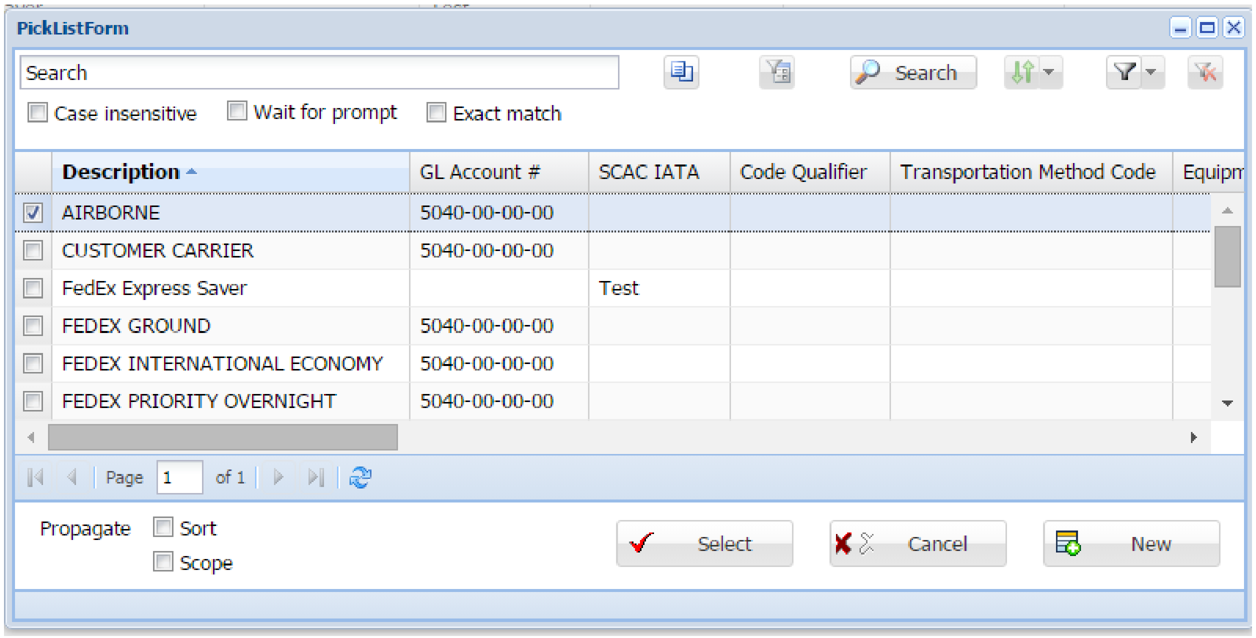
The current implementation uses IndexNames like 'DESCRIP:AN' (sort by DESCRIP in ascending order ignoring case).

It allows for multiple fields sorting (it will be implemented by adding a dialog as in current Pick List).



	Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipm
✓	FedEx Express Saver		Test			
☐	AIRBORNE					
☐	WILL CALL					
☐	UPS 2ND DAY AIR	5040-00-00-00				
☐	YELLOW FREIGHT- COLLECT	5040-00-00-00				
☐	WILLIG TRUCK LINE	5040-00-00-00				

Asking for ascending sorting returns a grid sorted by Description.


PickListForm

Search

☐ Case insensitive ☐ Wait for prompt ☐ Exact match

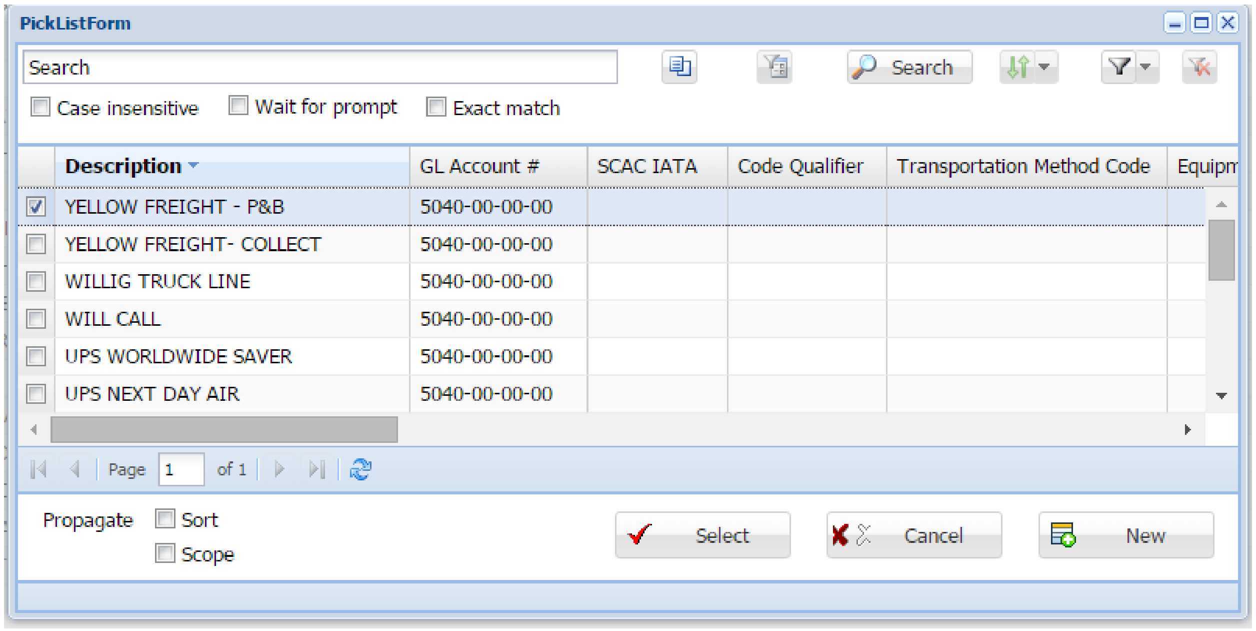
	Description ▲	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipm
☑	AIRBORNE	5040-00-00-00				
☐	CUSTOMER CARRIER	5040-00-00-00				
☐	FedEx Express Saver		Test			
☐	FEDEX GROUND	5040-00-00-00				
☐	FEDEX INTERNATIONAL ECONOMY	5040-00-00-00				
☐	FEDEX PRIORITY OVERNIGHT	5040-00-00-00				

Page 1 of 1

Propagate ☐ Sort ☐ Scope

Select Cancel New

Asking for descending order.


PickListForm

Search

☐ Case insensitive ☐ Wait for prompt ☐ Exact match

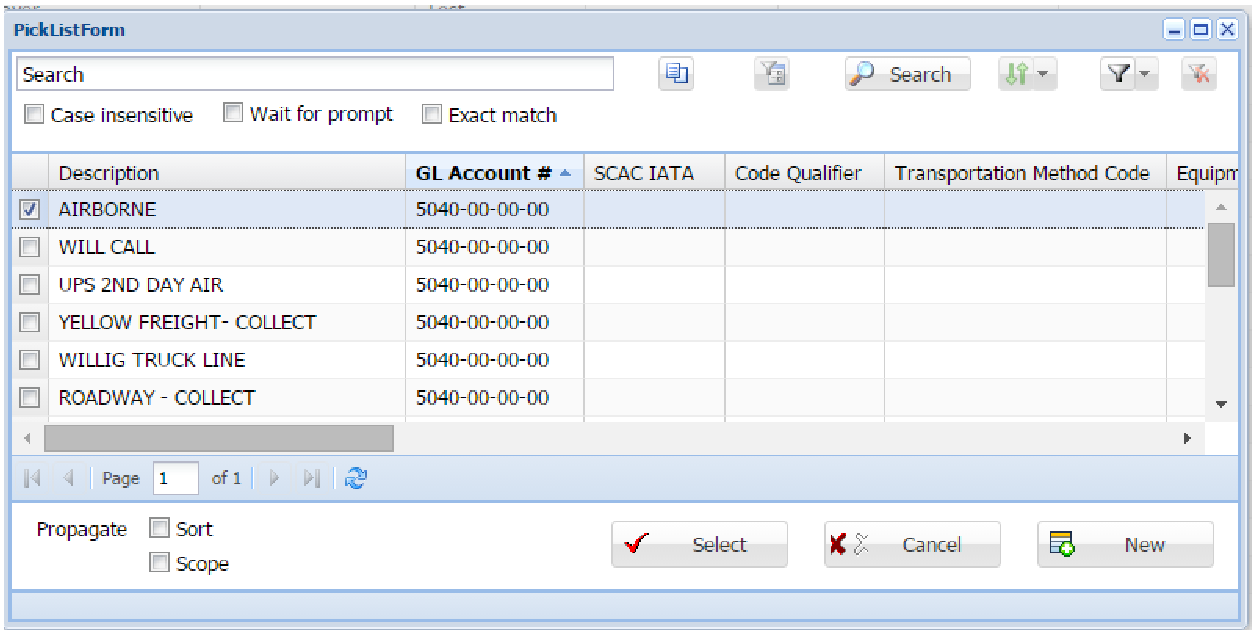
	Description ▼	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipm
☑	YELLOW FREIGHT - P&B	5040-00-00-00				
☐	YELLOW FREIGHT- COLLECT	5040-00-00-00				
☐	WILLIG TRUCK LINE	5040-00-00-00				
☐	WILL CALL	5040-00-00-00				
☐	UPS WORLDWIDE SAVER	5040-00-00-00				
☐	UPS NEXT DAY AIR	5040-00-00-00				

Page 1 of 1

Propagate ☐ Sort ☐ Scope

Select Cancel New

Setting the active column


PickListForm

Search

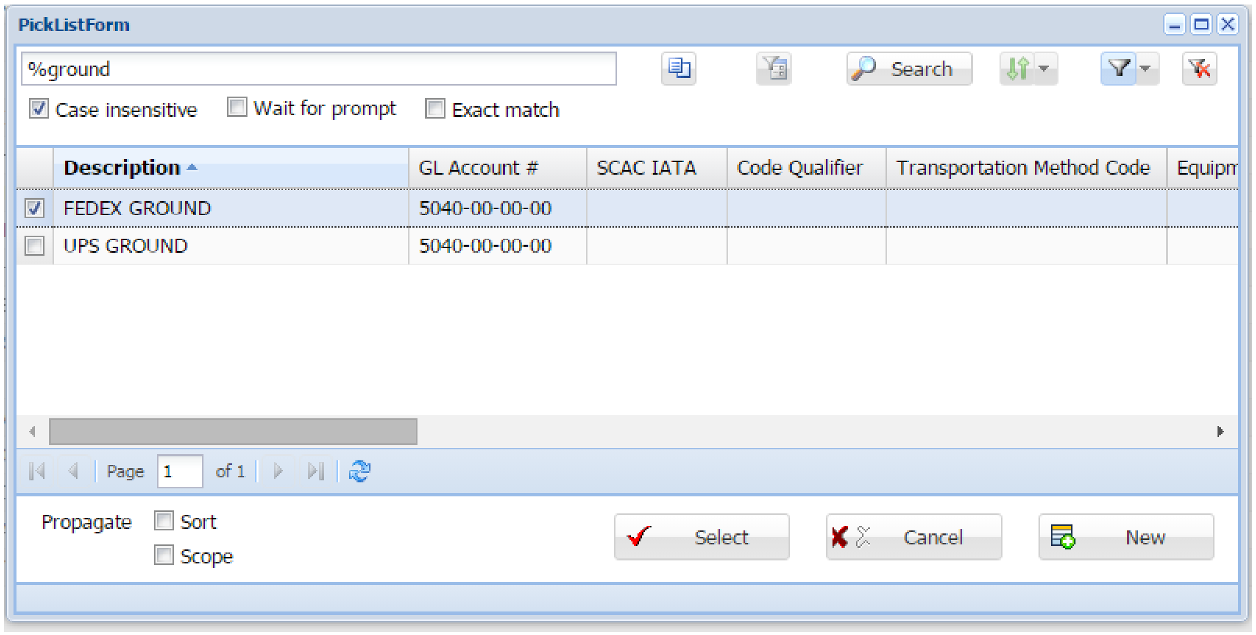
☐ Case insensitive ☐ Wait for prompt ☐ Exact match

	Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipm
✓	AIRBORNE	5040-00-00-00				
☐	WILL CALL	5040-00-00-00				
☐	UPS 2ND DAY AIR	5040-00-00-00				
☐	YELLOW FREIGHT- COLLECT	5040-00-00-00				
☐	WILLIG TRUCK LINE	5040-00-00-00				
☐	ROADWAY - COLLECT	5040-00-00-00				

Page 1 of 1

Propagate ☐ Sort ☐ Scope

Filtering


PickListForm

%ground

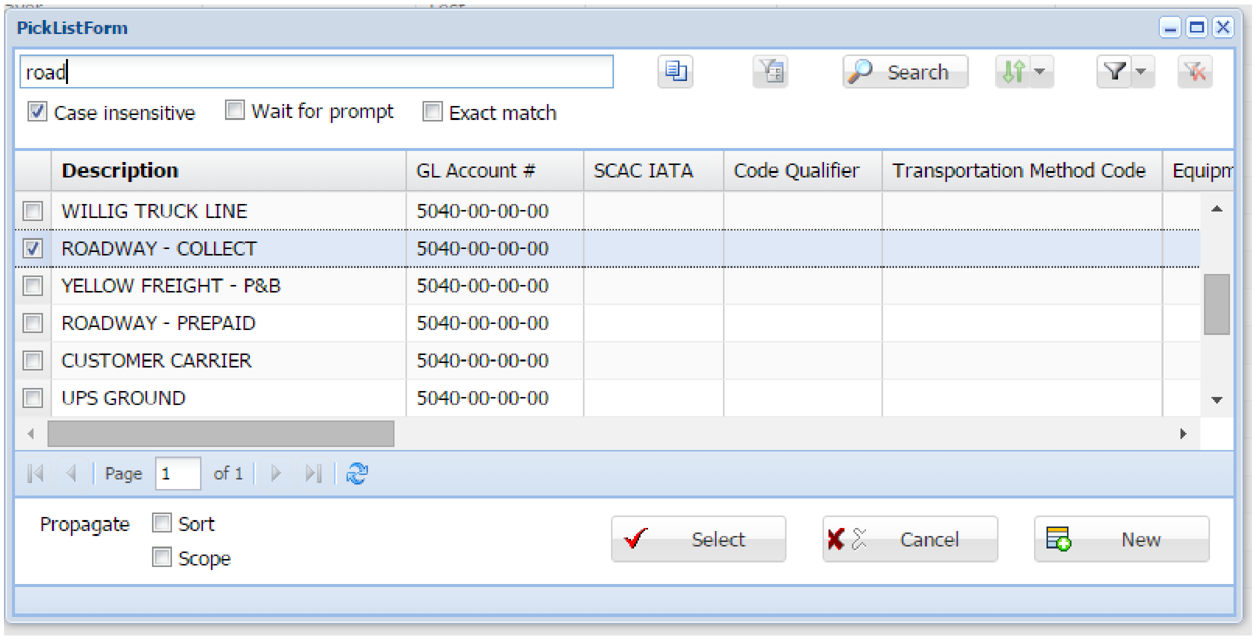
✓ Case insensitive ☐ Wait for prompt ☐ Exact match

	Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipm
✓	FEDEX GROUND	5040-00-00-00				
☐	UPS GROUND	5040-00-00-00				

Page 1 of 1

Propagate ☐ Sort ☐ Scope

Searching


PickListForm

road

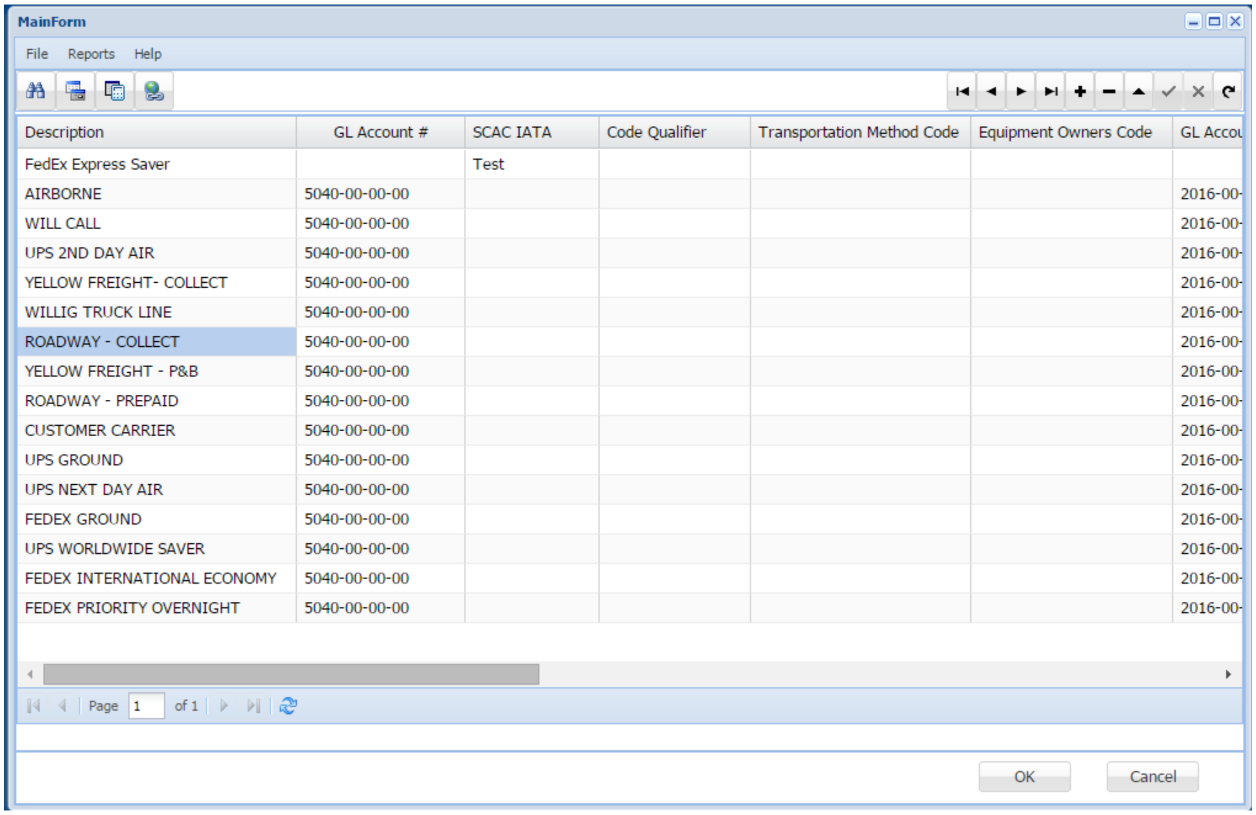
[✓] Case insensitive [ ] Wait for prompt [ ] Exact match

	Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipm
[ ]	WILLIG TRUCK LINE	5040-00-00-00				
[✓]	ROADWAY - COLLECT	5040-00-00-00				
[ ]	YELLOW FREIGHT - P&B	5040-00-00-00				
[ ]	ROADWAY - PREPAID	5040-00-00-00				
[ ]	CUSTOMER CARRIER	5040-00-00-00				
[ ]	UPS GROUND	5040-00-00-00				

Page 1 of 1

Propagate [ ] Sort [ ] Scope

After pressing “Select”, the correct row is selected in the main grid. As the pick dialog uses a cloned cursor, that assignment happens in the main form callback.


MainForm

Description	GL Account #	SCAC IATA	Code Qualifier	Transportation Method Code	Equipment Owners Code	GL Accou
FedEx Express Saver		Test				
AIRBORNE	5040-00-00-00					2016-00-
WILL CALL	5040-00-00-00					2016-00-
UPS 2ND DAY AIR	5040-00-00-00					2016-00-
YELLOW FREIGHT- COLLECT	5040-00-00-00					2016-00-
WILLIG TRUCK LINE	5040-00-00-00					2016-00-
ROADWAY - COLLECT	5040-00-00-00					2016-00-
YELLOW FREIGHT - P&B	5040-00-00-00					2016-00-
ROADWAY - PREPAID	5040-00-00-00					2016-00-
CUSTOMER CARRIER	5040-00-00-00					2016-00-
UPS GROUND	5040-00-00-00					2016-00-
UPS NEXT DAY AIR	5040-00-00-00					2016-00-
FEDEX GROUND	5040-00-00-00					2016-00-
UPS WORLDWIDE SAVER	5040-00-00-00					2016-00-
FEDEX INTERNATIONAL ECONOMY	5040-00-00-00					2016-00-
FEDEX PRIORITY OVERNIGHT	5040-00-00-00					2016-00-

Page 1 of 1

Implementation details for the Pick List

The pick list itself will be implemented as a simple component with two properties:

- Title

- Dataset

Two methods will allow to request a single or multiple selection from the dataset:

- DoSinglePickList
- DoMultiplePickList

This unit will be just a front end, the visible face of the Pick List, it is what the developer will need to use. Let's see the implementation and how the typical use is.

```
unit IQWebHPick;

interface

uses
  Classes,
  FireDAC.Stan.Intf,
  FireDAC.Stan.Option, FireDAC.Stan.Param, FireDAC.Stan.Error, FireDAC.DatS,
  FireDAC.Phys.Intf, FireDAC.DApt.Intf, FireDAC.Comp.DataSet,
  FireDAC.Comp.Client,
  Generics.Collections,
  PickListDlg,
  DB;

type

  TIQWebHPick = class(TComponent)
    protected
      FTitle    : string;
      FDataSet  : TFDDataSet;

      procedure SetTitle(const Value: string);
    public
      procedure DoSinglePickList  (aID : integer;        aCallBack:
TSinglePickListCallBack);
      procedure DoMultiplePickList(aIDs: TList<integer>; aCallBack:
TMultiplePickListCallBack);
    published

      property Title    : string      read FTitle   write SetTitle;
      property DataSet  : TFDDataSet  read FDataSet write FDataSet;
  end;

implementation

uses
  SysUtils,
  StrUtils;

procedure TIQWebHPick.DoSinglePickList(aID: integer; aCallBack:
TSinglePickListCallBack);
begin
  PickListDlg.DoSinglePickList(DataSet, aID, aCallBack);
end;

procedure TIQWebHPick.DoMultiplePickList(aIDs: TList<integer>; aCallBack:
TMultiplePickListCallBack);
```

```
begin
  PickListDlg.DoMultiplePickList(DataSet, aIDs, aCallBack);
end;

procedure TIQWebHPick.SetTitle(const Value: string);
var
  I: Integer;
begin
  FTitle:= Value;

  I:= Pos('picklist', LowerCase(FTitle));

  if I > 0 then
  begin
    if FTitle[ I ] = 'P' then  // ensure upper case when spelled Picklist ->
PickList
       FTitle[ I+4 ]:= 'L';
    FTitle:= StuffString( FTitle, I+4, 0, ' ' );
  end;
end;

end.
```

It is important to understand the asynchronous behavior of any communication like this. The calling form will request to pick the selection, but it will be called back from the PickListDlg when the user finishes. Let's see how it is done.

```
procedure TUniShipViaMaintenance.pickGLACCTClick(Sender: TObject);
var
  oldID: integer;
begin
  with SVM_DM do
    begin
      oldID := StrToIntDef(Trim(VarToStr( UniDBLookupComboBoxGLACCT.KeyValue
)), 0);

      IQWebHPickGLACCT.DoSinglePickList
      (
        oldID,
        (
          procedure (aResult: TModalResult; newID: integer)
          begin
            if (aResult = mrOK) and (newID <> oldID) then
              begin
                QryFreight.Edit;
                QryFreightGLACCT_ID_FREIGHT.Value := newID;
              end;
          end
        )
      );
    end;
end;
```

This is the event triggered by the pick list button used for selecting GL Account. Basically, we send the previous selection and provide an anonymous method (acting like an inline callback) which will execute when the selection is done. If the returned ID is different than the old ID, the current record is opened for edition and the value is updated.

By using anonymous methods, we preserve the context we need to access oldID which is a local variable to this procedure.

It won't be a surprise to see that this asynchronous behavior is present also in the back-end, the PickListDlg itself. Single selection is handled in this method:

```
procedure DoSinglePickList(aSrcDataset: TFDDataset; aID: integer; aCallBack:
TSinglePickListCallBack);
begin
  with
TPickListForm(TUniGUIMainModule(UniApplication.UniMainModule).GetFormInstance
(TPickListForm)) do
  begin
    memTable.CloneCursor(aSrcDataset, false, true);
    FSrcDataset := aSrcDataset;

    FormatGrid;
    UniDBGrid1.Options := UniDBGrid1.Options - [dgMultiSelect];

    if aID <> 0 then
      memTable.Locate('ID', aID, []);

    ShowModal
    (
      procedure (Sender: TComponent; AResult: Integer)
      var
        mr  : TModalResult;
        aID : integer;
      begin
        mr := TModalResult(AResult);
        if mr = mrOK then
          aID := trunc( memTable.FieldByName('ID').Value )
        else
          aID := 0;

        aCallBack(mr, aID);
      end
    );
  end;
end;
```

Here we also use one anonymous method for receiving the result from ShowModal and sending it back to our callback method.

Using Frames for creating visual controls

This feature is being available for quite some time in Delphi, but it is particularly useful when migrating or recreating an existing form with new controls. The original code is usually the result of exporting a visual design to code, but that original design is no longer available and it could require different parameters for the new controls. By using frames, it is possible to visually create the user interface and convert the frame later to a common visual control.

For this example, we will be using the User Defined Form (TIQUDEmbeddedForm).

Core shared components to support the migration

In order to provide a smooth migration path sharing a similar look-and-feel and functionality, it is necessary to develop a minimum set of shared components (in tab **IQWeb UniGUI**).

Login Form

As with current EnterpriseIQ, the entry point to EnterpriseIQWeb should be the login form. After being successfully identified, the user should be able to access the Module Launcher.

Module Launcher

The first version will be based on the All Features Demo from uniGUI, as it will allow to add any new module by physically creating a new folder and adding the corresponding module main unit with the same name. Some of the included features are:

- Collapsible module menu to the left
- Running modules as tabs in a page control (which can be closed on demand)

Common expected features for each module

Each new module should follow the same rules as the desktop application EnterpriseIQ:

- The main menu with most of the same options currently available
- A Data Module with data access components related to the module
    - Currently, there are data access components on several forms
    - The Data Module usually includes desktop-related visual interactions
- A toolbar with speed buttons and a DB Navigator
- The first speed buttons are usually Search (for locating a record using a Pick List) and Grid / Inspector toggle (for switching between the grid and the inspector for the currently selected row)
- Reusable components like:
    - User Defined Labels,
    - User Defined Fields,
    - User Defined Forms
    - Documents

In addition to these visual components, we also share many non-visual components and units which will need some degree of refactoring for avoiding references to the Windows desktop (or VCL components).

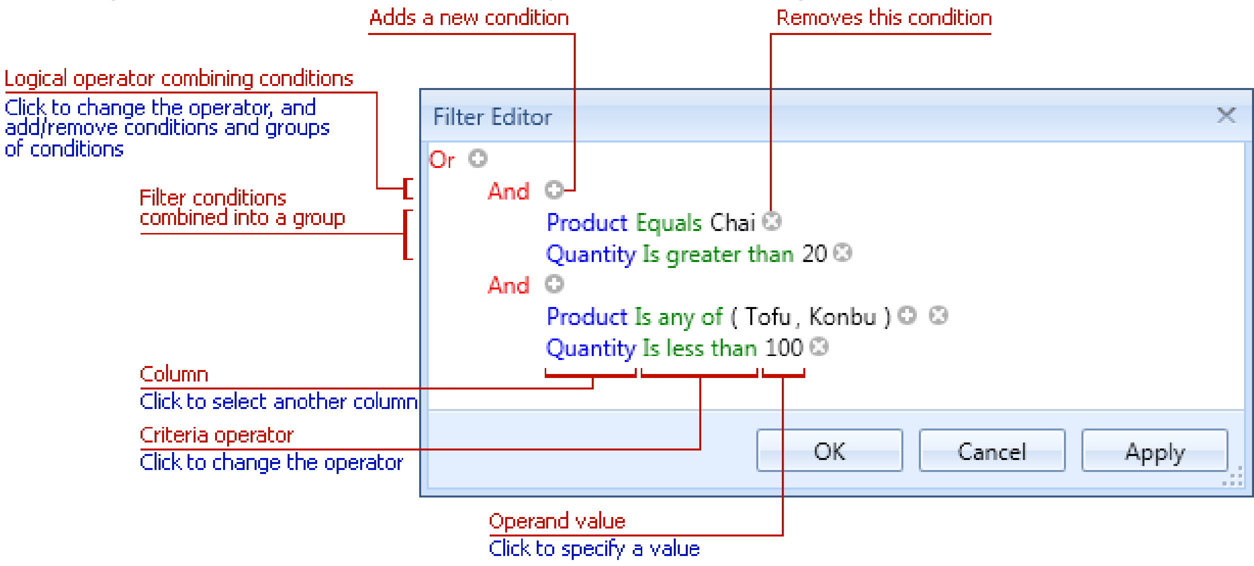
PickListDlg

In the current proof-of-concept application, the Pick List Dialog allows to select one or multiple records but is using an internal TFDMemTable which requires loading all the source dataset records at the beginning. It is not scalable. There are several decisions to take:

1. Are we going to support only FireDAC data access technology or any of the technologies currently used in EnterpriseIQ?
    a. If we only support FireDAC, we could avoid handling and modifying SQL code for sorting the database (by one or multiple fields) by using the property IndexNames.
    b. Properties Filter and Filtered are available in the standard TDataSet, meaning that both sorting and filtering could be available without modifying the underlying SQL text.
    c. One possible drawback is the fact that client-side paging is implemented by querying RecordCount in the server-side dataset. Depending on the performance, it could be better to implement a faster

way by overriding the method OnGetRecordCount. That being said, a Pick List is not supposed to pull many columns, so that the performance difference should be negligible. If needed, FireDAC provides a mode cmTotal for reporting the TFDQuery.RecordCount as the total count.

2. How to handle sorting by several columns?
    a. The best option is to show the common dialog with all the fields / columns to the left and the selected fields to the right, ordered by priority (allowing to drag and drop in that list or between left and right). All the columns in that selection should have the caption in bold to identify them.
    b. As a shortcut, it should be possible to exclude any of them from the list by doing Ctrl + Click on the column caption (is this possible?). If just using Click, that column will become the only key for sorting.
3. What about supporting complex filters (AND, OR, AND NOT)?
    a. Current Pick List leverages Woll2Woll wwFilterDialog component, but we need to implement our own Filter Dialog and Filter Manager. If that is the case, we have the opportunity to provide something better and allow to reuse filters. DevExpress uses the following idea


Adds a new condition
Removes this condition
Logical operator combining conditions
Click to change the operator, and add/remove conditions and groups of conditions
Filter conditions combined into a group
Filter Editor
Column
Click to select another column
Criteria operator
Click to change the operator
OK
Cancel
Apply
Operand value
Click to specify a value

Or
And
Product Equals Chai
Quantity Is greater than 20
And
Product Is any of (Tofu , Konbu)
Quantity Is less than 100

    b. We could keep a history of filters applied to the same Pick List and save them as local cookies or shared on the server.
4. How to support Single and Multiple Selection?
    a. Internally, the Pick List Dialog uses database bookmarks for any selected record in the provided dataset. But the caller could be using a totally different dataset so that it should receive only the primary keys in the selection. That translation is currently done in the internal callback.
    b. Current multiple selection, as implemented by TUniDBGrid, only allows selecting multiple records in one page. If the feature is not improved, we will need to monitor the page changes (scrolling) and keep our own selection.
5. How to persist user changes in the visual design (column width, column order, etc)?
    a. As in EnterpriseIQ we save that information in the local registry, it makes sense to save the same information to local cookies.
    b. Some possible options are to make the cookies local to the current user or shared to all users of that PC. Another option is to save that information to the server, both for the current user or to all users. In any case, it is important to take into account that, even if it is the same user, working on a different PC could make a configuration invalid. A good starting point is to save the configuration to a current user cookie.
6. How to propagate Sort and Scope (Filter)?

   a. When propagating the current active Sort and Filter in the Pick List to the main form DataSet, we are assuming that the Pick List is using a subset of the fields in the DataSet.
   b. Another thing is that we have two options for returning both Sort and Scope: as strings used by the DataSet (like what we use internally in the Pick List Dialog), or as strings used in a SELECT statement.
   c. The syntax required by the Filter expression is compatible with the condition in the WHERE clause so that we can always return our current filter.
   d. The syntax used in IndexFieldNames for our internal sorting is particular to FireDAC, and we should provide a class function for converting it to a common ORDER BY expression.
   e. If we allow passing optional parameters for recovering these options, we could get both as part of the dialog callback (let's say, Filter, SortIndexNames, SortOrderBy).
7. There are several other features in our current Pick List which should be evaluated for inclusion:
   a. Sub queries
   b. Custom buttons

TIQWebHPick non-visual component (using PickListDlg)

This component will expose two properties: Title (to be shown in the Pick List Dialog) and the DataSet (FireDAC or a common TDataSet).

Its goal is to allow single or multiple selection (propagating Sort and / or Scope if requested), according to the intended use.

For example, in EnterpriseIQ we use it in the OnClick event of a button for providing our own TDBLookupComboBox.

Visual component like TUniDBLookupComboBox but using TIQWebHPick

In order to avoid manually designing data-entry forms like our current main forms capable of switching between Grid and Form View, we need to provide a visual control for all fields intended to be entered by using Pick Lists.

This component will require the same information of a common DBLookupComboBox in addition to the link to a TIQWebHPick. At run-time, instead of showing a combo box, it will show the Pick List Dialog.

Visual component providing dataset-based dual view (Grid / Inspector)

Visual component implementing a dataset-based hierarchical data inspector

Useful References to UniGUI Demos

Basic jQuery

- It shows how to embed HTML code using jQuery for rendering JavaScript widgets.
- The demo itself shows two digital clocks, while I added a new analog clock provided by a UniGUI user.

Collapsible Panels

- TUniSplitter
- TUniPanel with Collapsible = True and CollapseDirection = cdDefault

Client Events

- CE-1 Showing how to attach client-side events in JavaScript which allow client-side actions like
  - `function form.Onmousemove(sender, x, y)`
  - `{`

        - `MainForm.UniEdit1.setValue(x+' : '+y);`
    - `}`
- CE-2 Also, triggering a client-side AJAX request which is resolved as a server-side event
    - Client-side event
        - `function Onmousedown(sender, x, y)`
        - `{`
        - `    ajaxRequest(sender, 'myAjaxEvent', ['param0=MyParam', 'X='+x, 'Y='+y] );`
        - `}`
    - The client-side AJAX request is received as a server-side AJAX event (OnAjaxEvent) and handled as any Delphi event.
        - `procedure TMainForm.UniPanel1AjaxEvent(Sender: TComponent; EventName: string; Params: TStrings);`
        - `begin`
        - `  if EventName='myAjaxEvent' then`
        - `  begin`
        - `    UniMemo1.Lines.Add('Server Response:');`
        - `    UniMemo1.Lines.Add('===============');`
        - `    UniMemo1.Lines.Add(Params.Values['param0']);`
        - `    UniMemo1.Lines.Add(Params.Values['X']);`
        - `    UniMemo1.Lines.Add(Params.Values['Y']);`
        - `    UniMemo1.Lines.Add('');`
        - `  end;`
        - `end;`
- CE-4 Shows how to send an AJAX request and receive the answer as an AJAX callback
    - Button1 intercepts the AJAX request and its response
    - Button2 sends a response from the server and receive it at the client callback
    - Button3 sends a response which is directly executed as JavaScript code
- CE-5 Uses form.Onkeydown for capturing key and shift status in order to move a panel without executing server code.

```
function form.Onkeydown(sender, key, shift)
{
   var xy=MainForm.UniPanel1.getPosition(true);
   var x=xy[0];
   var y=xy[1];

   var inc=5;
   if (shift & 1) inc=10;  // shift
   if (shift & 4) inc=1;   // ctrl
   if (shift & 2) inc=20;  // alt

   switch(key)
   {
      case 40 : y+=inc; break;
      case 38 : y-=inc; break;
      case 37 : x-=inc; break;
      case 39 : x+=inc; break;
   }

   if (y<0) y=0;
   if (x<0) x=0;

   if (y>MainForm.form.getHeight()) y=MainForm.form.getHeight();
   if (x>MainForm.form.getWidth()) x=MainForm.form.getWidth();

   MainForm.UniPanel1.setPosition(x, y);
}
```

Dynamic Client Events

- Client Events can be defined by assigning them at runtime using a simple interface:

```
procedure TMainForm.UniFormCreate(Sender: TObject);
begin
UniButton1.ClientEvents.ExtEvents.Values['click']:='function(sender){alert("Click")}';

UniEdit1.ClientEvents.ExtEvents.Values['change']:='function(sender, newValue){MainForm.UniEdit2.setValue(newValue)}';

Self.ClientEvents.ExtEvents.Values['form.click']:='function(sender){MainForm.UniEdit2.setValue("form.click")}';

Self.ClientEvents.ExtEvents.Values['window.move']:='function(sender){MainForm.UniEdit2.setValue("window.move")}';
end;
```

- This kind of feature could allow to define client-side validations which could be requested dynamically from the server to apply some business rules without required round trips to the server (something like the RemObjects DataAbstract client-side business rules).

Client Info

- It shows how to get information about the browser, IP address, etc.
- By using JavaScript events, any mouse movement over the window changes the background color to Blue.

Client-side Alignment

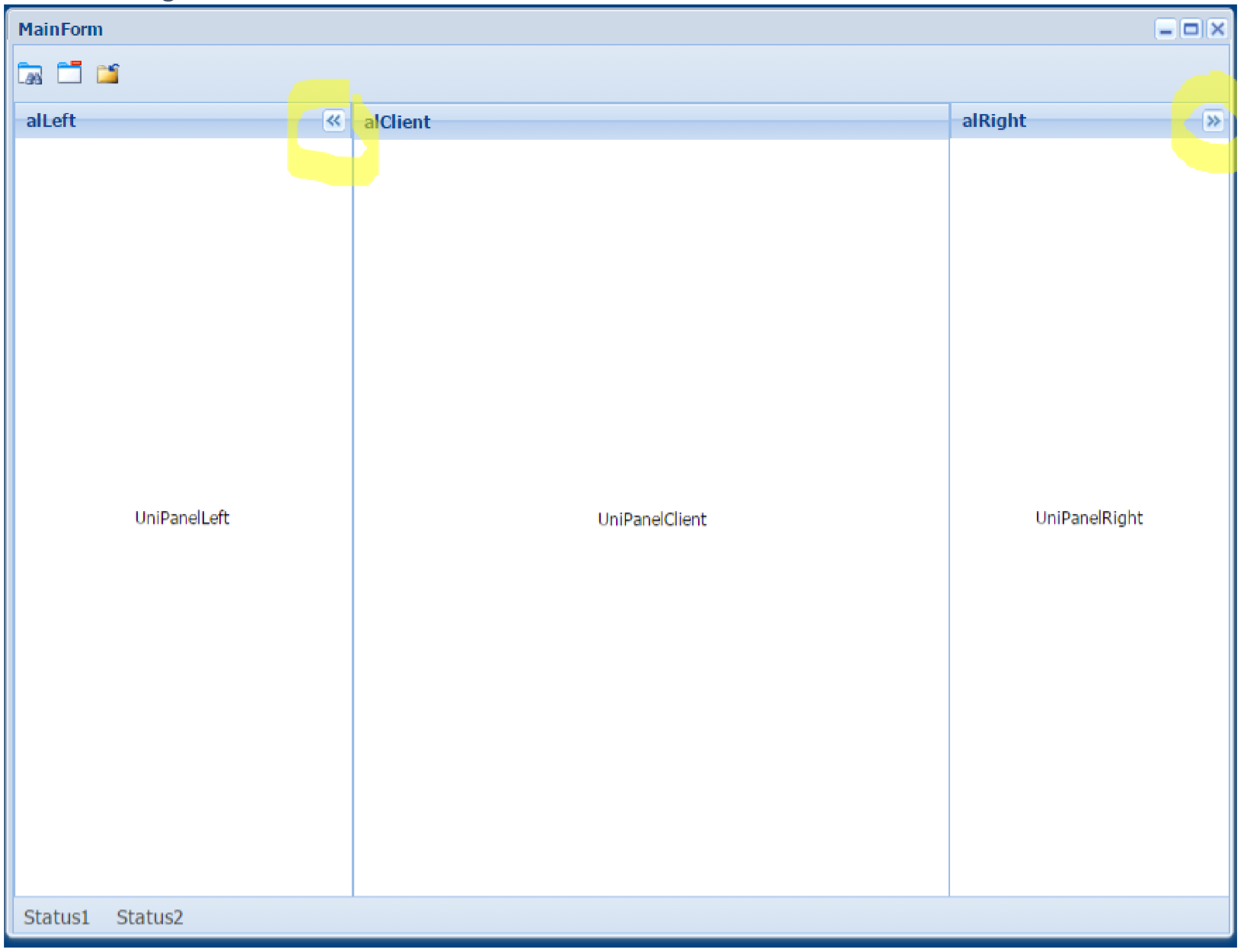
UniGUI leverages all native Sencha layouts, all of them configured in the properties related to LayoutControl.

Sencha references:

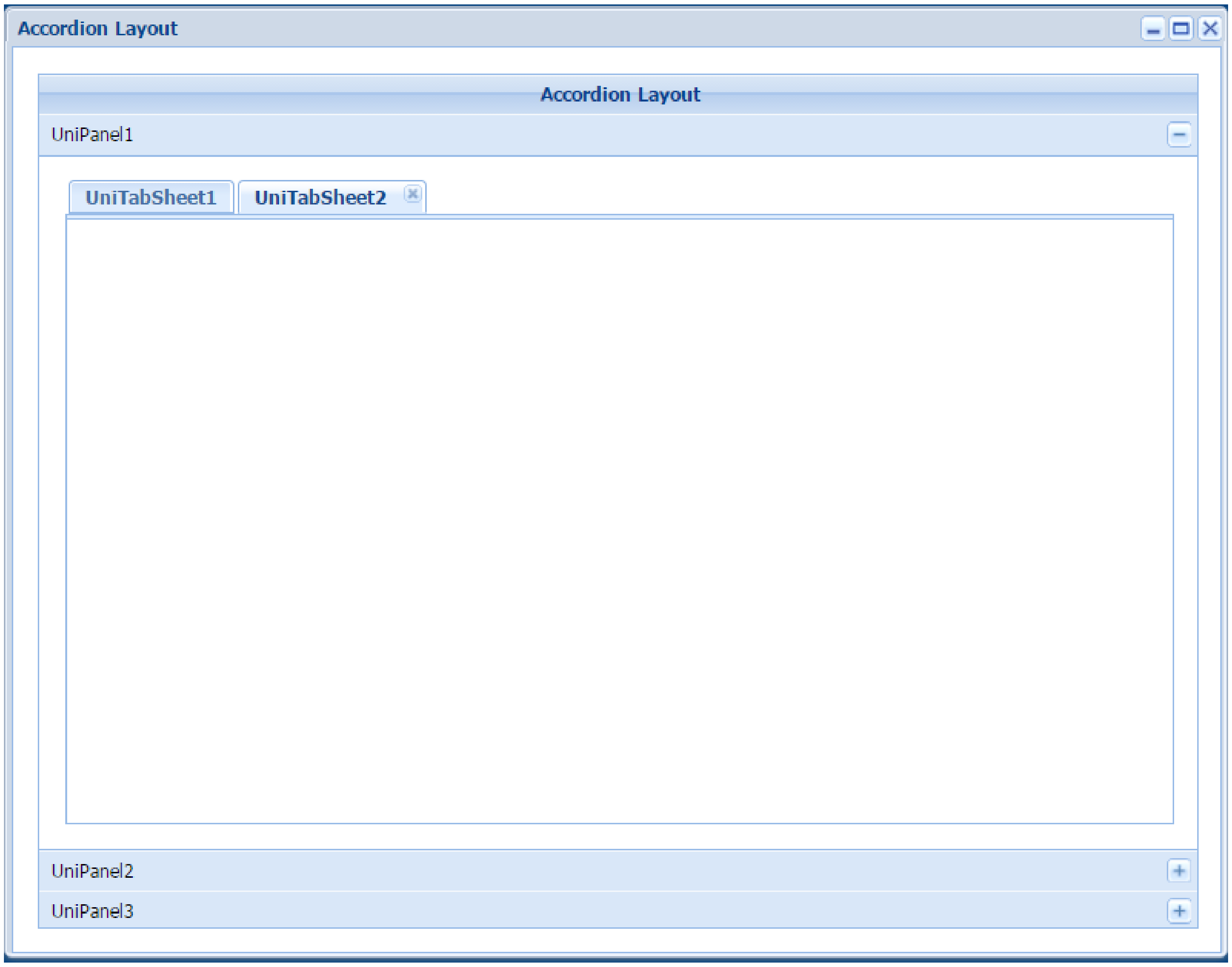
http://docs.sencha.com/extjs/4.2.4/extjs-build/examples/layout-browser/layout-browser.html

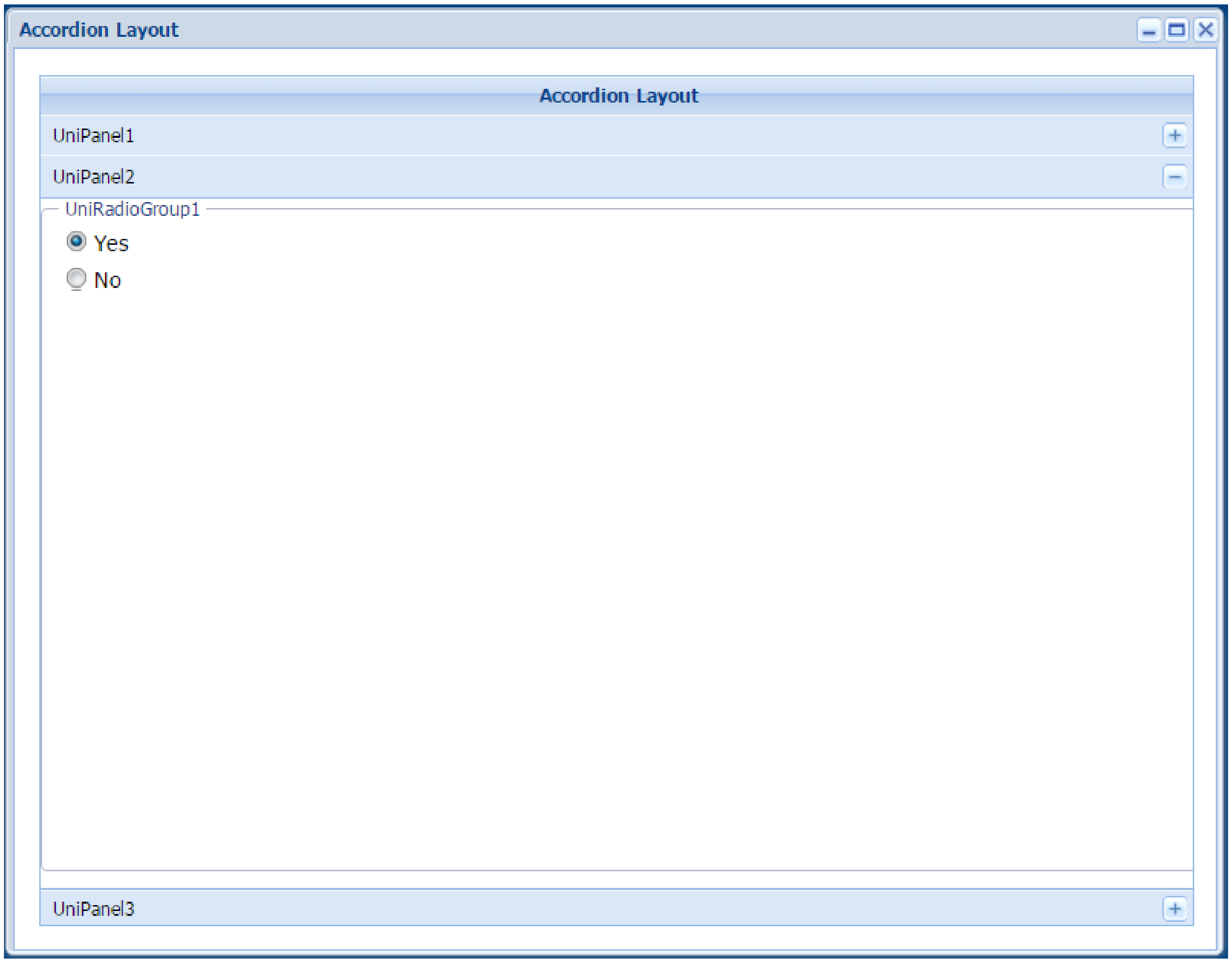
http://docs.sencha.com/extjs/4.2.4/#!/api/Ext.enums.Layout

Dock and Align

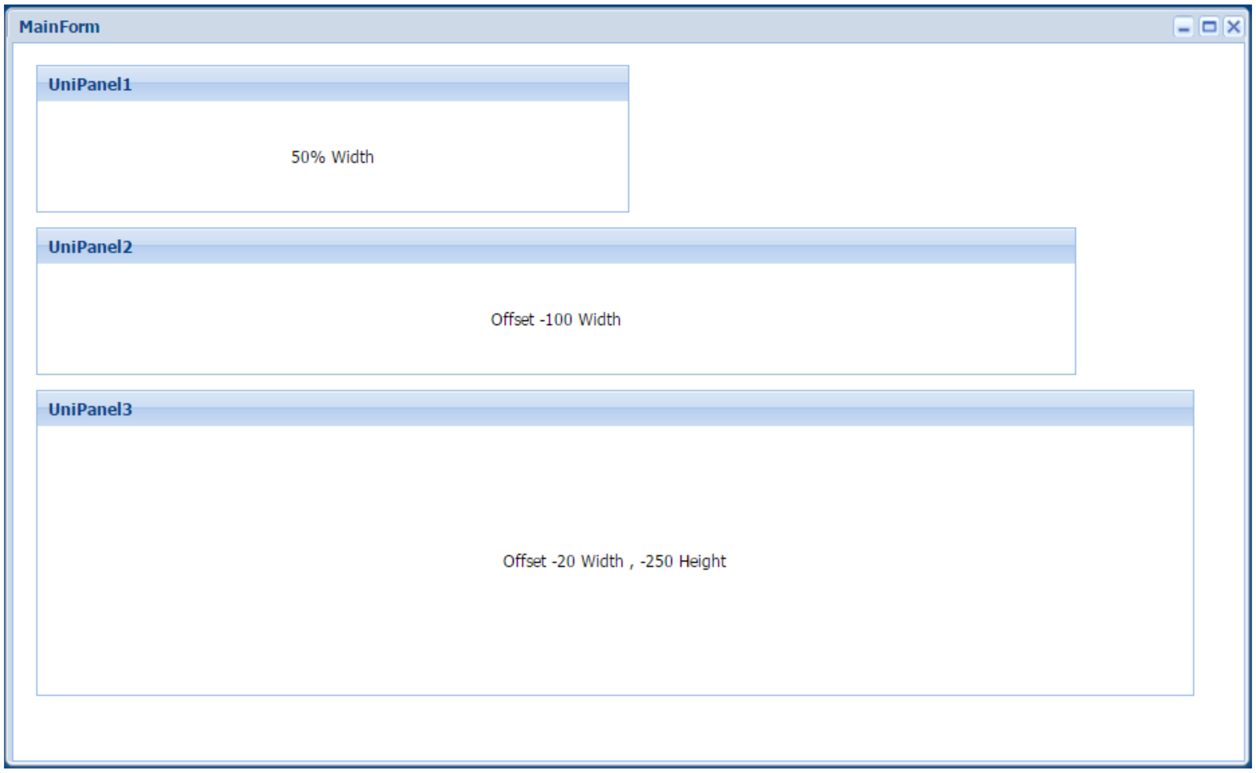


Accordion

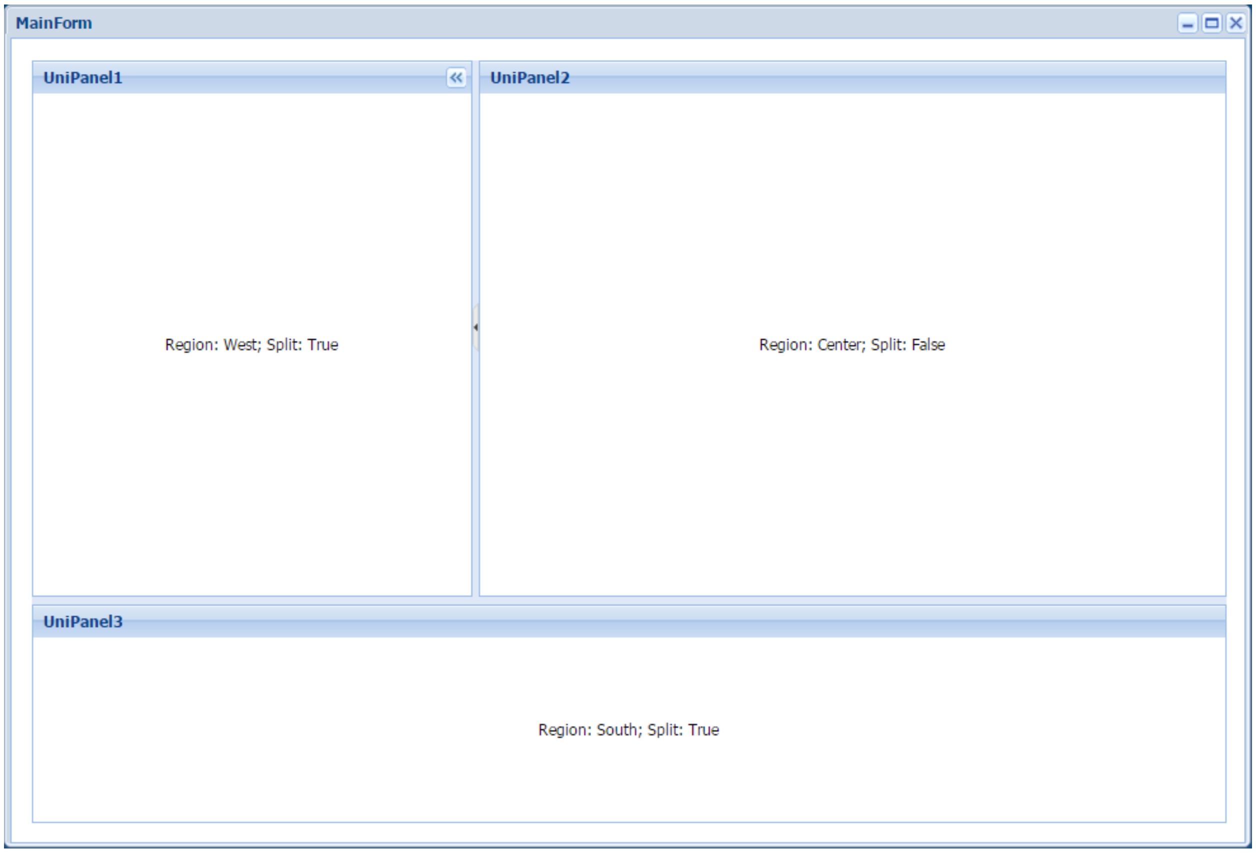

Accordion Layout
Accordion Layout
UniPanel1
UniTabSheet1
UniTabSheet2
UniPanel2
UniPanel3


Accordion Layout
Accordion Layout
UniPanel1
UniPanel2
UniRadioGroup1
Yes
No
UniPanel3

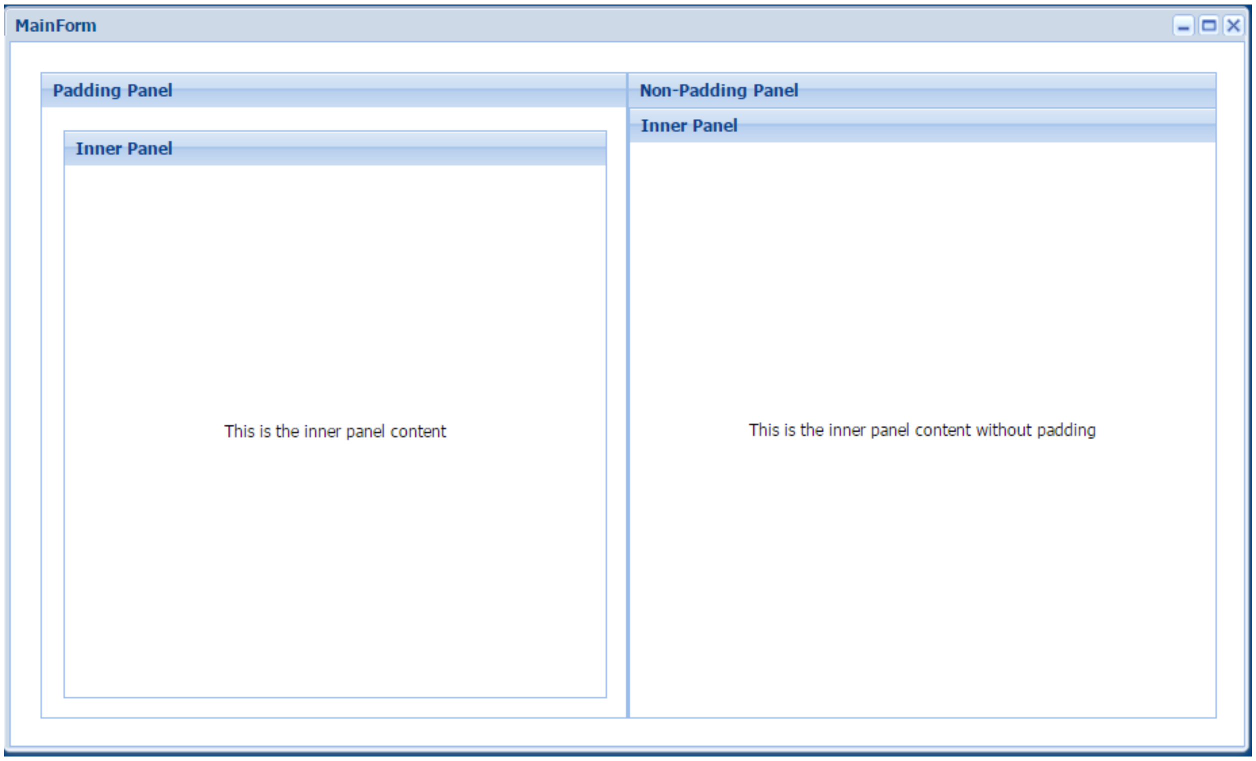
Anchor



Border


MainForm
UniPanel1
Region: West; Split: True
UniPanel2
Region: Center; Split: False
UniPanel3
Region: South; Split: True

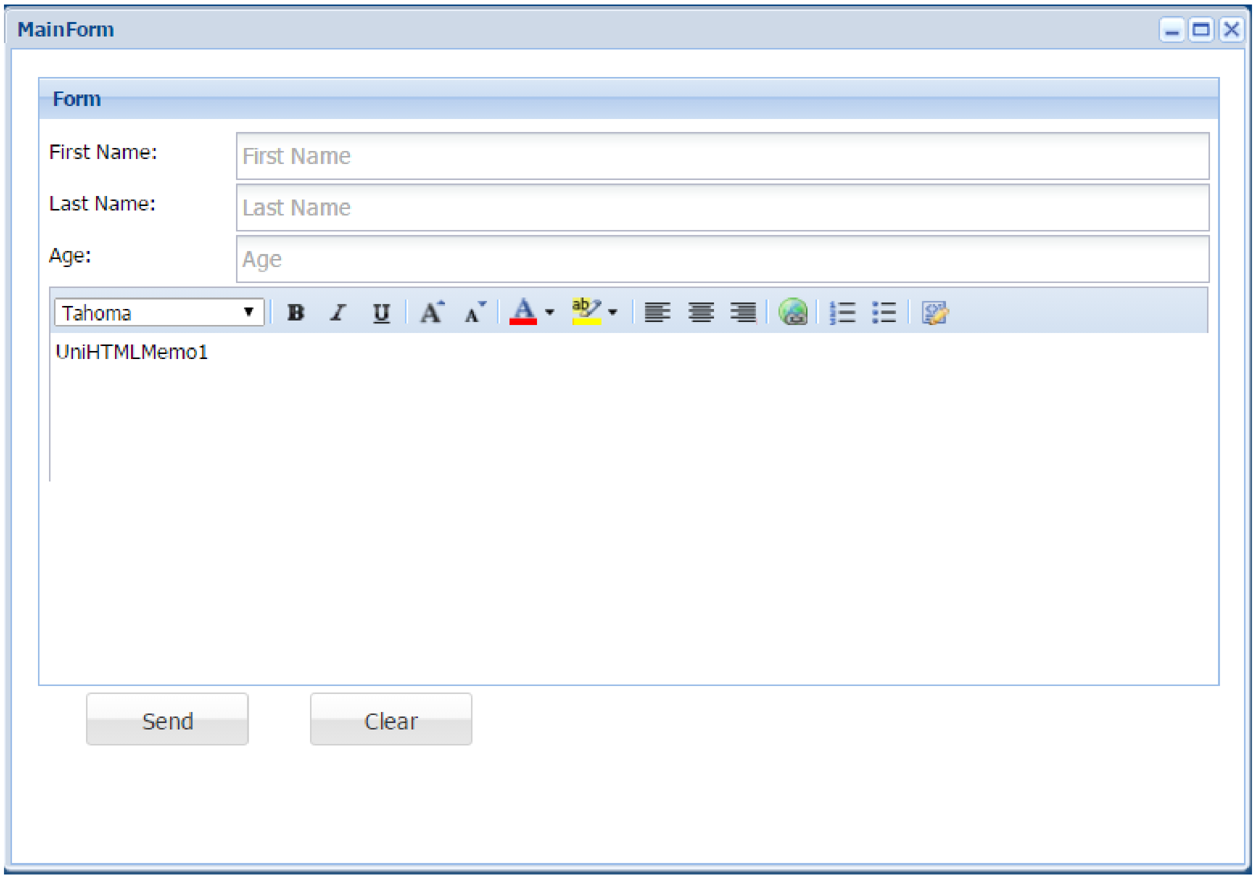
Fit



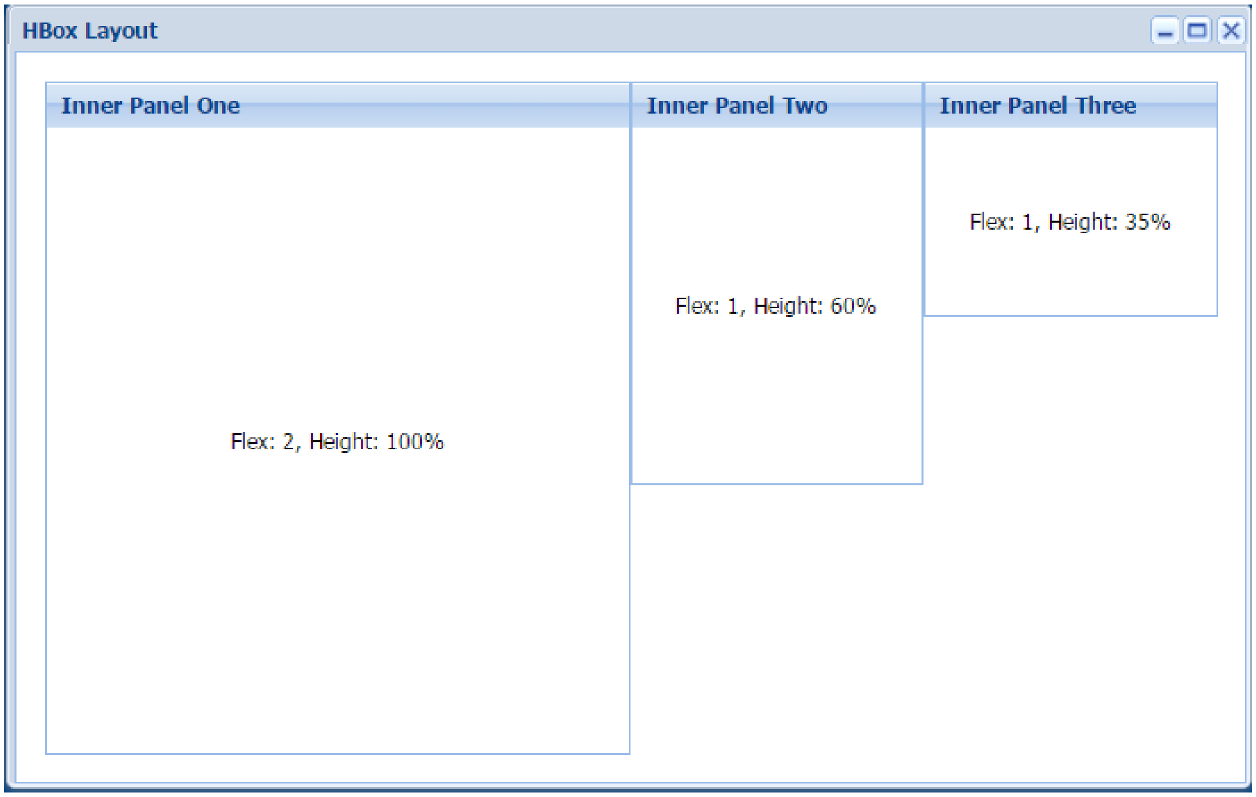
This is the inner panel content

This is the inner panel content without padding

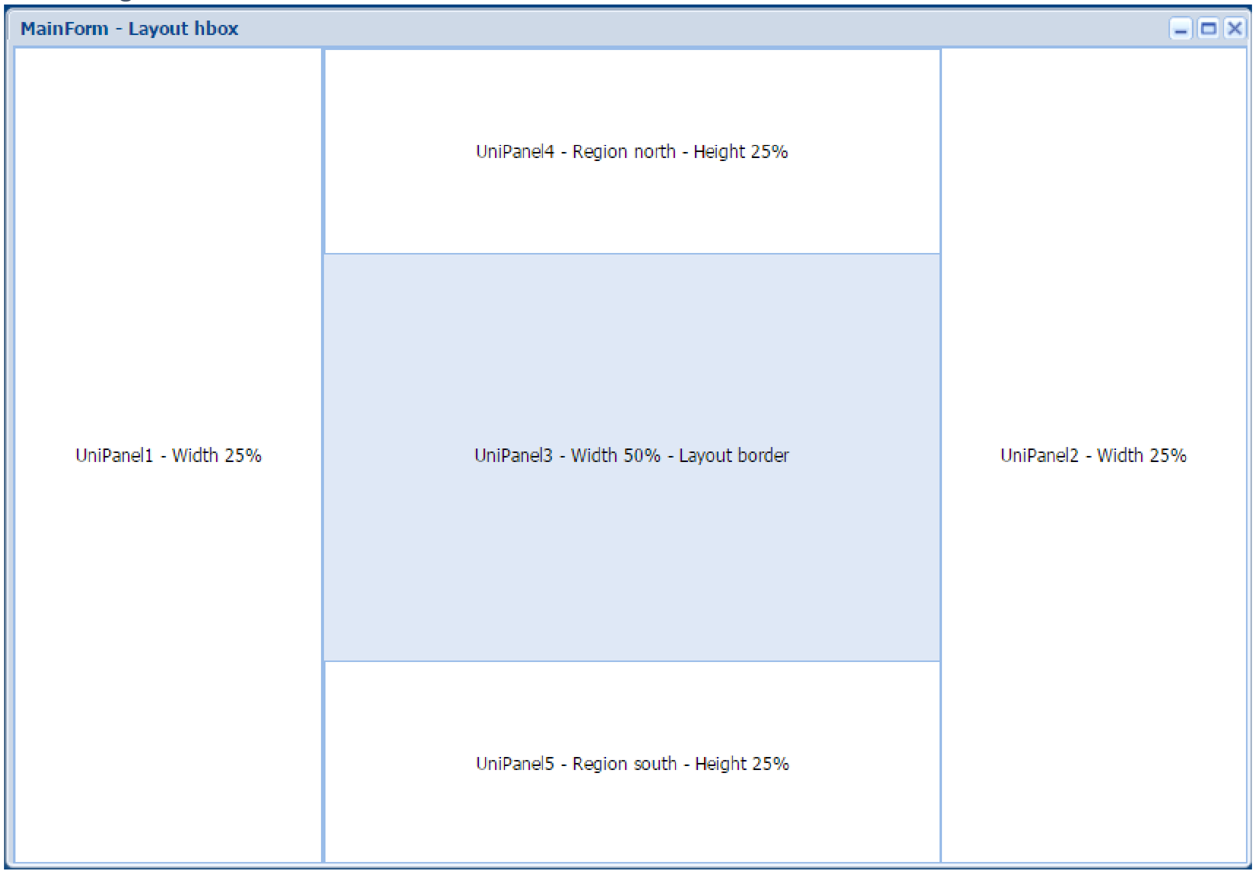
Form



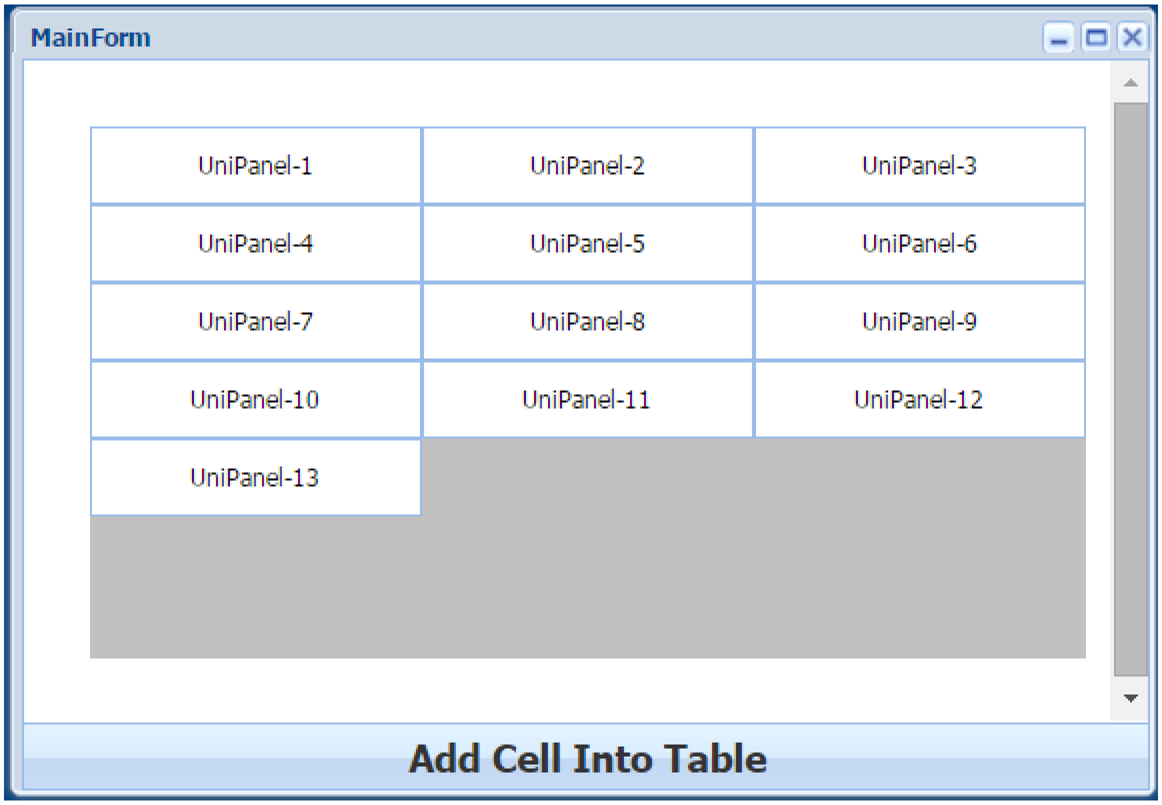
HBox


HBox Layout
Inner Panel One
Inner Panel Two
Inner Panel Three
Flex: 2, Height: 100%
Flex: 1, Height: 60%
Flex: 1, Height: 35%

Percentage

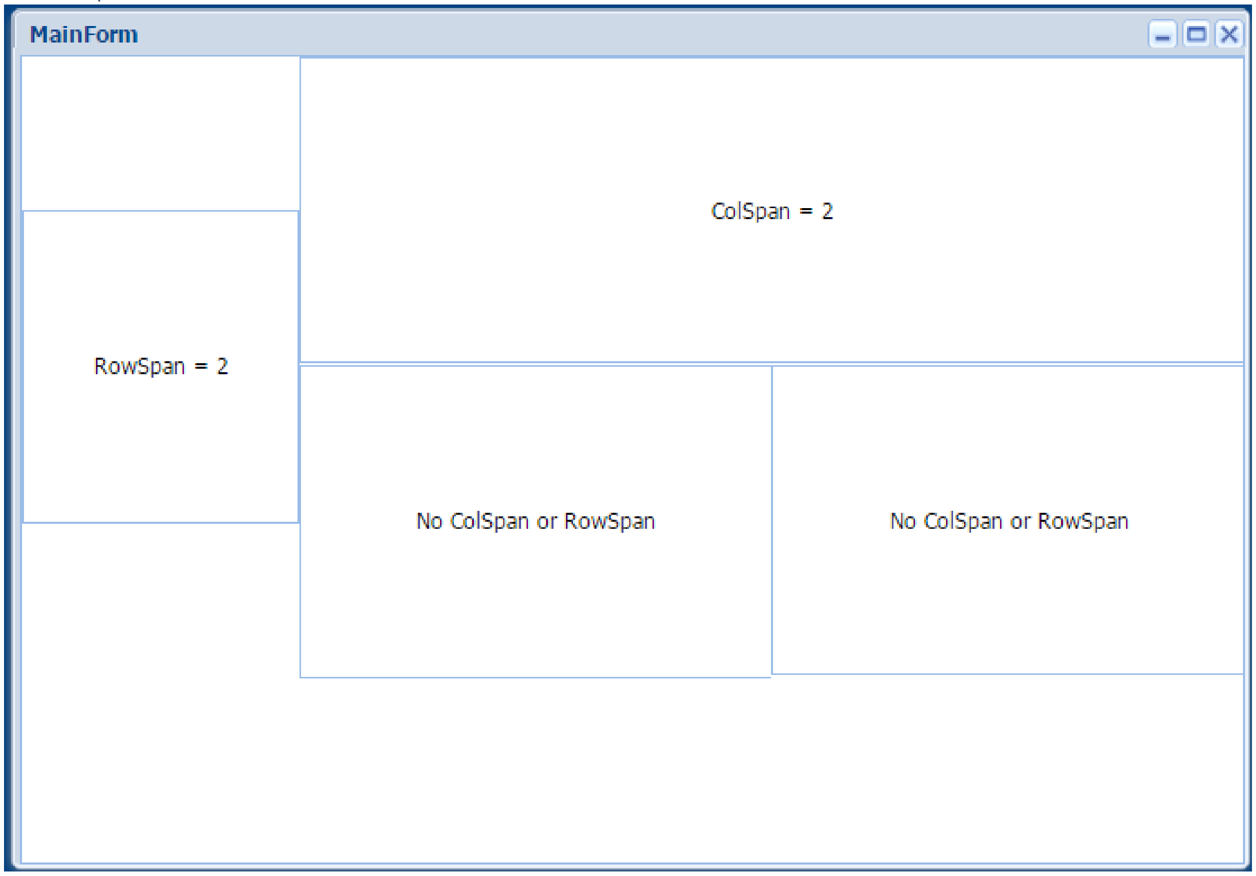

MainForm - Layout hbox
UniPanel4 - Region north - Height 25%
UniPanel1 - Width 25%
UniPanel3 - Width 50% - Layout border
UniPanel2 - Width 25%
UniPanel5 - Region south - Height 25%

Table



UniPanel-1	UniPanel-2	UniPanel-3
UniPanel-4	UniPanel-5	UniPanel-6
UniPanel-7	UniPanel-8	UniPanel-9
UniPanel-10	UniPanel-11	UniPanel-12
UniPanel-13		

Table Span


MainForm
ColSpan = 2
RowSpan = 2
No ColSpan or RowSpan
No ColSpan or RowSpan

VBox

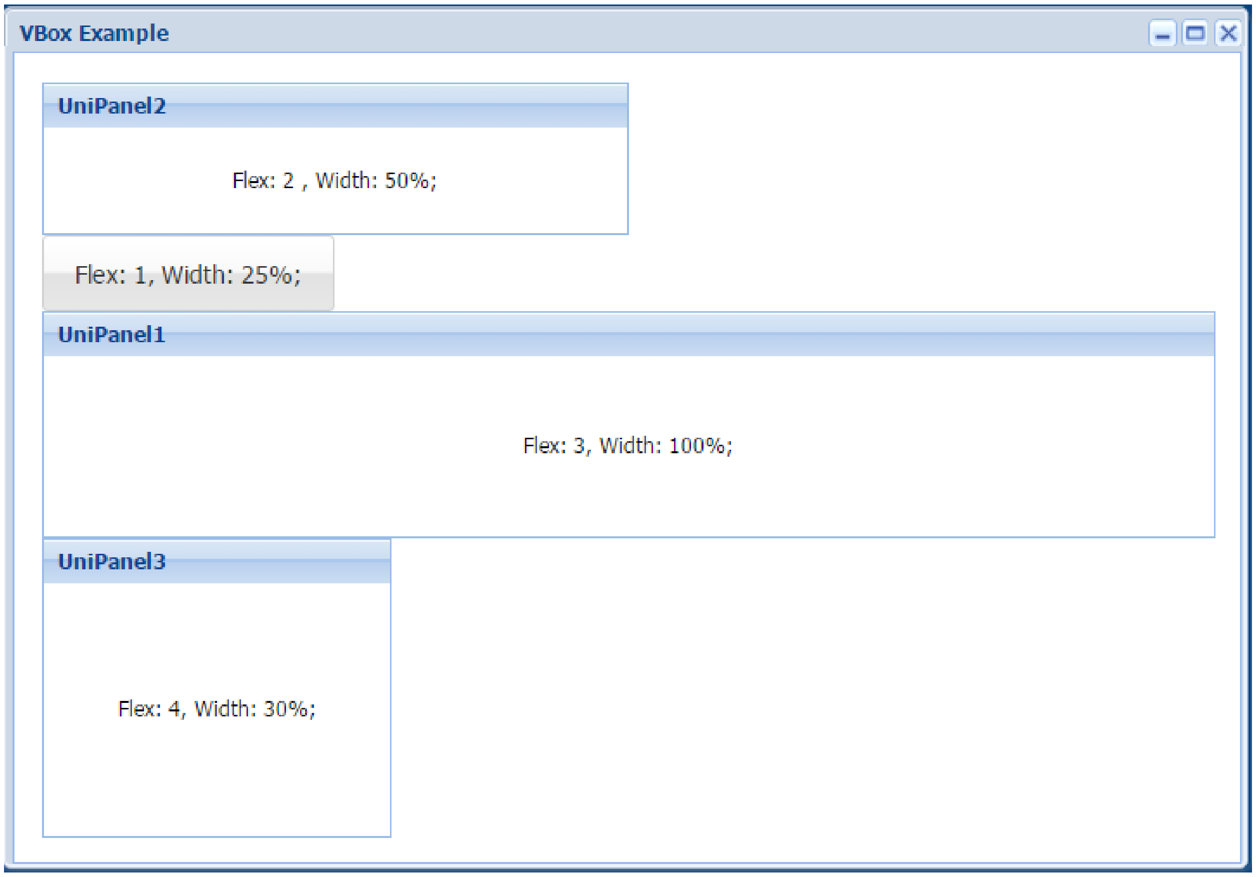
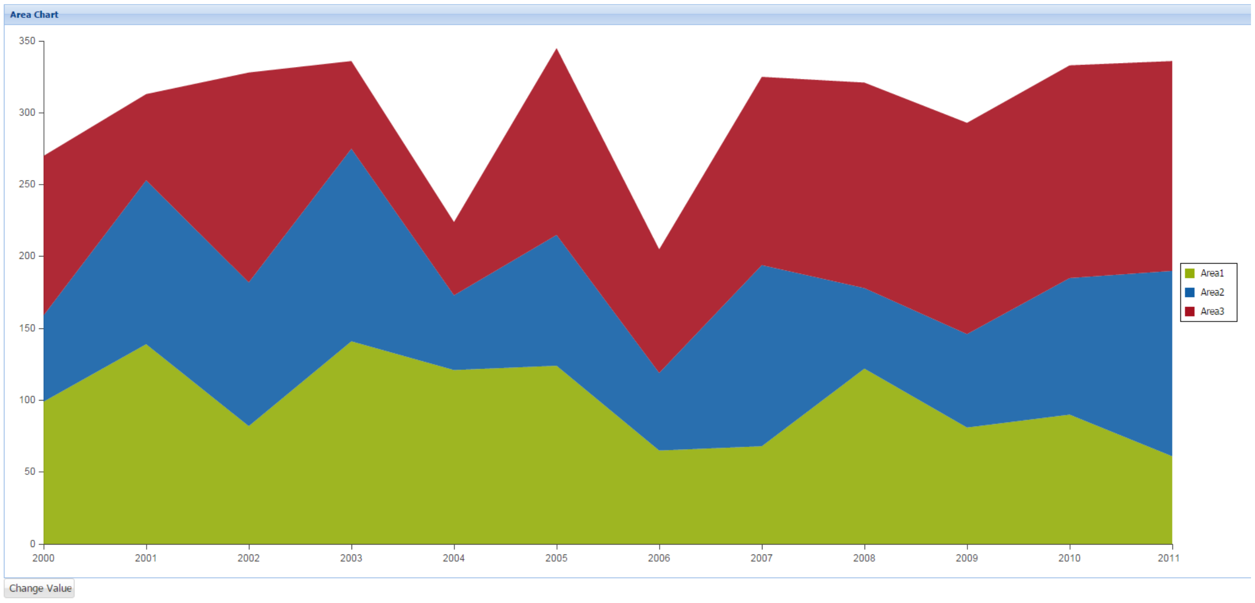
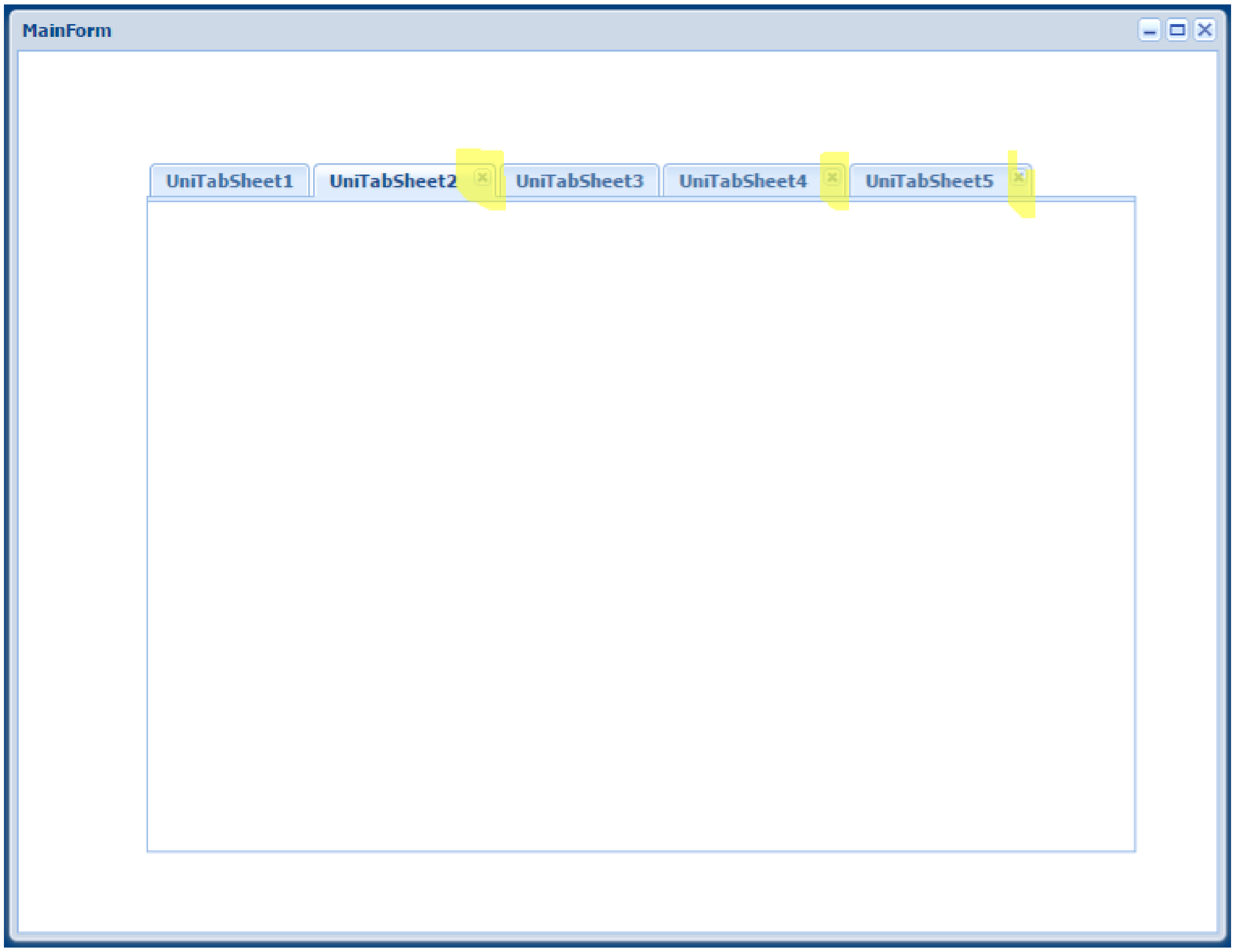


Chart Areas



Closable Tabs

- It is a per-tab property



Login Form

If there is a form inheriting from **TUniLoginForm**, that form will be the first to be shown by the application. If its ModalResult is mrOK, then the **MainForm** will be shown.

Session List

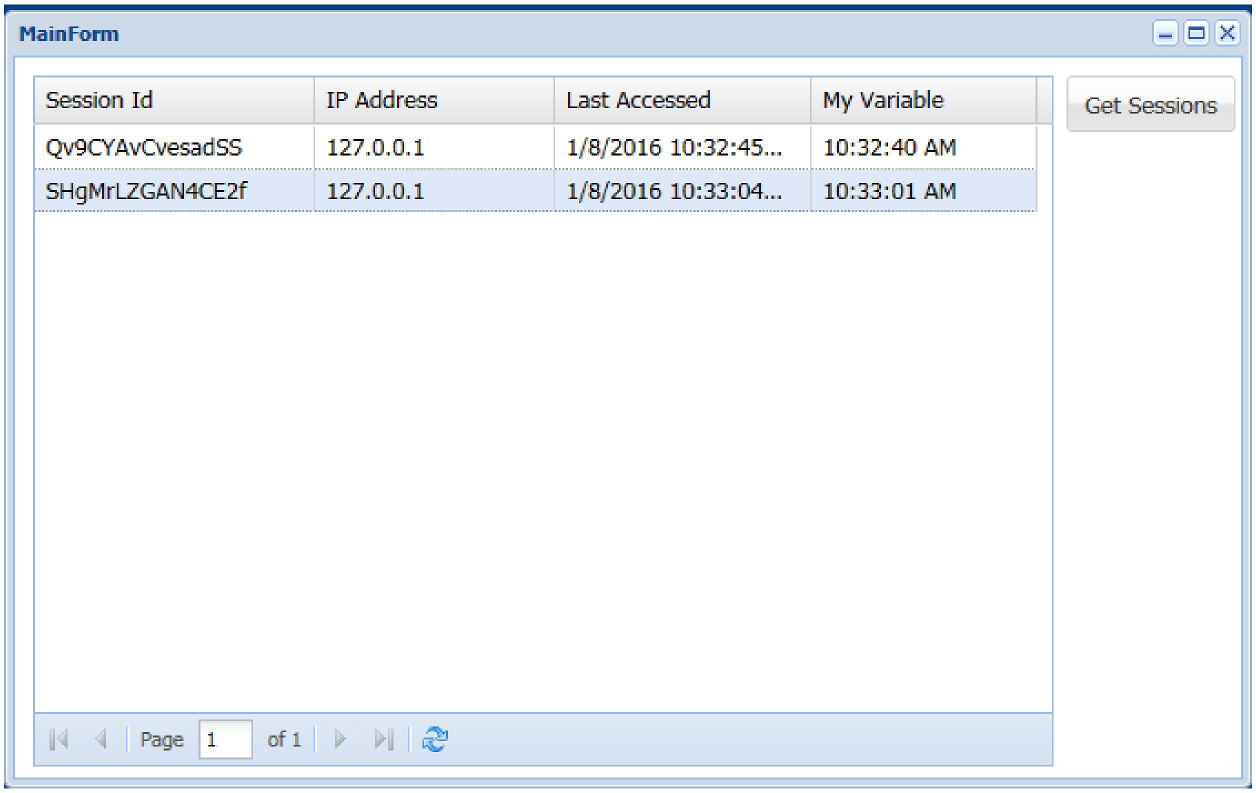
ServerModule is a singleton containing the application configuration and resources which will be shared across all sessions.

MainModule (or any TDataModule created with UniGUI Wizard) will be instantiated by session.

Sessions can be managed as shown in this demo.

For example, if some user tries to connect to the system, we will check licenses before allowing his access. Some scenarios come to mind:

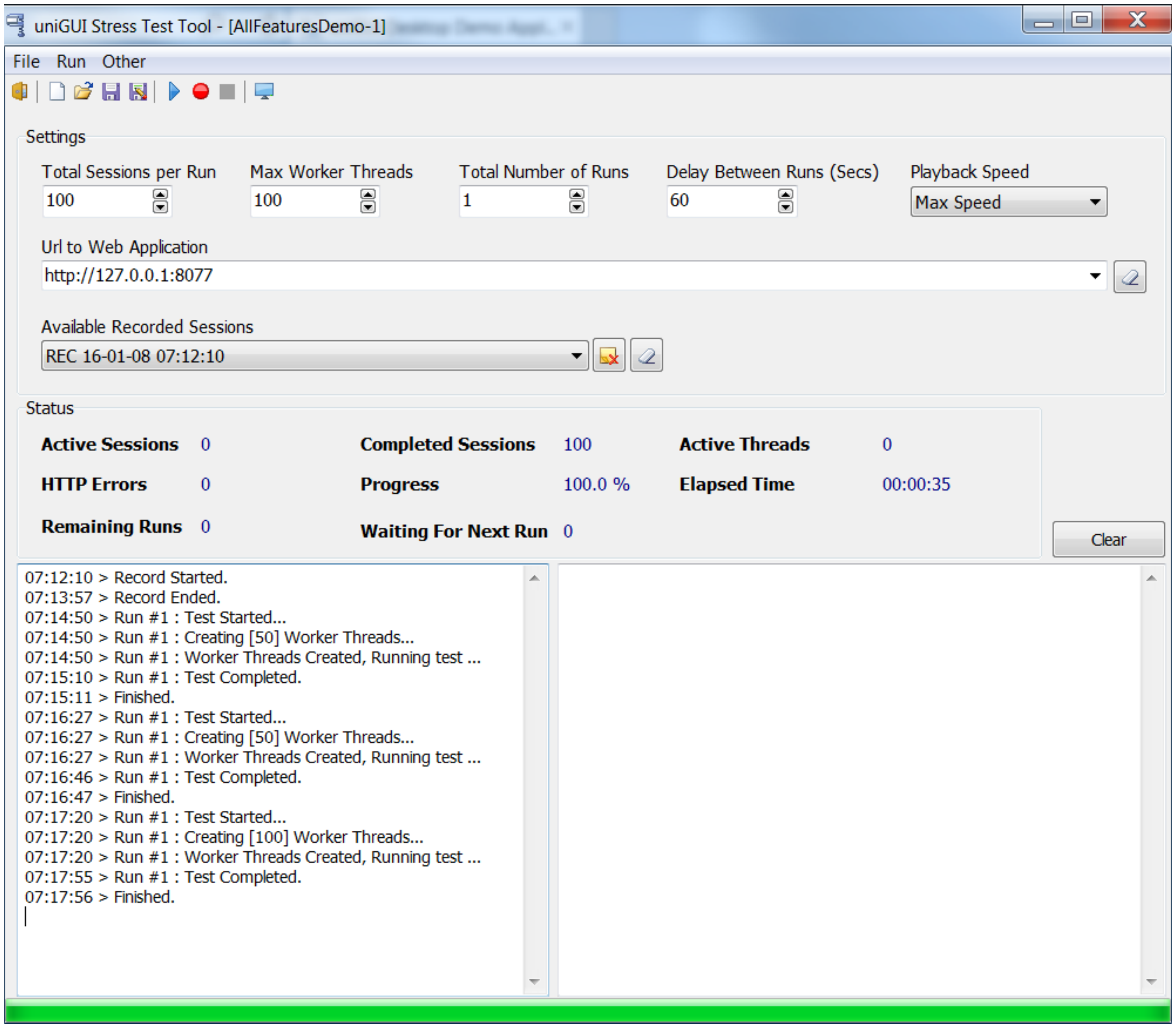
- The same user is already logged in from other computer (the program is still open there), and he could choose to terminate that session and initiate a new one.
- Another user is logged in, but he just left the program open, so that his session could be terminated (if the new user is allowed to do that).
- Same previous scenario, but the user has the right to terminate any other session (Super Admin).



Session Id	IP Address	Last Accessed	My Variable
Qv9CYAvCvesadSS	127.0.0.1	1/8/2016 10:32:45...	10:32:40 AM
SHgMrLZGAN4CE2f	127.0.0.1	1/8/2016 10:33:04...	10:33:01 AM

Stress Test Tool

This tool allows recording live sessions to simulate multiple clients executing the same requests to the server. Before being able to record sessions for an application, that option must be enabled in the Server Module. Recording and Stress can be done from the tool itself.

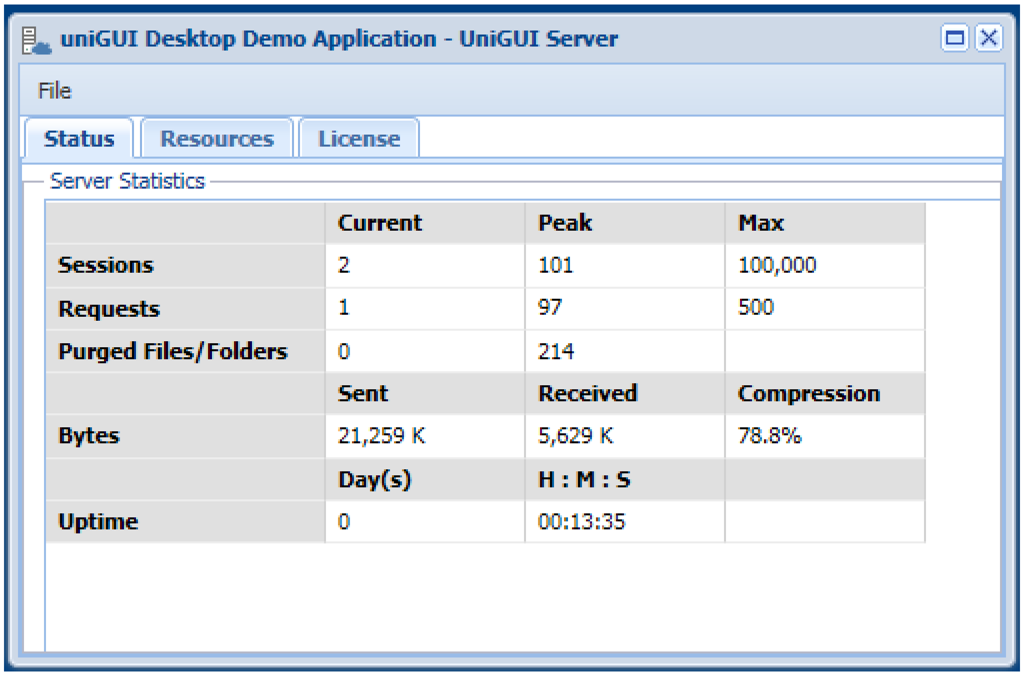


Total Sessions per Run: 100
Max Worker Threads: 100
Total Number of Runs: 1
Delay Between Runs (Secs): 60
Playback Speed: Max Speed
Url to Web Application: http://127.0.0.1:8077
Available Recorded Sessions: REC 16-01-08 07:12:10

Active Sessions	0	**Completed Sessions**	100	**Active Threads**	0
HTTP Errors	0	**Progress**	100.0 %	**Elapsed Time**	00:00:35
Remaining Runs	0	**Waiting For Next Run**	0		

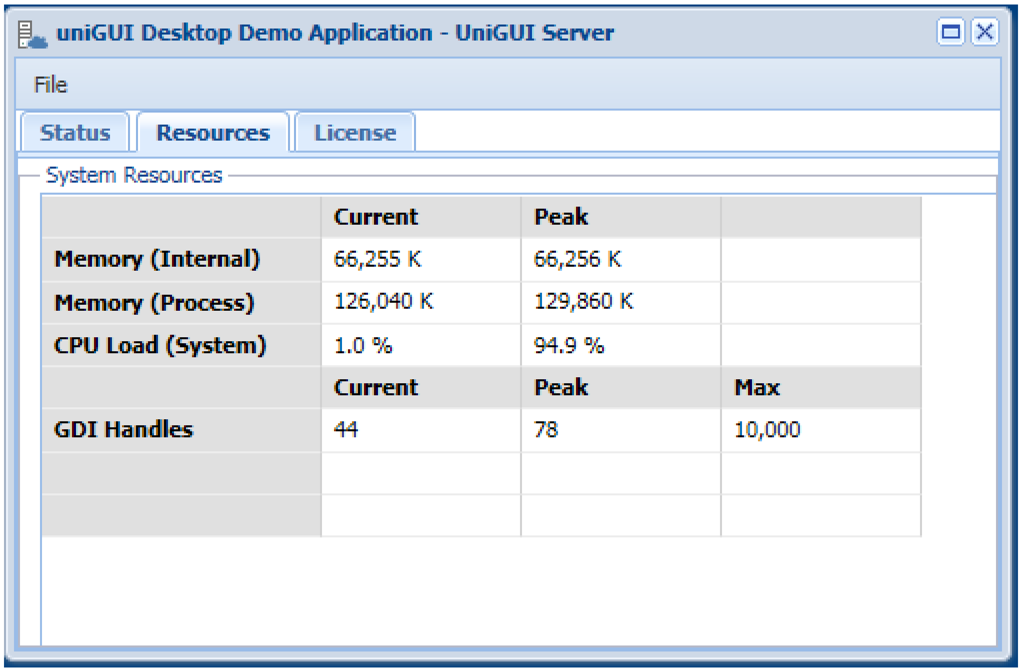
```
07:12:10 > Record Started.
07:13:57 > Record Ended.
07:14:50 > Run #1 : Test Started...
07:14:50 > Run #1 : Creating [50] Worker Threads...
07:14:50 > Run #1 : Worker Threads Created, Running test ...
07:15:10 > Run #1 : Test Completed.
07:15:11 > Finished.
07:16:27 > Run #1 : Test Started...
07:16:27 > Run #1 : Creating [50] Worker Threads...
07:16:27 > Run #1 : Worker Threads Created, Running test ...
07:16:46 > Run #1 : Test Completed.
07:16:47 > Finished.
07:17:20 > Run #1 : Test Started...
07:17:20 > Run #1 : Creating [100] Worker Threads...
07:17:20 > Run #1 : Worker Threads Created, Running test ...
07:17:55 > Run #1 : Test Completed.
07:17:56 > Finished.
```

Once a test is executed, it is possible to examine the status of the application under test.



	Current	Peak	Max
Sessions	2	101	100,000
Requests	1	97	500
Purged Files/Folders	0	214	
	Sent	Received	Compression
Bytes	21,259 K	5,629 K	78.8%
	Day(s)	H : M : S	
Uptime	0	00:13:35	

This test simulated 100 simultaneous sessions.



	Current	Peak	
Memory (Internal)	66,255 K	66,256 K	
Memory (Process)	126,040 K	129,860 K	
CPU Load (System)	1.0 %	94.9 %	
	Current	Peak	Max
GDI Handles	44	78	10,000

As each client receives a session, it is important to monitor memory usage looking for leaks that could render the application unusable after some time.