ccDevnet

Data Access Server

Developers Guide

Content

Introduction

The cc Devnet Data Access Server platform provides the base around which you can develop powerful business applications. cc Devnet's Data Access Server can be used for building stable, secure and scalable business applications.

This document is intended for you, the developer who has downloaded and run the full cc Devnet Data Access Server setup onto his development machine.

The full setup includes the server itself, the server control panel, Firebird Database Server, and three sample applications with source code.

This gives you a complete working environment isolated to your single workstation where you can run the sample applications against the server and where you can develop your own applications and test them.

Once you are ready to test the sample apps in your own cloud based environment, you should download and run the server only setup on your cloud server and change the server and the clients' settings as documented below.

General Note: Throughout this document we refer to the folder "C:\Program Files\ccDevnet\" or folders below it. If you have a 64bit machine, please replace this with "C:\Program Files(x86)\ccDevnet\". If you do not require support for the Firebird Database, we can supply a full 64bit setup on request.

All of the sample projects are in a My Documents folder called "DataAccessServer Projects"

If you require further assistance, please feel free to email support@cc-devnet.com

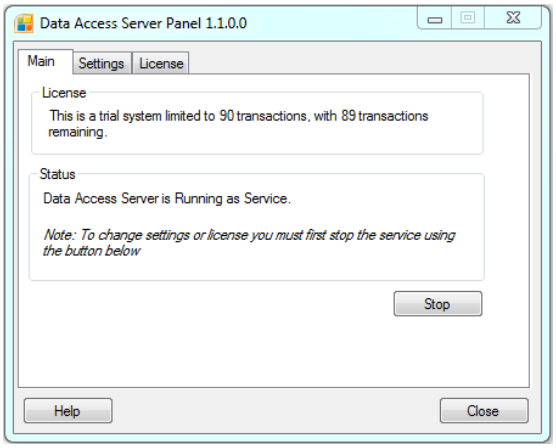
Data Access Server Control Panel

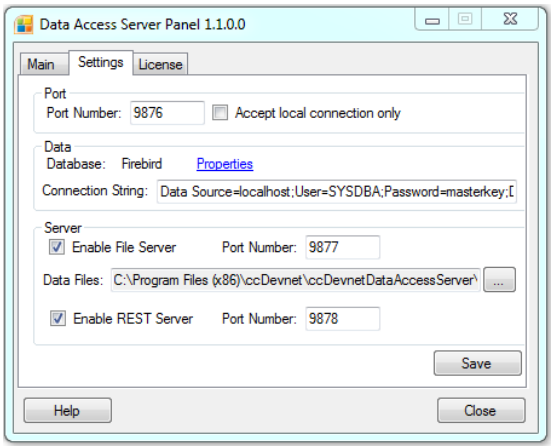
The Data Access Server runs as a service, on your machine for development or on a LAN or cloud server for production.

It is controlled via the Data Access Server Control Panel, which can be opened by clicking on the "ccDevnet Data Access Server" icon on your desktop or via Windows Start button. This must be done directly on the machine where the server is running as a service.

The default setup will have automatically configured the system to be fully operational with your machine acting as server and a workstation and will to be connected to the free included Firebird Database.

Use the instructions below to make appropriate changes to the default server or database settings.





Server Settings

To view the main server and database settings, click on the “Settings” tab.

You can only make changes to these settings if the server is not running. To do so return back to the “Main” tab and press “Stop” and wait until the “Status” changes to “Server Not Running”.

Note, we use a Main Data Communication Port (9876), a File Transfers Port (9877) and a REST Server Port (9878). Usually with a local server no configuration is required for port numbers, but with a remote server you have to set rules in your firewall to allow requests from specified ports.

You may change these port numbers to any 4 digit values and make the corresponding changes in your firewall.

There is a security key called “passkey” which both the server and the client must know in order to manage secure communications between themselves. On the server side the” passkey” is defined in the xml file “C:\Program Files\ccDevnet\ccDevnetDataAccessServer\Server\system.xml”. Its default value is “DemoKey”.

On the client side, it’s up to the developer how this value should be handled, i.e. within the code or added to client thought an encrypted file.

Database Settings

The system currently supports Firebird (included free with the installation) and Microsoft SQL Server for which you need to purchase separately from Microsoft.

Firebird Server

If you chose firebird server, you don’t need to make any changes. A sample Contacts database is included with the installation. The default details are as below.

Default installation path:
“C:\Program Files\ccDevnet\ccDevnetDataAccessServer\DB\CONTACTSDB.FDB”
Default Username: SYSDBA
Default Password: masterkey

You may want to download the free Firebird Server administration tool FlameRobin for data manipulation.

SQL Server

For SQL Server follow the steps below.

1. Check if you have SQL Server installed, if not, you can download SQL Server Express, available free to developers from Microsoft's website.
2. Open SQL Server Management Studio and create new database named CONTACTSDB.
3. Execute the following script in CONTACTSDB; "C:\Program Files\ccDevnet\ccDevnetDataAccessServer\SqlServer".
4. Go back to the Data Access Server Control Panel and press "Data/Properties". Choose database, test connection and accept.

File Server

Check the default location is suitable for you to save files for file server and change if required.

REST Server

If you need to work with REST API specially for ios, android clients enable that, else there is no need to enable REST Server.

Save Changes

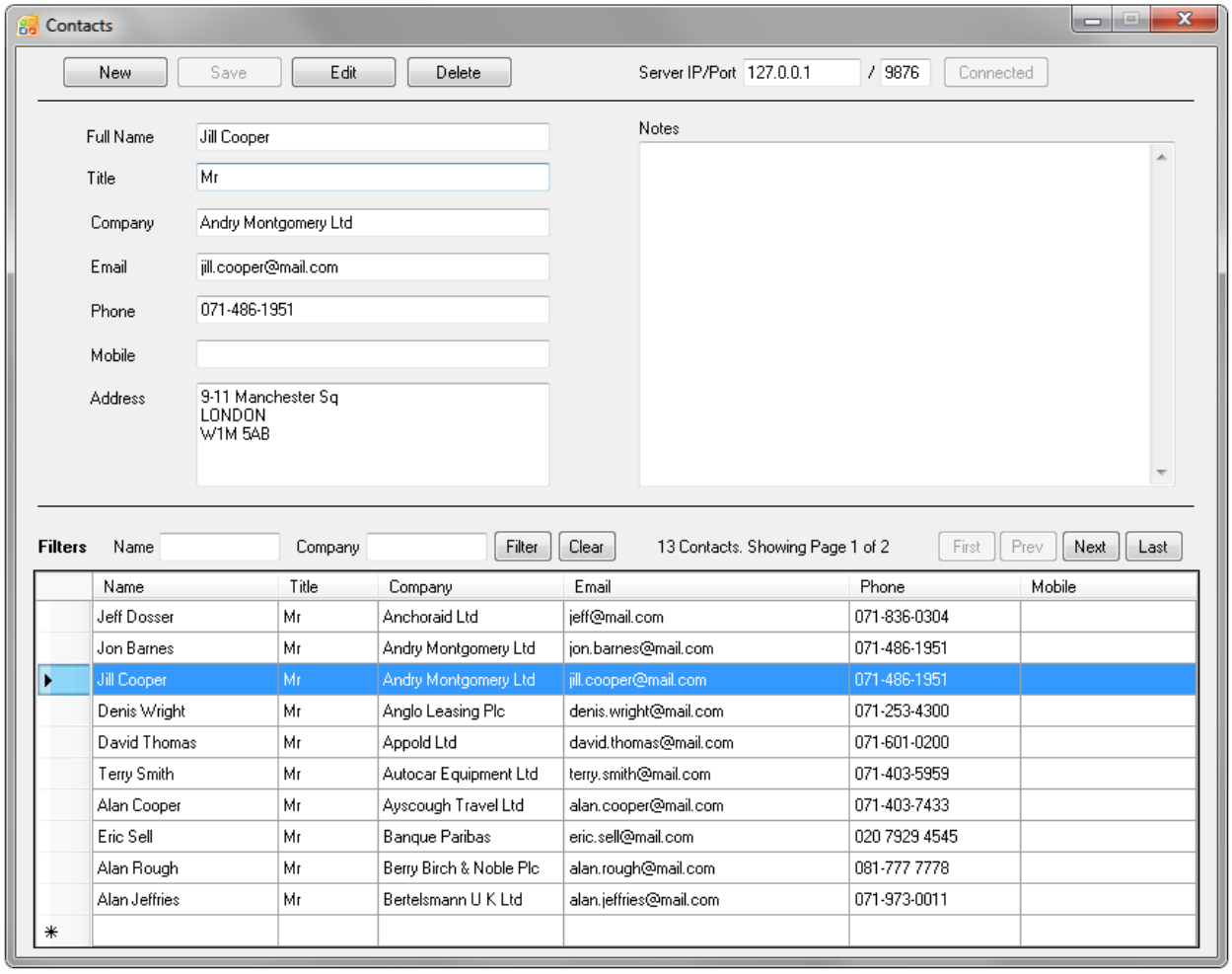
Press the "Save" button to save the changes you made and go back to the Main tab and press the Start button. This will re-start the ccDevnet Data Access Server.

You can close ccDevnet Data Access Server panel now. The server will continue to running in the background as service until you press "Stop" button.

Running the Sample Client Applications

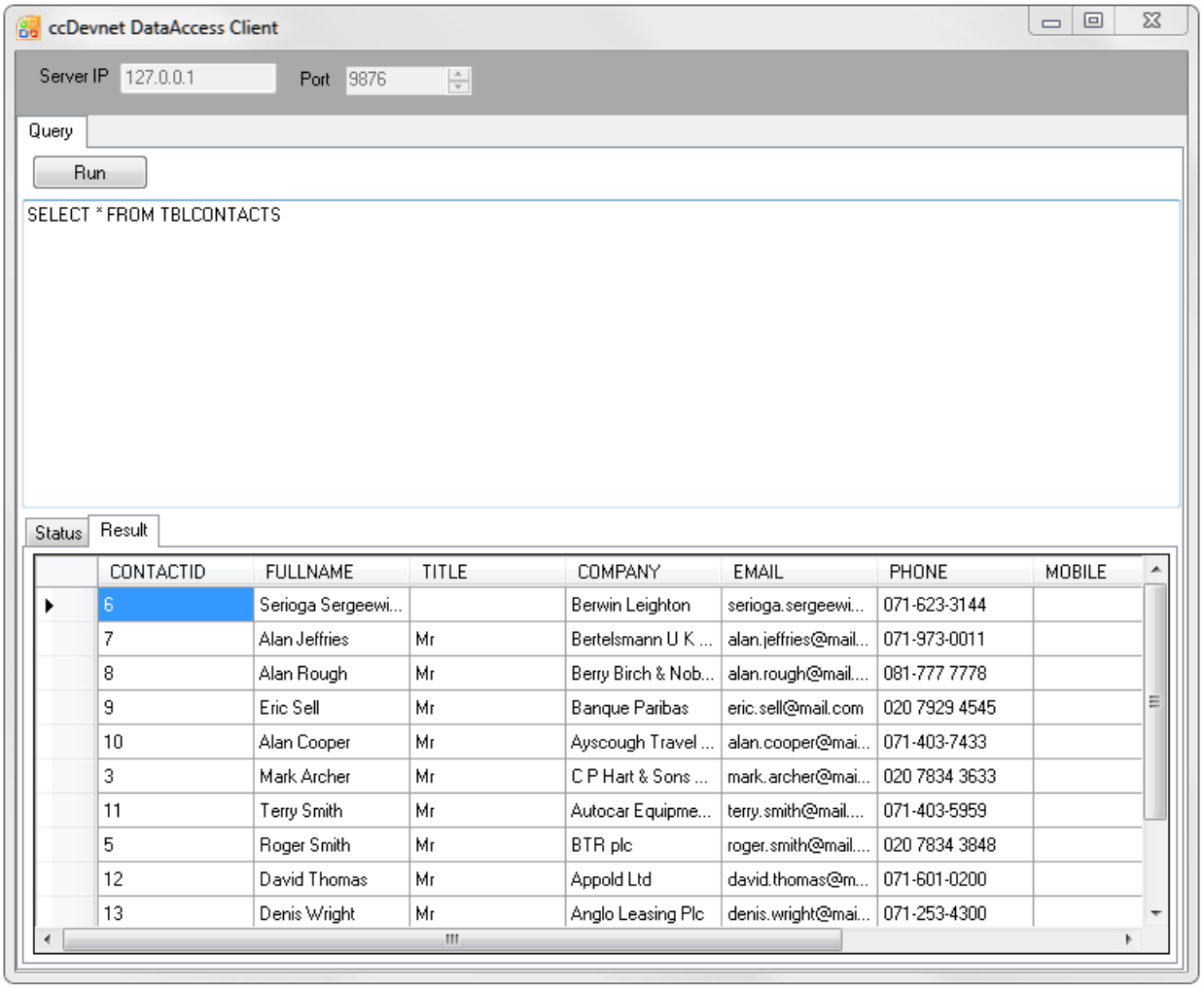
The full setup comes with three sample client applications with source code. You should find the following three application shortcuts on your desktop

Sample Contacts Manager



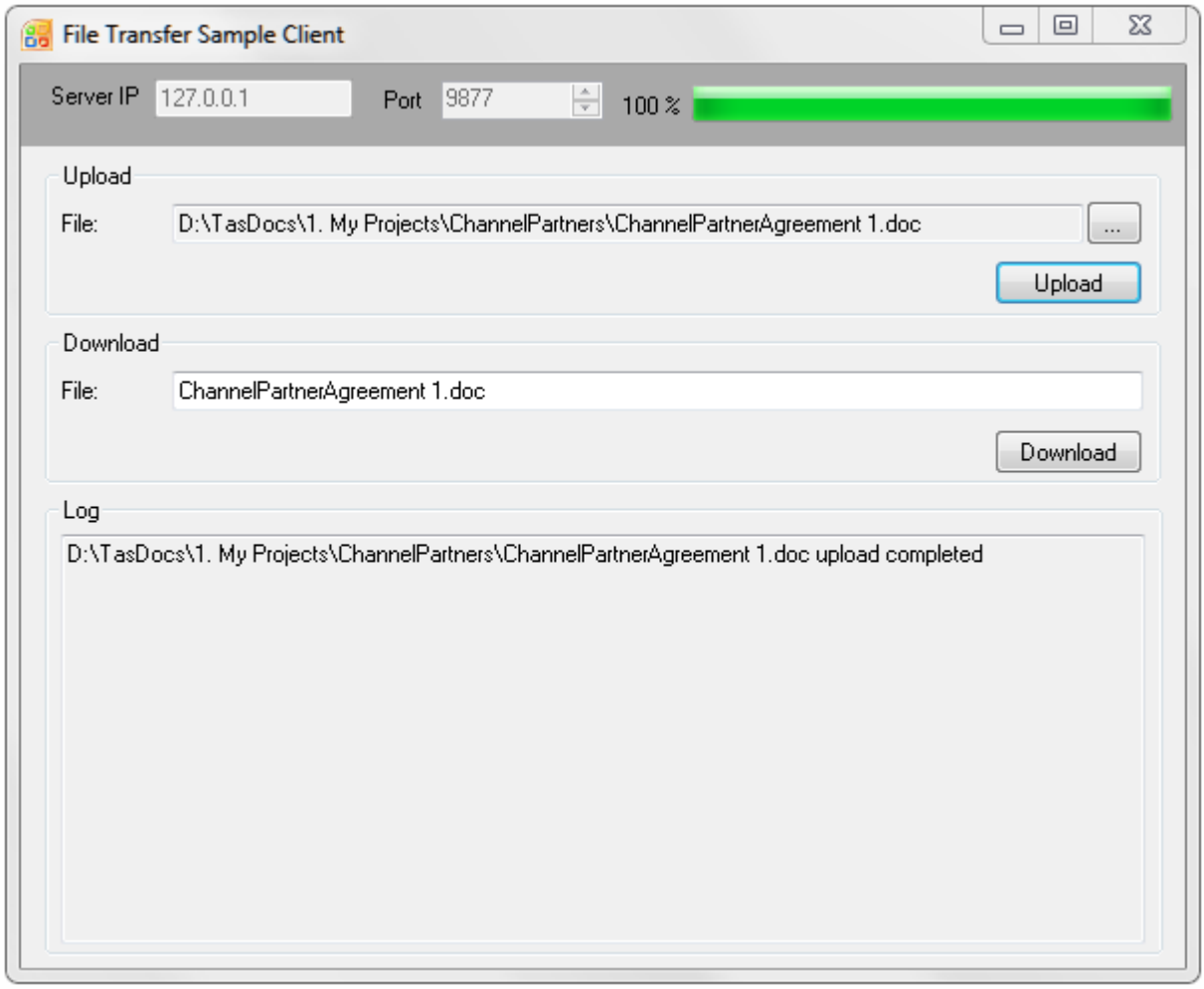
1. Ensure you have the correct details for Server IP and Port.
2. You can add, edit and delete records from the buttons on the top and filter from the bar above the grid.
3. Note that its worth comparing the performance against the server running on your own machine (ip: 127.0.0.1) and a cloud based server using the “Server Only Setup” to see how well the system performs across the cloud.

Demo Client



CONTACTID	FULLNAME	TITLE	COMPANY	EMAIL	PHONE	MOBILE
6	Serioga Sergeewi...		Berwin Leighton	serioga.sergeewi...	071-623-3144	
7	Alan Jeffries	Mr	Bertelsmann U K ...	alan.jeffries@mail...	071-973-0011	
8	Alan Rough	Mr	Berry Birch & Nob...	alan.rough@mail....	081-777 7778	
9	Eric Sell	Mr	Banque Paribas	eric.sell@mail.com	020 7929 4545	
10	Alan Cooper	Mr	Ayscough Travel ...	alan.cooper@mai...	071-403-7433	
3	Mark Archer	Mr	C P Hart & Sons ...	mark.archer@mai...	020 7834 3633	
11	Terry Smith	Mr	Autocar Equipme...	terry.smith@mail....	071-403-5959	
5	Roger Smith	Mr	BTR plc	roger.smith@mail....	020 7834 3848	
12	David Thomas	Mr	Appold Ltd	david.thomas@m...	071-601-0200	
13	Denis Wright	Mr	Anglo Leasing Plc	denis.wright@mai...	071-253-4300	

File Transfer Client



Sample Applications Code

See below for details of the sample clients' source code. Note that the source code for the server is only available with the "Corporate with Source" pack.

You will need Microsoft Visual Studio to be able to open the projects described below.

Note that as a client developer you will need to have access to a security key called the "passkey". This will be used as a part of the client class constructor as a parameter. For server side, see section on "Server Settings".

Developing a console based client:

We will start with a console based client.

1. Open Visual Studio, click on File > Open > Project and open the project file "DataAccessServer Projects\Samples\BasicClient\BasicConsoleClient.csproj" in My Documents.
2. Add reference to the data access server client library from the solution explorer see references. If you see reference "ccDevnetDataAccessLib" with a yellow icon next to it then remove it and

add again using the browse option with the following path "C:\Program Files\ccDevnet\ccDevnetDataAccessServer\Lib\ccDevnetDataAccessLib.dll"

3. Open Program.cs and explore code. The code is relatively simple; we created an object of client class and got data into a DataSet from the Contacts table. Try and run this application, this will display the received data in XML format.

Transaction based Windows client:

In this section we will explore the "Demo Client" application code. This is one you can run from your desktop.

1. You can find the relevant code project in folder "DataAccessServer Projects\Samples\TransactionsWinForm" in My Documents and do steps 1 and 2 for this project as we did in previous section for console client.
2. You can explore code for the main-form FrmTransactionClient.cs, try and run application.
3. Pass any query (select, insert, update, delete), it will show results of select query in a data grid.

File Transfer client:

We will now explore the code for the "File Transfer Client" application.

1. You can find the relevant code project in folder "DataAccessServer Projects\Samples\FTClient" in My Documents and do steps1 and 2 for this project as we did before for console client.
2. If you explore the code for the main-form frmMain.cs, you can see we created an object of FileTransferClient class and used async upload and download methods, also see code region "File Transfer Object" for event handlers.
3. Try and Run this application to test file upload and download.

Server Side Objects

In the previous section we looked at a client application for data access, where we passed SQL commands from client side code. In some applications this maybe be necessary, however it is not an ideal approach to have SQL Commands and Business Logic on client side. Best is to have all the database and business logic on server side and the client should call methods that are available on server side, e.g. client side code calls a method GetAllContects() instead of sending direct sql "select * from tblContacts". You can develop server side objects as shown below to facilitate this.

An object is usually a "dll" file having classes and methods, e.g. We can have "ContactsDAC.dll" a server side object which provides functionality and CRUD operations related to the "Contacts" database.

How to create a server side object:

To create a server side object make a "class library" type project, and we will get a "dll" when you build the project. This DLL is a server side object.

Deployment of a server side object is simple.

1. Open server dialogue and stop the server.
2. Copy your object (dll) and paste into folder "C:\Program Files\ccDevnet\ccDevnetDataAccessServer\Server\Objects".
3. Start server from server dialogue and the object is ready to use.

Creating a Test Object:

You can find sample code for Test Object in folder "DataAccessServer Projects\Samples\SampleObjects" in My Documents.

1. Open visual studio and open project by browsing file "TestObject.csproj"
2. Open TestClass.cs and explore the code. Here you will see the "SayHello()" method which returns a string. You can also explore other methods for example how to return a class is shown in method ClassReturn().

Creating a Test Object with Data Access (TestObjectDac):

You can find sample code for TestObjectDac in folder "DataAccessServer Projects\Samples\SampleObjects" in My Documents.

1. Open Visual Studio and open project by browsing for file "TestObjectDac.csproj"
2. Go to Solution Explorer and delete reference to "ObjectsCommon" if you see a yellow icon next to it and add reference again by browsing path C:\Program Files\ccDevnet\ccDevnetDataAccessServer\Lib\ObjectsCommon.dll
3. Open TestDacObject.cs and explore the code.

Suppose we want to use the two available features in this class from ObjectsCommon. Class constructor has 2 parameters a) dataAccess b) dataStore and they are being assigned to 2 local variables "_dataAccess" and "_dataStore" to be used in other methods within the class.

a) dataAccess is used for all types of data operations (CRUD operations) with the database.
b) dataStore is used as a server cache. This is a key, value based mechanism.

Each class constructor can have 0 to 2 parameters depending on your requirement.

Explore the method GetContacts() which shows sample usage of dataAccess and dataStore. It tries to find contacts DataSet from Server's cache, if the DataSet is not found then it gets from the database and adds it to the server cache for faster requests in the future.

Sample Usage of Server Side Objects

In the previous sections we created 2 objects "TestObject.dll" and "TestObjectDac.dll". Now it is time to utilize their functionality in a client application.

1. Open Visual Studio and open project by browsing for file "DataAccessServer Projects\Samples\SampleObjects\SampleUsage \ClientDevTests.csproj" in My Documents.
2. Open solution explorer and delete reference ccDevnetDataAccessLib if you see yellow icon next to it and add reference again from "C:\Program Files\ccDevnet\ccDevnetDataAccessServer\Lib\ccDevnetDataAccessLib.dll" and build.

3. Open program.cs and explore code. First 2 lines shows that we have created object of Client class and have got data in the dataset.

We have already practiced this in previous sessions. We will now see how to utilize server side objects. There are 3 approaches.

a. Utilizing server objects by Classic approach
   See region "Server Objects Classic" for sample code. We can call static methods via CallStaticServerObject() method, create server object via CreateServerObject() method and call non-static methods via CallServerObject() method.
   Note that, we are passing namespace.class_name when we create server object (this is common in all 3 approaches)
b. Utilizing server objects using Duck typing
   See region "Duck Typing Example" for sample code. Here we create server object with method CreateServerObjectProxy() and call methods directly with object name. But duck typing doesn't give visual studio intellisense.
   If you are not familiar with duck typing, you may google "c# duck typing"
c. Utilizing server objects by Interface proxy (**Recommended and Easy to use**)
   See region "InterfaceProxyExample" for sample code. Here we create server object with method CreateServerObjectProxy<T>() and call methods with object name with Visual Studio intellisense support. Here "T" is an interface. In this case interface is "ITestDacObjectClient" which defines the structure of 1 method to be used. We will use this approach in our Sample Contacts Manager app which is described in next section.

Note: Interface proxy can be generated with our "Objects Proxy Generator Client" tool, which you can find this on your desktop.

Exploring Sample Contacts Manager

In this section we will look at the code for the "Sample Contacts Manager" application. The application icon is available at your desktop.

Contacts Manager Application has 2 techniques to pass on and get back the data. Technique1 gets data by passing SQL queries to the database via Data Access Server. Technique2 gets data by calling server side object.

1. Find code in project "DataAccessServer Projects\Samples\ContactsSample\ContactsSample.csproj" in My Documents and repeat step1 and 2 from previous section. Then build the solution, try and run application.
2. Technique1 has main form frmContactsSample1.cs which uses DAC.ContactsSample1 class for all data communication with data access server by passing direct SQL queries.
   Note! that in DAC.ContactsSample1 class we have a "dbType" class level variable which is used in the GetContacts() method to tell which database is being used with data access server, and in GetContacts() we used some SQL which is specific for each db type. So "dbType" is useful here to check db type and make condition accordingly.

3. Technique2 has main form frmContactsSample2.cs which uses DAC.ContactsSample2 class for all data communication with data access server by calling methods of server side object "ContactsDac.dll"

In DAC.ContactsSample2 class we are utilizing server side object by Interface Proxy method, which is described as point c) in previous section.

Sample2 uses server side object "ContactsDac.dll". Its code can be found in project "DataAccessServer Projects\Samples\SampleObjects\ContactsDac.csproj" in My Documents, and this is a similar concept to the one we talked about in the previous section "Creating a test object with Data Access (TestObjectDac)".

Registering your Data Access Server via an API

If you are making a product based on the Data Access Server, you might not want to expose your license key to your clients (eventual product users).

In this case Data Access Server will be installed as a part of your product's setup. When an end user installs your product, the Data Access Server will also get installed with your product and it will register itself for that installation.

This is how you can do that with our API:

1. Find the following dll and add reference to your project "C:\Program Files\ccDevnet\ccDevnetDataAccessServer\Server\ServerConfig.dll"
2. Call RegisterThisInstallation() static method to register for that install, its signature is: bool ServerConfig.ServerRegistration.RegisterThisInstallation(string key, string name, string location)

Parameters details are:

- **key**: your license key
- **name**: your customer's name or their physical server name
- **location**: your customer's city or country

Note: Master registration must be done before calling this method, master registration is your own (or 1st) registration for Data Access Server where you enter your own info and set username/password for your ccdevnet data access server web control panel. From www.cc-devnet.com login you will be able to see all installations that your customers have done, and see how many installations you have left (if you don't have a unlimited installs license).

Data Access Server for Other Platforms

Introduction

Data from the Data Access Server can be accessed and updated from almost any platforms including Windows, MAC, Linux and mobile platforms iOS, Android, and Windows etc. using the RESTful API. For more information on how the RESTful service works, visit http://en.wikipedia.org/wiki/Representational_state_transfer.

How to Enable

RESTful service can be enabled from Data Access Server Settings tab with given port to run service on.

Note that as a client developer you must know a security key called the “passkey” and you should pass this as a request header with each request.

passkey:DemoKey

For server side see section “Server Settings”.

How to Use

Data on “Data Access Server” can be accessed with uri. A typical uri format is http://ServerAddress:Port/db/table

For example, if we want to get the Customers table from DAS (Data Access Server) running on port 9878 at address localhost, request uri will be

http://localhost:9878/db/Customers

This uri will return data from Customers table.

Data Format

Data Access Server supports both xml and json format to retrieve and post data. But it is recommended that you use the JSON format for better performance and reliability. To read more about json format, visit http://en.wikipedia.org/wiki/JSON

Data Access Server auto detects client type based on the request and returns data in the required format. If client can accept both formats (Browsers) then data format will be xml. If browser supports JSON format only (mobile devices) then data return format will be JSON. User can manually specify which format he/she wants by specifying “Accept” and “Content-Type” header parameter of request.

For Example to get data in JSON format,

Set request header Accept: application/json and to send data as JSON, set request header Content-Type: application/json

For mobile apps, default format is json and developers don't need to mention format manually.

CRUD Operation:

CRUD (Create Read Update Delete) operations can be performed on database table via the RESTful API. Here is short description of the RESTful methods.

GET:

Fetch or get data from database table.

POST:

Create new record.

PUT:

Update an existing record.

DELETE:

Delete an existing record

Examples:

Here is a list containing few examples of RESTful API usage

Get contacts table from examples db (supplied with this setup)

http://localhost:9878/db/tblcontacts/

Get single contact with id = 10

http://localhost:9878/db/tblcontacts/?where=contactid%3D10

Get email of contact with id=10

http://localhost:9878/db/tblcontacts/?where=contactid%3D10&select=email

Get contacts table order by name

http://localhost:9878/db/tblcontacts/?orderBy=fullname