



Getting Started with BrightXpress



Table of Contents

Introduction.....	3
Installation.....	4
User Interface.....	4
Projects Panel.....	5
Servers Panel.....	5
Workspace.....	6
Output	6
Project Overview	7
Bright Software Project Design Concepts	8
Forms	8
Tables.....	9
Queries	10
Sync Rules	11
Additional Elements.....	11
BrightServer Design Concepts.....	12
Sync Points.....	12
Scripts	13
Mappings.....	13
Tables.....	14
BrightServer Runtime Concepts	15
Server Runtime Settings	15
Licenses.....	15
Users	16



Introduction

The **BrightXpress Mobility Suite** is a flexible, modular, robust and scalable software development platform for creating end-to-end enterprise mobile business mobile solutions. It is consisted of an easy-to-use drop mobile application builder, a scalable middleware server product for integrating and mobilising backend system to remote field users and a powerful mobile device engine for providing field users with natural easy-to-use mobile applications on various types of device platforms.

BrightXpress Mobility Suite is an easy-to-use mobility enabling platform solution that allows mobile workers to connect to the organisations back-end information systems. It provides simple and efficient ways to transform paper-based systems and manage data flow to and from the field.

BrightXpress Mobility Suite provides a hassle free environment without the complexity needed to integrate with enterprise infrastructure environments.

BrightBuilder™ is the mobile application designer. It is the integrated development environment for designing, developing and deploying mobile applications. **BrightBuilder™** is also the central management console for managing and maintaining mobile solutions. Using **BrightBuilder™**, you can create mobile applications running on the remote field devices, and create and manage server configurations to mobile your business data to the field workers.

BrightForms™ Mobile Device Engine is the virtual forms engine running on the field devices and is simply the front-end of mobile solutions. It provides abstraction between mobile applications and operating systems, device hardware, and peripherals. **BrightForms™** mobile applications are guaranteed to work on every platform where **BrightForms™** is available without modification. It provides a rich occasionally connected mobile application experience interfacing to barcode scanners, cameras, onboard or external GPS devices, magnetic card readers, mobile printers etc. With the built-in synchronisation engine, it can communicate with the backend systems helping users in the field access business data and collect and transmit the field data securely back to the backend systems.

BrightServer™ is a flexible mobile data gateway designed to handle complex business mobility requirements. **BrightServer™** enables **BrightForms™** powered mobile devices to connect with server-side data sources including powerful relational databases or file based data sources at the backend. **BrightServer™** can reach the data sources, whether they are databases or flat files, via its synchronisation engine, and can detect the changes in the data and send the new changes to remote field clients. **BrightServer™** does not require a store-and-forward database. Your mobile solution data sources can be as simple as a flat CSV files, Microsoft Access database, or as complex as large relational databases such as Microsoft SQL Server, Oracle or DB2. Further you can use the JavaScript based scripting mechanism to create your own sync sources and destinations, for instance, to read and write to web services and integrate into legacy systems.



Installation

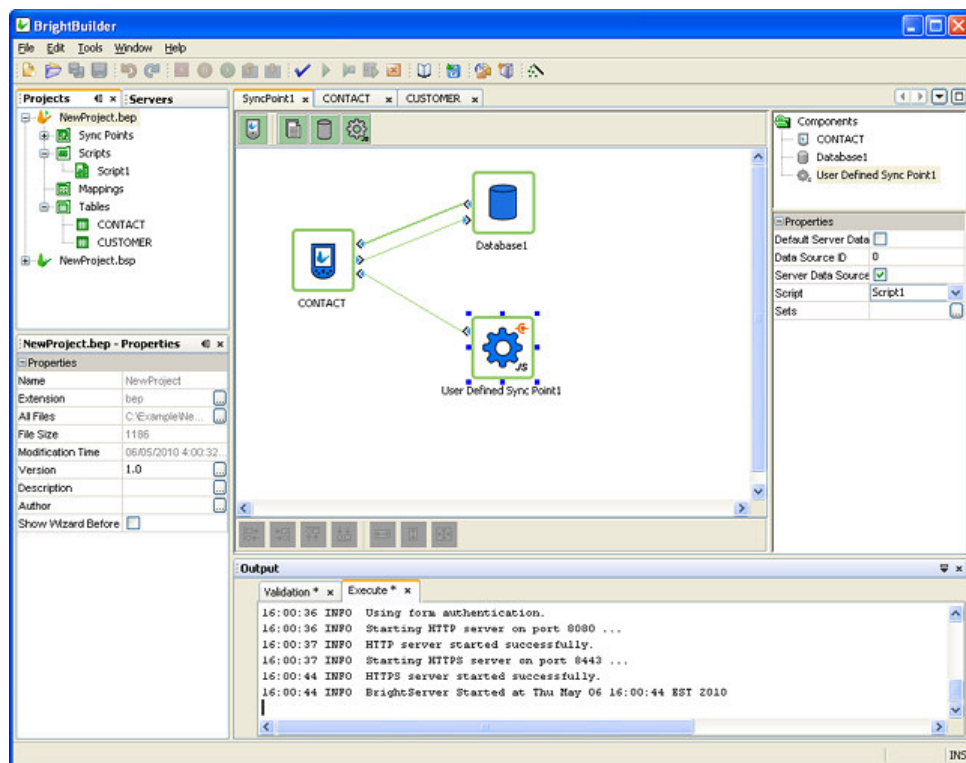
The **BrightXpress Mobility Suite** comes with both **BrightServer™** and **BrightBuilder™** installations. This is advantageous especially for development because you can start and stop BrightServer within the **BrightBuilder™** IDE and fix any issues with the configuration before production.

BrightForms™ may be installed bundled with **BrightBuilder™** or separately installed on the desktop. Installation is also available for mobile devices, either locally or via SD card.

User Interface

BrightBuilder allows you to create powerful mobile applications via BrightForms BSP projects and flexible server configurations via BrightServer BEP projects quickly and effectively.

Below is the image of BrightBuilder IDE for a BrightServer project which is divided into several windows namely, Project, Properties, Editor and Output.



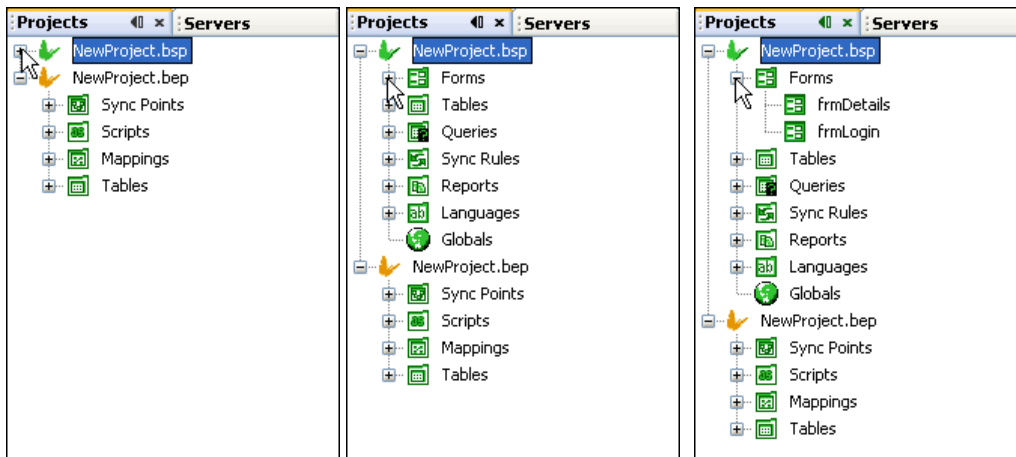
To create an end-to-end mobile solution, using BrightBuilder, the following two types of projects need to be created.

- A BrightForms project (BSP) that defines the mobile application running on field devices,
- A BrightServer project (BEP) that defines, in simple terms, the data sources that the remote field devices will be sending data to and from

Starting BrightBuilder for the first time, please note the standard toolbars and menus of the interface which allow standard operations such as opening, saving and creating these project files. You may also notice a number of panels in the IDE, each serving a purpose in application development, which will be described below.

Projects Panel

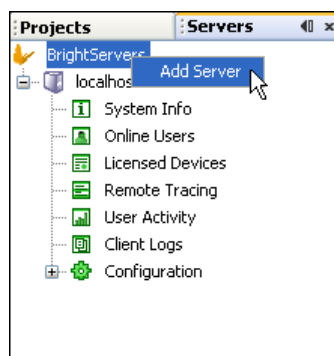
The projects panel manages the currently opened BEP or BSP projects. In this panel, projects are represented in a tree view, and may be expanded to reveal the components they are comprised of. These nodes may then be expanded to reveal each element individually, then in turn be opened, removed and have other management operations performed in them via the right-click context menu. Opening an element results in the element being accessible and editable in the workspace.



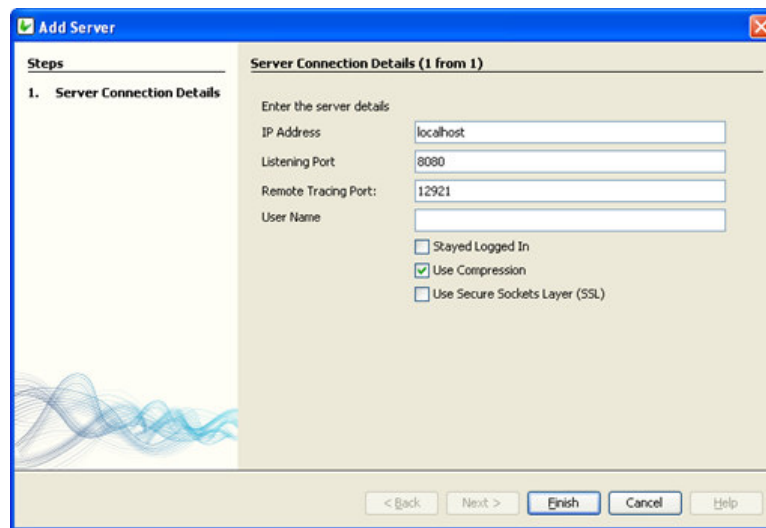
The projects themselves may also have a number of operations which may be performed via this menu or while selected, such as executing Bright Software Projects, or Starting and Stopping BrightServer Projects.

Servers Panel

Similar to the projects panel, and located next to it by default, the Servers panel handles runtime configuration elements of BrightServer instances. Here, BrightServer instances may be monitored and configured, with components organised in a similar fashion to BEP and BSP projects. However, instead of opening files and configuration designs, servers must be connected to.



When a server is run locally and configured from the projects panel, it is automatically added to this tree. Alternatively, remote servers may be accessed via right clicking the parent 'BrightServers' node of the tree and selecting 'Add Server'. From here, the add server dialog may be completed, and the remote server and its components accessible.



Workspace

Clicking on an element will open it up for editing in the working space. The tab screen will vary from element to element, but it is important to note that any changes in these tabs will reopen in their previous state if closed, provided the project is open. However, if the project is closed and isn't saved, any changes will be lost. This applies to both Bright Software and BrightServer projects, but not BrightServer instances as they do not operate via file formats; configuration must be downloaded and uploaded accordingly.

Output

This panel displays messages to the user, namely, the validation of the project. It is used to confirm whether the design definitions in either type of project are correct, and thus if it will run successfully on either BrightForms or BrightServer.

Project Overview

As mentioned previously, an end-to-end mobile solution using BrightBuilder consists of the following two types of projects needing to be created.

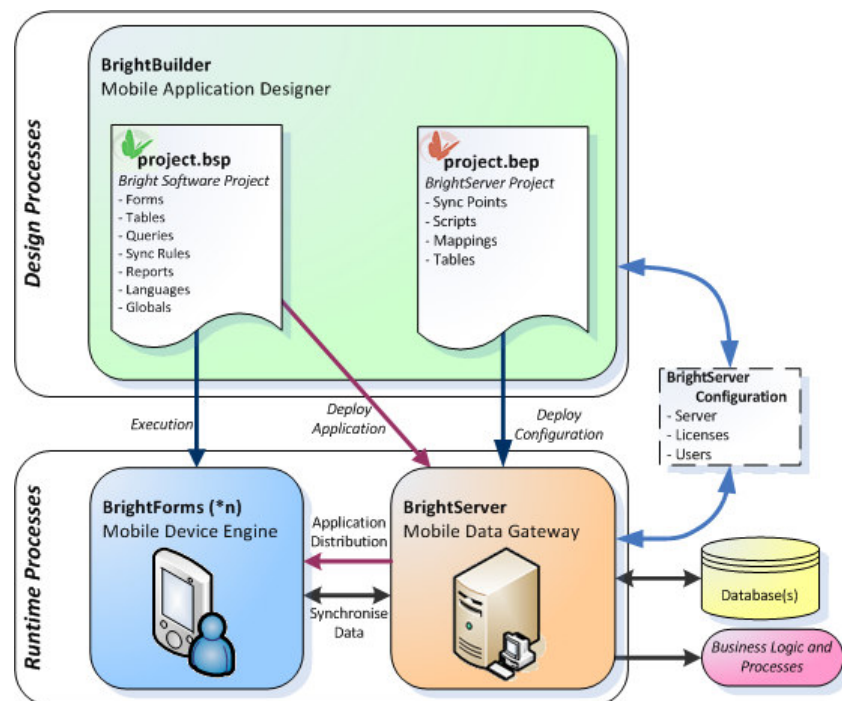
- A BrightForms project (BSP) that defines the mobile application running on field devices,
- A BrightServer project (BEP) that defines, in simple terms, the data sources that the remote field devices will be sending data to and from

These two projects represent the design portion of using BrightBuilder. When designing a project, there are two processes to be made aware of, the **design process**, which defines how the application will behave, and the **runtime process**, the processes involved when these files created in the design process are executed. Design choices need to be made such that execution processes may be executed smoothly.

BrightBuilder handles all of a BSP's design processes; in the case of an application, the data creation, retrieval and manipulation all take place in BrightForms, and constitute runtime processes.

However, for BrightServer projects, you need to manage the runtime configuration elements using BrightBuilder. This means there are design time settings which include sync points, scripts, tables and mappings, and runtime settings which include server settings, licenses and users configuration. The design time settings are configured via a BrightServer BEP project, and the runtime settings are configured online through BrightBuilder Servers panel by connecting to a server instance.

The diagram below explains how these processes relate to one another, and how they execute.



As seen above by the dark blue arrows, the BSP and BEP files are designed and deployed in BrightBuilder, and run on BrightForms and BrightServer respectively.

BrightBuilder may also deploy project BSPs directly to BrightServer, which allows the server to distribute application updates and any changes that need to be made. This is shown in magenta above.

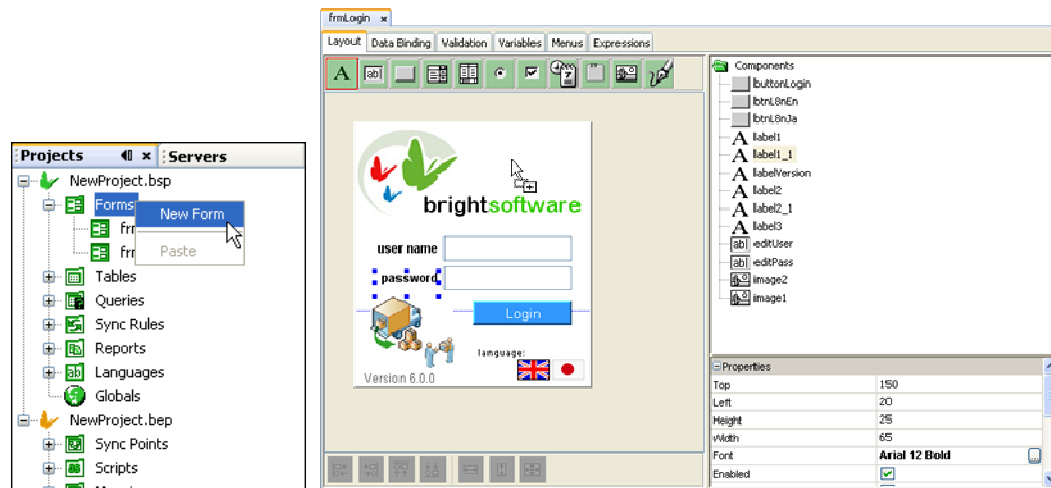
The light blue arrows also demonstrate the uploading and downloading of the server configuration files by BrightBuilder to BrightServer.



Also to be noted is the data processes, marked in grey, are runtime processes only. This data consumption and transmission between server and device is conducted based on the sync rules, queries, tables and forms of the BSP, while the external sources interfaced by BrightServer is defined by sync points, mappings, tables, and scripts in the BEP. For more information about both these BSP or BEP design concepts, please refer to the guide below.

Bright Software Project Design Concepts

Forms



A form is the heart of all graphical operations in Bright Applications, and in these applications, all data creation, retrieval and manipulation is handled through form operations.

Forms mainly perform these operations visibly via **controls**, which are used to store and capture data. They also facilitate the process, such as with labels. A form may be comprised of several types of controls, each performing different roles and functionality in a given application.

When editing a layout, the green **Controls Toolbar** shows all the controls available to use in the form, and you can use BrightBuilder's drag and drop interface to add them to a form. Once added, they may be selected, moved, and have their properties altered to meet the requirements of the application.

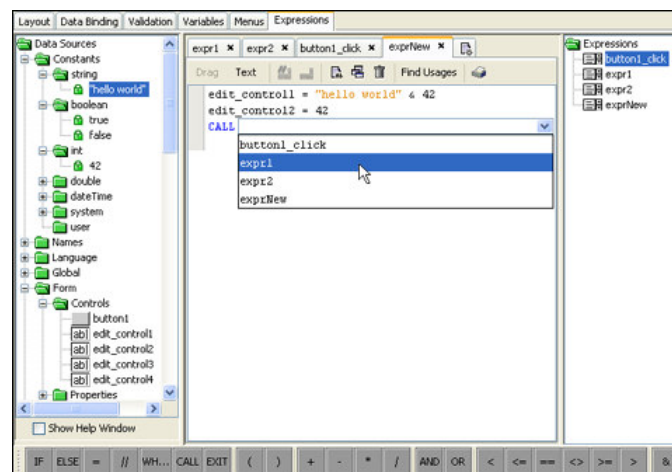


Below is detail regarding each control's function:

- **Labels** - used for static text appearing on the form.
- **Edit Controls** - enables the user to view, type, or edit text in a field.
- **Buttons** - initiates an action when clicked.
- **Combo boxes** - allows the user to select an option from a list and/or (if enabled) type text into the field. The list can be configured to be fixed or dynamic.
- **Listviews** - displays multiple rows of data, and can initiate an action when clicked or double-clicked. For viewing/selecting from a record set based on a query.
- **Radio buttons** - used for choosing one of a set of mutually exclusive attributes (e.g. selecting gender, male or female)
- **Check boxes** - allows the user to select or decline an option. Turns on/off a Boolean attribute.
- **Date Time pickers** - used for selecting a date (and/or) time.

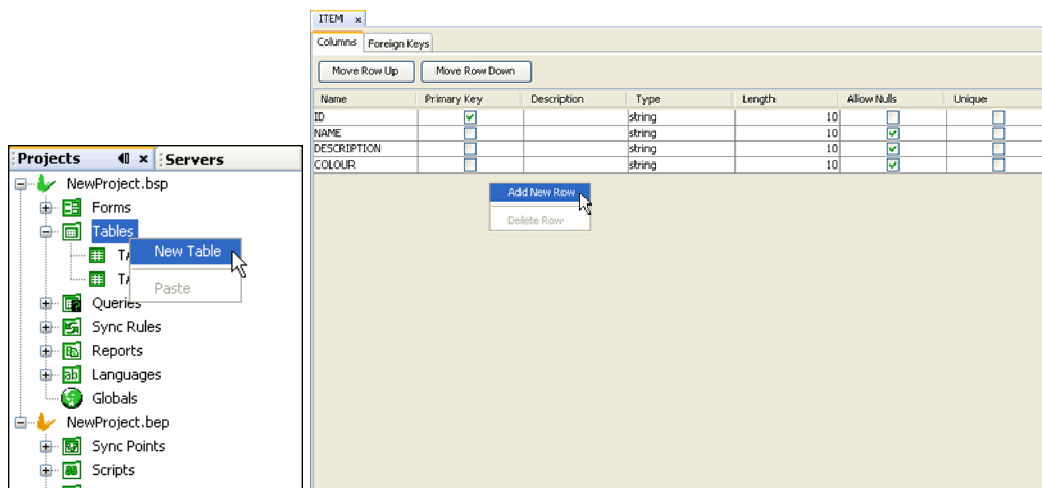
- **Group boxes** - used to visually group a set of controls on the form.
- **Bitmaps** - displays images in the layout.
- **Scribble** - accepts and displays scribble images that can be loaded with bitmap images and can also save these bitmaps with user (free) input.

Controls are visible forms of data sources in forms. They may both store and hold data, and this process is handled on runtime via another form feature; **expressions**. Expressions enable the manipulation of data sources within forms, and are executed at runtime. They may also access external objects, such as the **Database** and **Synchroniser**; used for data transmission and consumption to BrightServer.



Expressions make heavy use of the **data source tree**, shown to the left in the above picture, which represents what data is accessible to the form or current expression. As there is a finite set of data available for each form, the expressions and data source tree features of BrightBuilder utilise a drag and drop interface, providing a quick, yet powerful development environment for application design.

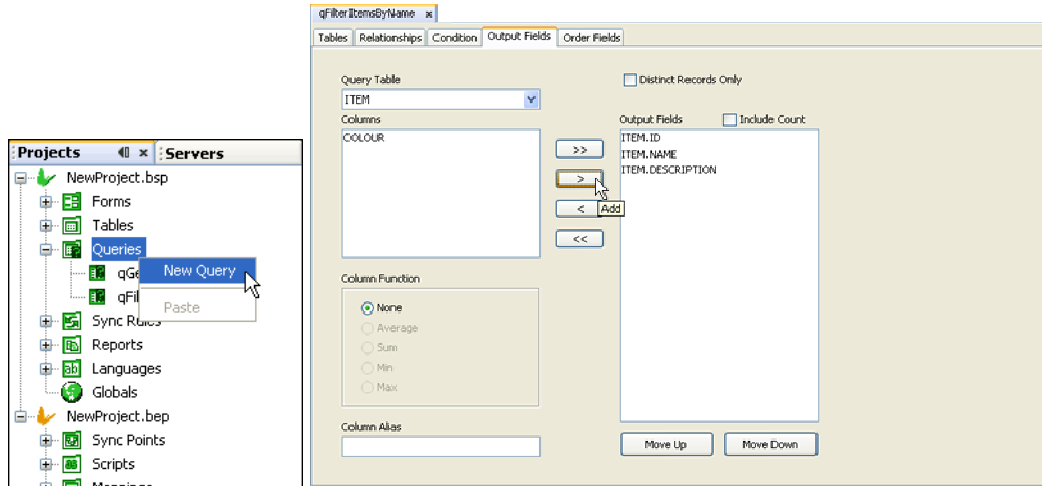
Tables



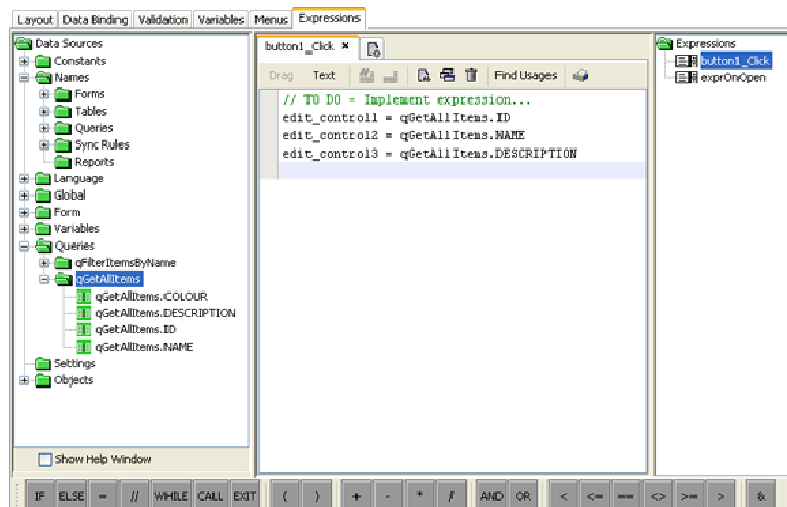
Tables model data to be stored by the application, namely the **fields** and **types** for the data to be recorded. When editing a table, the right click context menu is used to create rows, and from there the columns of each row may be changed. Note, that each record must be unique in order for queries and other operations to perform successfully, and thus a **primary key** must be defined for each table to facilitate this.

Tables typically have their values inserted via form expressions, using the **Database** object, or via data binding.

Queries



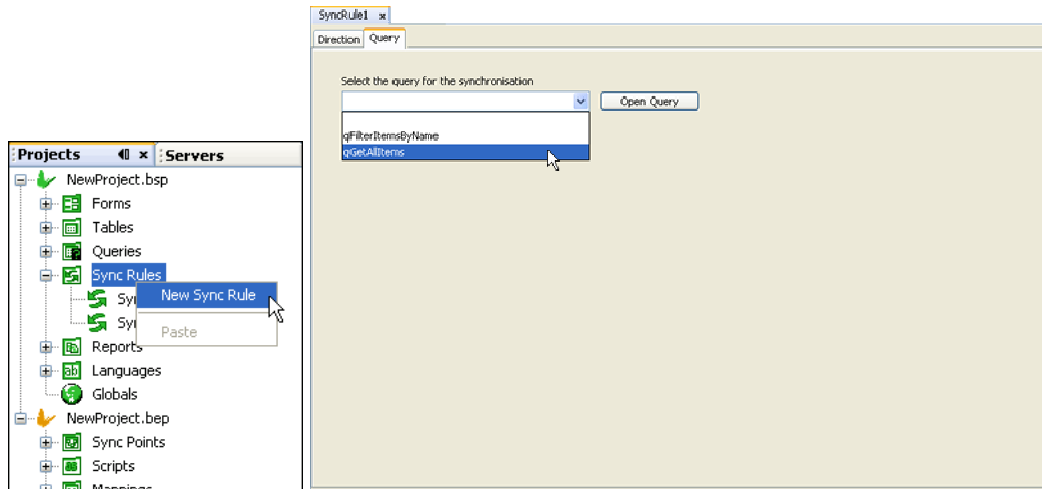
Queries get executed on runtime to return a specific **record set** from the database in that particular point of time. Once set, they may be referenced or even linked within forms themselves, enabling the forms access to the data to which they supply. This may be handled via several means; either via data binding or through **expressions** and the data source tree.



Queries also serve a role during the synchronisation process, typically allowing sync rules to send and receive the records which the query applies to. These records are then analysed by BrightServer with database updates made accordingly.

Queries may be defined as **simple**, **advanced**, or even **server** referenced, and this type changed via each query's property. They can retrieve records over one or many tables, specified by the **table(s)** the query uses, and the **relationships** between them if more than one exists. Queries may also be able to provide **conditional** filtering on sets of records, either on results via conditions or simply by the **output** fields.

Sync Rules



Sync rules define the synchronisation of a set of records to or from a server, based on the **direction** of the rule. A client to server sync rule will have the BrightForms device send records for the server to consume, and likewise, a server to client sync rule specifies the server sending records for the client to consume.

Depending on the query selected, more information may need to be defined for the sync rule to execute correctly. The server must also support the direction of synchronisation, defined by configuration of its **Sync Points** panel.

Sync rules are handled and executed via Project **expressions**, more specifically, the **Synchroniser** object and its methods in the **data source tree**.

Additional Elements

Forms, Queries, Tables and Sync rules are necessary for a fully functioning connected system, but there are also several additional project element types that may be utilised, depending on the project developed. They are as follows:

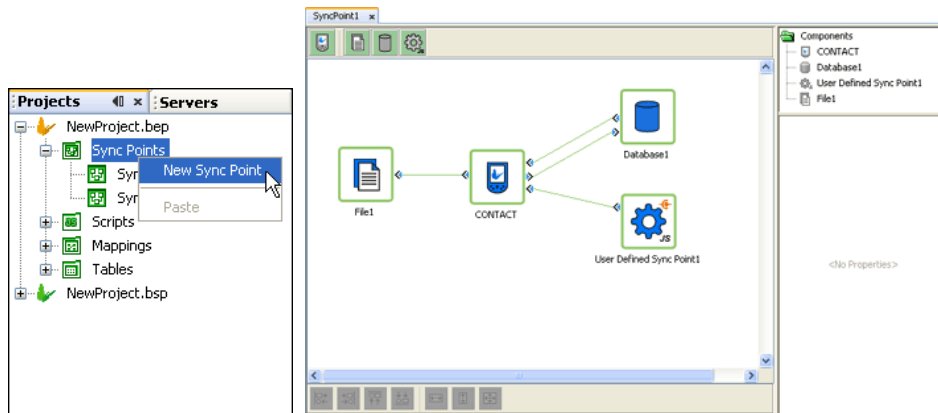
Reports - used to format data in a printable document, with printing executed by forms. The formatting of reports is very similar to the layout of a form, and is used to provide a formatted document which reflects the data of a system in a given state.

Language - The languages node is used to enable developers to localise and thus cater their projects to different markets/locales through the language table, which allows the switching of language depending on environment, yet having the project retain all functionality in the process.





Global - The global node allows the sharing of data project wide, without the containment feature inherent of forms.

BrightServer Design Concepts

Sync Points



Sync Point elements define the data flow between BrightServer and BrightForms. Each Sync Point element is a workspace to graphically layout the data flow components. The following are the different components in the Sync Points element.

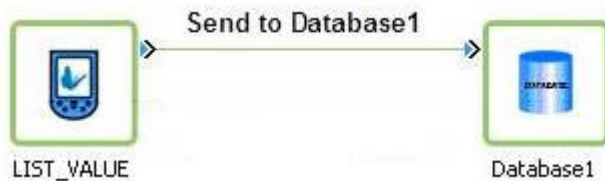
Icon	Description
	SyncPoint – Defines a table which BrightForms synchronises to or from BrightServer. A table must be selected for each syncpoint control dragged to the workspace.
	File – Represents the physical file to read or write data. This has a Sets properties that defines the path and file name and also the mapping between the Bright table and how it will be written or read to/from the file.
	Database – Represents the actual physical database that BrightServer will connect to and interact with. There are two properties that need to be setup, namely Database and Sets. The Database property defines the connection settings for the data source. And Sets defines the mapping between the Bright tables and the actual tables on the data source.
	User Defined Sync Point - Represents a server side script which handles the a table's data sources. This is defined and supported by a generic scripting framework in Javascript in BrightServer.

The direction between a SyncPoint and a Data Source determines whether the SyncPoint is a Sync Source or a Sync Destination.

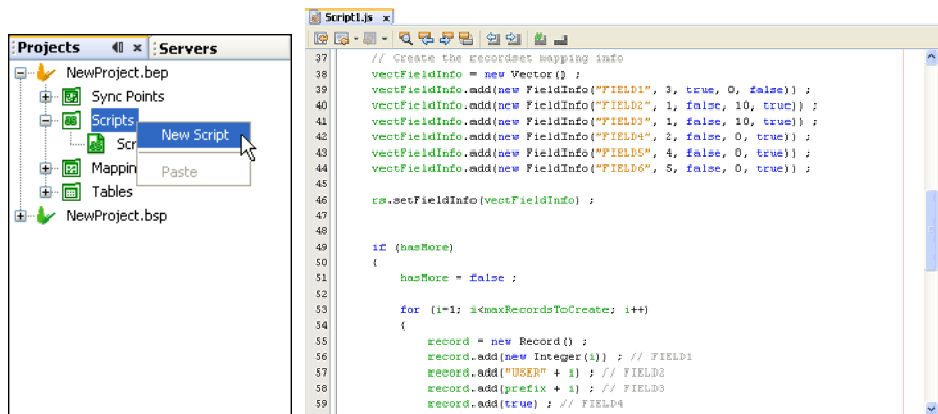
A Sync Source represents a table which BrightForms can synchronise data FROM. A Sync Source is defined when a Data Source (either a file or an database) is connected to a Sync Point.



A Sync Destination represents a table which BrightForms can synchronise data TO. A Sync Destination is defined when a Sync Point is connected to a Data Source (either a file or an database).

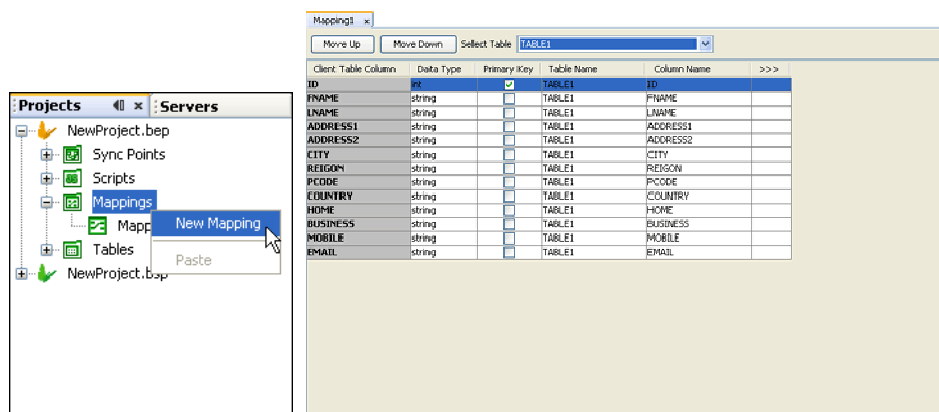


Scripts



The purpose of scripts is to allow user defined data sync points to be implemented using JavaScript. Scripts can be used to implement a sync source, or a sync destination. Using scripting, scheduled jobs can also implemented. The scripting is generally used when the integration point is other than files or databases. Scripts can provide very flexible and powerful server functionality. However it is intended for advanced users with some degree of programming experience.

Mappings





The purpose of a mapping is to map data from a data source to a table defined in BrightServer which in turn maps to a table BrightForms. Mappings define how data is arranged in a data source and how it represents records that BrightForms consumes.

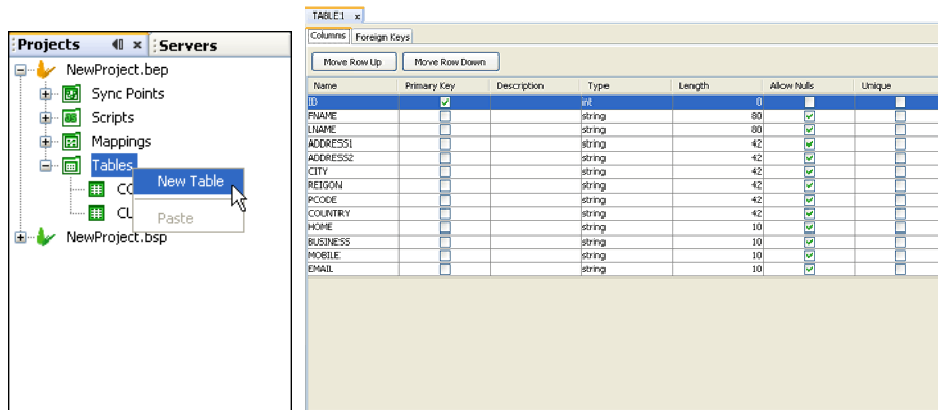
In other words, mappings are used to take the raw data from data sources such as files, and convert that data into a structured format, so that it can be made available to BrightForms to synchronise.

A mapping can be one of three types:

- Query Mapping - Maps a BrightForms table to a table in a relational database
- CSV File Mapping - Maps a BrightForms table to a file containing character separated values.
- Fixed Length File Mapping - Maps a BrightForms table to a file containing fixed length values.

Tables

The Tables element is where all the table definitions are managed.

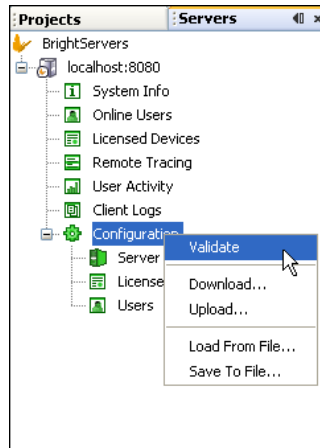


A table is composed of columns and rows and is defined by the column names, data types and table constraints.

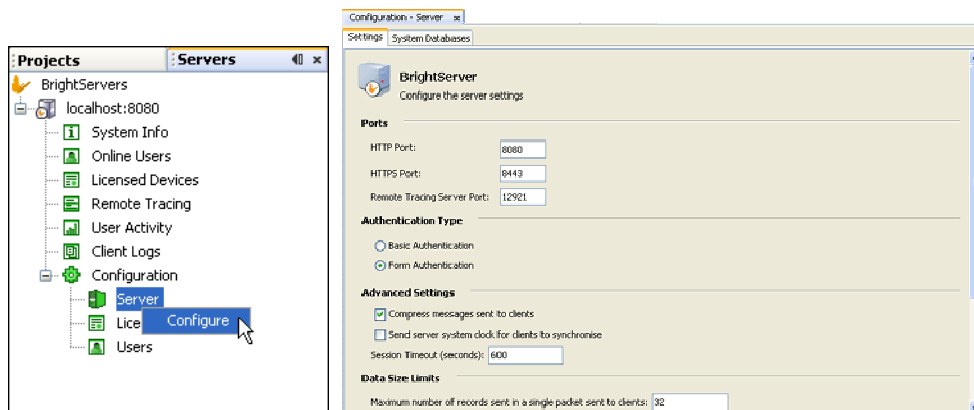


BrightServer Runtime Concepts

A BrightServer instance may be accessed and configured via the Servers panel in BrightBuilder. Here, you may check several server settings, and also download and upload configuration details to the server on the fly remotely.



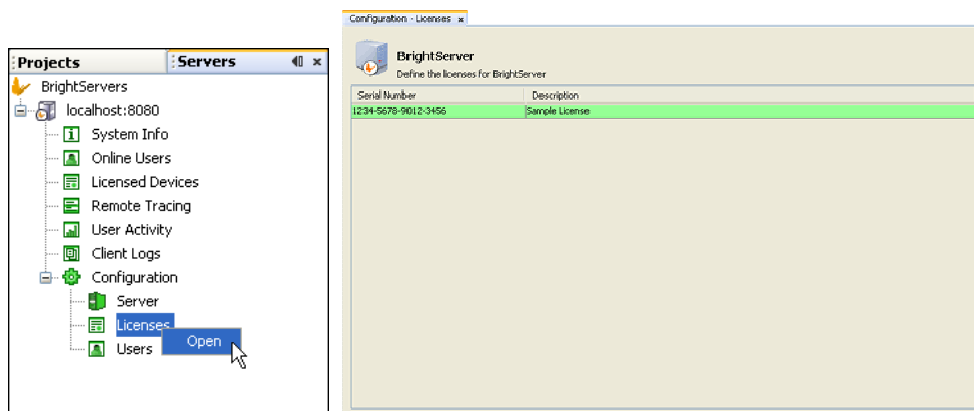
Server Runtime Settings



The server element allows users to configure the BrightServer server settings.

Server settings for new projects are set to default values, which runs smoothly out-of-the-box.

Licenses





The licenses element allows license details to be entered i.e. a serial number and a license file. It is possible to have multiple serial number and license file combinations for license upgrades when the project grows over time.

If there are no valid license specified, then BrightServer will operate in demo mode. BrightServer operates in demo mode for only 30 days and will only allow a single client connection.

Users

The users element provides the interface to create and manage users and the application distribution process. Because BrightServer only permits access to the backend data for authorised users.

The minimum requirement for BrightServer to run on default settings is to have at least one user as show below.

The screenshot shows the BrightServer configuration interface. On the left, a tree view under 'Servers' shows the 'Users' folder selected, with a context menu open showing 'Open', 'Copy All Users', and 'Paste'. On the right, the 'Configuration - Users' window is displayed, showing the 'BrightServer' logo and the text 'Define the users and groups for BrightServer'. The window has tabs for 'Users' and 'Groups'. Below the tabs is a button 'Release Applications to Multiple Users...'. A table lists the current users:

ID	Name	Password	Active	Group	Application	Version
-2	bsadmin	changeit	<input checked="" type="checkbox"/>			-1
1	User1	User1	<input checked="" type="checkbox"/>			-1