NetObjects Fusion Connector for Microsoft Active Server User Guide

The NetObjects Fusion™ Connector for Microsoft® Active Server consists of several custom NetObjects Java components (NFX) which act as design-time controls within NetObjects Fusion. These components are designed to provide both input and output access to databases through Microsoft Active Server Pages (ASP). With them you can add live database connectivity to your web site and allow users to view and update data instantly. For instance, you can design a Human Resources application that allows employees to see how many vacation days they have left and allows them to update personal information such as their address and phone number.

The ASP components provide a different kind of database access than NetObjects Fusion's built-in data publishing capabilities. The data lists and stacked pages created by the built-in facilities are static. Once published to a web site, the data doesn't change until you republish pages using new data. The ASP components actually generate code that runs on a Microsoft IIS web server and creates a live connection to a database. As the underlying data changes, site visitors see the changes without the need for the site designer to re-publish pages. When site visitors make changes to data, other site visitors can immediately make use of the changed data. Both the components and the built-in data publishing features are useful in different situations. In general, use the built-in features if you do not have IIS or if your data changes infrequently. Use the ASP components when you want to provide up to date information or allow site visitors to change data from a web site.

The components have a broad range of functionality for incorporating live data into your web sites. Using the components, you can display record sets in a list or grid format and show individual records as straight HTML text or in HTML edit boxes. You can also add new records, update existing records, and delete records. There are even components to allow site visitors to search for records in a database and browse the result set, navigating both forward and backward through a view of the data.

When the components are published with NetObjects Fusion, they generate HTML and server-side JScript code for you that will run on IIS. Because code is automatically generated, you never need to write any JScript or HTML code to use these components, and using them requires no knowledge of JScript. A good working knowledge of relational databases, such as Microsoft Access, and some knowledge of simple SQL SELECT queries will be helpful when using these components. To run the server-side JScript code created by the components, you will need Windows NT 4.0, Microsoft IIS 3.0 or higher with ASP, and a database supported by ASP.

Summary of Components

- **DBConnection:** Establishes a connection to a database
- **DBDynaField:** Displays data for a single field in a table or query as formatted HTML text, text within an HTML form text input element, or text within an HTML form textarea input element
- **DBList:** Displays data supplied by a DBQuery component in list format as a dynamically generated HTML table
- **DBNav:** A set of user-definable icons that allow web site users to browse records in a DBList or DBDynaField, moving forward and backward through the records at a specified record intervals
- **DBQuery:** Creates a database cursor or recordset based upon a user-defined SQL SELECT query and returns a data set for use by other database components
- **DBSearch:** A button or image that allows users to perform searches on a database
- **DBUpdate:** A button or image that allows users to add, modify or delete records

Caution: Double quotes are not allowed for any property values for any of the components. Always use single quotes when needed. The components will not warn you if you use double quotes, but you may receive errors when compiling or running your ASP applications if you use double quotes.

Requirements

While each component has its own specific requirements, all of the components share the following requirements:

- All pages containing any of the ASP components must have their extension changed to .asp in NetObjects Fusion. The file extension for each page can be set from Site view under the Page tab of the Properties dialog.

- If you're using NetObjects Fusion 3.0, you MUST configure NetObjects Fusion to publish by asset type. You can set this parameter in Publish view by changing the Directory Structure parameter under the Directory Structure tab of the Publish Settings dialog to **By Asset Type**. The components do not support the other directory structure settings.

DBConnection

Version 2.0

The DBConnection component creates a connection to a database. Once it establishes a connection, the other ASP database components use that connection to perform operations on the database. It is a non-visual component, which means that while it has a visible icon in NetObjects Fusion, it does not appear on the published HTML page. However, it does publish several hidden form fields that are used by the DBSearch and DBUpdate components.

Caution: Because DBConnection publishes hidden form fields, you must be sure to place DBConnection inside the same form area as the corresponding DBUpdate or DBSearch field if you're using NetObjects Fusion 3.0, so that the hidden fields will be passed when the user submits the form. This only applies to non-global connections.

The DBConnection component can be used in two ways. You can connect to several different databases on the same web site, provided each connection exists on a different page. You also have the option of using just one DBConnection component with the entire site by taking advantage of the ASP global.asa file.

Note: When using a global database connection, you must create your IIS alias such that it points to the folder that contains the global.asa file. The global.asa file is always published to the top level directory that you publish a site to. Specifying an IIS alias that points to some folder other than the one containing the global file will result in errors.

If you set the DBConnection property "Global Connection" (described below) to **true** you can establish a global connection for the entire site. In such a case, place only one DBConnection component on any one page in the site, and all pages will have access to the connection.

Note: Only one DBConnection component is allowed per page. If you choose to use a global connection, a maximum of one DBConnection component is allowed per site.

To use the component, first establish a valid ODBC data source and configure a data source name (DSN) for the database. If the data is password protected, you must also have a valid user name and password. See your database server administrator or software documentation for more information.

Properties

- **ODBC Type:** The specific type of ODBC database you will be connecting to (e.g. MS Access, MS

SQL Server, etc.)

- **DSN Name:** The name of your ODBC DSN.

- **User Name:** A valid user name for the database you will be connecting to. In most databases, the user name determines the level of database access granted.

- **Password:** A password that corresponds to the user name.

- **Global Connection:** Set this property to **true** if you intend to establish a single global database connection for the entire application through use of the ASP global file. With this option you only need one database connection component for the entire web site. The component will automatically generate the global file (`global.asa`) for you when you create the application. Set this property to **false** if you wish to establish a separate connection on each page that requires database access. This option allows you to connect to different database servers on different pages within your site, and does not require or produce a global file.

Requirements

- All pages containing any database components must have their extension changed to .asp in NetObjects Fusion.
- You must have an ODBC compliant database server such as Microsoft SQL Server or Microsoft Access to store your data.
- The user name and password you provide must have proper permissions and rights for the level of access you require. For instance, make sure you have write access if your users need to perform update operations on the data.
- When created, this component copies its JScript resource files, `MSDBConnection.inc`, `MSGlobal.inc` and `adojavas.inc`, into the `assets\lib` subdirectory of the root publishing directory.

Example

To add database connectivity to a page, you must either use a global DBConnection component that uses the ASP global file, or use one component per page. To prepare a page in the latter case, follow these steps:

1) In Page view, display the page where you want to place the DBConnection component.

2) Select the “ASP Database Connection” tool from the Microsoft ASP toolbar.

3) Place the component anywhere within the Layout. Do not place the component in a MasterBorder.

4) Set the connectivity parameters to match those of your database server. For example, to connect to a Microsoft Access database with an ODBC System DSN of **CustomerDB**, a user name of **admin** and password of **admin**, set the properties to the following values:

    a) ODBC Type: **MS Access**

    b) DSN Name: **CustomerDB** (name of ODBC DSN)

    c) User Name: **admin**

    d) Password: **admin**

5) Since this is a non-global connection make sure the Global Connection property is set to **false**.

6) In Site view, choose the Page tab in the Properties window, choose Custom Names, and enter **.asp** as the File Extension. IIS will not execute the generated server-side JScript code if you do not change the file extension of the page to .asp.

The DBConnection component cannot be tested meaningfully by itself. To test your database connection, you could add a DBQuery and DBList component to the page and see if your DBList prints out the intended data values.

DBDynaField

Version 2.0

The DBDynaField component displays data from one field in a record. It's generally used to display data as either formatted HTML text or within an HTML form element so that the data may be modified. Often DBDynaFields are used in conjunction with a DBNav component to allow browsing of individual records. DBDynaFields may also be combined with a DBList on the same page to show data from a master-detail view. In a master-detail scenario DBDynaFields would show data from a single master record, such as a customer from a Customers table, and a DBList would display detailed records associated with the master record, such as associated orders from an Orders table

DBDynaFields may be configured to be one of two basic types: editable and non-editable. When editable, a DBDynaField will appear on the HTML page as an HTML form text element. When not editable, its font properties may be set and it will be displayed on the page as formatted HTML text.

Properties

- **Use Query:** Set this property to **true** if you want the component to display data from a field in a database. Set it to **false** if you want to use it as a blank data entry field or want to use your own default value.

- **Query Component:** Enter the name of the DBQuery component on the page that the DBDynaField will use to retrieve its data. You only need to enter a name into this field if Use Query is set to **true**. This is a case-sensitive property.

- **Data Field:** Enter the exact name of the database field that you wish to associate with the DBDynaField. The field name may be an actual field name in a table or may be an SQL alias. However, you may not use the SQL *table.fieldname* syntax here, only the name of the field. If you need to distinguish between ambiguous field names, define an SQL alias in your query for the ambiguous field name and refer to the alias instead. Valid names must start with a letter, and can only contain letters, numbers and underscores (_).

- **Data Type:** Choose the data type of the field.

- **Default Value:** (optional) This property will only be used by the component if Use Query has been set to **false**. It is ignored otherwise. You can use the field to define a default value that will appear in the DBDynaField. For example, you might want to provide default values for certain fields when adding a record to a table or performing a search.

- **Is Editable:** When set to **true**, the DBDynaField will appear on the page as an HTML form text input field. When **false**, the DBDynaField will appear as HTML text and you can use the properties

defined below for formatting. Editable DBDynaFields cannot have their font properties modified.

Note: Netscape Navigator will not display your DBDynaField components if the Is Editable property has been set to **true** unless you have placed the DBDynaField in a region designated to act as a form region. For non-editable DBDynaFields, no such requirements exist.

- **Visible Length:** Set this property equal to the number of characters you wish to display. For editable DBDynaFields, this defines how wide the input box will be. For non-editable components, this simply allocates space on the HTML page for displaying the data. Be sure to set this value to be wide enough to display your data as you desire for both editable and non-editable types. If you are displaying data from a fixed-width column in a non-editable DBDynaField, you should set the Visible Length property to be equal to the length of the column to ensure that all of your data is displayed.

- **Max Length:** This property only has relevance to editable DBDynaFields. It defines the maximum number of characters that a user is allowed to enter into the input box. This value must always be greater than or equal to Visible Length. If you are displaying data from a fixed-width column in an editable DBDynaField, you should set the Max Length property to be equal to the length of the column to ensure that all of your data is displayed properly within the field.

- **Visible Height:** This property modifies the DBDynaField differently depending upon whether or not the object is editable or not. If the DBDynaField is non-editable, the property is used to allocate additional space on the page in case you are displaying a large amount of text. However, the component does not wrap text for you, so this property should only be used to reserve vertical space on the page. If the DBDynaField is editable, this property will create a text area field with the number of rows equal to the visible height, if the visible height is set to a number greater than 1.

Note: The following properties only apply to non-editable DBDynaFields.

- **Font:** Enter a font face such as *Times New Roman* into this field. All data displayed in this field will use this font if the viewer's browser supports it.

- **Font Size:** Choose a font size for the displayed data.

- **Bold:** Set to *true* to make the text appear in bold letters.

- **Italic:** Set to *true* to italicize the text.

- **Underline:** Set to *true* to underline the text.

- **Color:** Choose a color for the displayed data.

Requirements

- Data aware DBDynaField components require that a DBQuery component exist on the page with a valid SQL query. Data aware DBDynaFields obtain their data from the result set returned by the DBQuery component on the page.
- When viewing editable DBDynaFields with Netscape Navigator, you must have at least one DBUpdate component or HTML form button placed on the same page or your fields may not appear in the browser. For non-editable DBDynaFields, no such requirement exists.
- Because of the way the components manipulate SQL queries, users cannot use single quotes by themselves when using DBDynaFields to search on or update values that contain single quotes. Users must instead enter two single quotes in a row, e.g. '', in order to search on or update a value containing one single quote. The database server will strip off the extra single quote before using the value in an update or search.
- In NetObjects Fusion, you must designate some area within the layout to be a form area. This allows you to have more than one form area on the page, if you wish. The DBDynaField component must also be placed within a form layout area on the page, just like any other HTML form element.
- When published, this component copies its JavaScript resource file, `MSDBDynaField.js`, into the `assets\lib` subdirectory of the root publishing directory.

Example

DBDynaFields are commonly used as data aware, editable text input fields. When both editable and data aware, DBDynaFields can allow users to modify a field's value in the database if used in combination with a DBUpdate component.

To place a data aware, editable DBDynaField on a page and connect it to a data source, you'll need to have one DBConnection component on the same page (or one global DBConnection component somewhere in the site) and one DBQuery component. The DBQuery component will have to be configured with a query that will return the field you wish to display with your DBDynaField.

To configure your DBDynaField to work with the other components mentioned above, edit its property settings as follows:

1) In Page view, display the page where you want to place the DBDynaField component.
2) Select the "ASP Database Dynafield" tool from the Microsoft ASP toolbar.
3) Place the component anyplace within the Layout. Do not place it in a MasterBorder.
4) Set the parameters for the component. To configure the component to display the field **CustomerName** from a DBQuery component called **CustQuery**, set the parameters as follows:

a) Use Query: **true**. Set to true to fill in the DBDynaField with data from a field in the associated query.

b) Query Component: **CustQuery**. Set this to the name of the DBQuery component on the same page.

c) Data Field: **CustomerName**. The field **CustomerName** must exist in the result set defined by the SQL query that **CustQuery** generates.

d) Data Type: **string**.

e) Is Editable: **true**.

f) Visible Length: **30**

g) Max Length: **40**. This value should generally be set equal to the length of the field in the database, in this example **CustomerName**.

5) In Site view, choose the Page tab in the Properties window, choose Custom Names, and enter **.asp** as the File Extension. IIS will not execute the generated server-side JScript code if you do not change the file extension of the page to .asp.

Once the component is configured and accompanying DBQuery and DBConnection components have also been configured, you may publish the site and test the DBDynaField.

DBList

Version 2.0

The DBList component displays formatted data from a database in an HTML table. It obtains its data from the DBQuery component on the same page. The component allows you to display lists of records, and you can choose which fields from the query's result set you would like to display. You can even choose to hyperlink the data items in one column of data, allowing users to view a different page that may contain more specific data related to the selected record. Typically, this component is used to give a broad view of several records, allow users to select an individual record, and drill down to a more detailed view.

The DBList component provides extensive control over the way data is presented to site visitors. You can define the font properties of both the column labels and the column data, including font face, color, size, etc. You can also control the HTML table formatting properties such as border size, cell padding, and so forth.

To use this component, you should be a site designer with information design expertise. Familiarity with font properties and HTML table properties is also required, as well as an understanding of the concept of key fields in database tables.

In the world of relational databases, there are two common types of key fields, primary keys and foreign keys. A primary key is comprised of one or more fields from a table that uniquely identify a record. For example, in a table of employees, Social Security number might be used as a primary key because no two employees can have the same Social Security number. A foreign key is a field in one table that refers to a primary key in another table. For example, the table Employees may contain a field called DeptNo that refers to the primary key of the Department table. This relationship between foreign key and primary key can tell you in which department an employee belongs.

Properties

- **Name:** A name for this instance of the List component. The name must be unique among all other ASP components on the page. Valid names must start with a letter, and can only contain letters, numbers and underscores (_).

- **Query Component:** The name of the DBQuery component on the page that will supply data to this component. This is a case-sensitive name, so *MSDBQuery1* is not the same as *msdbquery1*.

- **Nav Component:** (optional) The name of the DBNav component on the page that will navigate the records displayed by the list. Use this property only you will be using the DBList component with a DBNav component. This is a case-sensitive name, so *MSDBNav1* is not the same as *msdbnav1*.

- **Label Font:** A font face for the column labels that will appear in a single row at the top of the table. Possible values include *Helvetica*, *Times New Roman*, etc.
- **Label Size:** The font size of the column labels.
- **Label Color:** The color of the column labels.
- **Label Bold:** Set this property to *true* to make the column labels bold.
- **Label Italic:** Set this property to *true* to italicize the column labels.
- **Label Underline:** Set this property to *true* to underline the column labels.
- **Data Font:** A font face for the data values that will appear within the table area. Possible values include *Helvetica*, *Times New Roman*, etc.
- **Data Size:** The font size of the data values.
- **Data Color:** The color of the data values.
- **Data Bold:** Set this property to *true* to make the data values bold.
- **Data Italic:** Set this property to *true* to italicize the data values.
- **Data Underline:** Set this property to *true* to underline the data values.
- **Cell Padding:** Defines the cell padding in the HTML table.
- **Cell Spacing:** Defines the cell spacing in the HTML table.
- **Border Size:** Defines the size of the border that will appear around the HTML table, columns, and rows. Setting the value to 0 will hide the borders.
- **Hyperlink Page:** (optional) Designates a column as a hyperlink to another page within the site. Typically, the hyperlinked page will contain other components, such as DBDynaFields, to display more detailed information about a selected record.
- **Hyperlink Field:** (optional) If a hyperlink page is selected, the Hyperlink Field defines which column will be hyperlinked.

- **Key Field Count:** (optional) A positive value for this field indicates the number of fields that comprise the primary key for the data being displayed. Setting this property to a number greater than zero is only useful if you have selected a Hyperlink Page and field. It is used in conjunction with the Key Field Name, Key Field Label and Key Field Type properties. Combined their purpose is to uniquely identify a record that a user clicks on so that a detailed view of that record can be displayed on the Hyperlink Page.

  For example, if the primary (unique) key of a table called Employee is a string field called "SSN," set the Key Field Count to 1. When you do, the Key Field Name, Key Field Label and Key Field Type properties are created for this primary key field. In this example, the correct entries for each field are **SSN** for the Key Field Name, **SSN** for the Key Field Label and choose **string** for the Key Field Type.

- **Key Field Name:** The name of the key field as it exists in your query. With most databases this name is not case-sensitive, but it must be spelled exactly the same as the field name or SQL alias name. You may not use the SQL *table.fieldname* syntax here, only the name of the field. To distinguish between ambiguous field names, define an SQL alias in the query for the ambiguous field name and refer to the alias instead.

- **Key Field Label:** The name of the key field that will be passed to the Hyperlink page when a site visitor selects a record. Generally, this value is identical to the Key Field Name property. You should only enter a different value if the DBQuery component on the detail page contains a query that uses a different corresponding key field name.

  For example, suppose a table called Customers has a primary key field name of **CustomerID**, and a table called Orders has a foreign key field called **CustID**, which relates to the primary key of the Customers table (see ***Figure 1*** below). Suppose also that the DBList component displays a list of customers, and when a site visitor selects a customer name from the list, that customer's orders must be displayed. The two tables are already related in the database by the primary key and foreign key relationship. In this example, set Key Field Name to the name of the primary key of the Customers table, **CustomerID**, and Key Field Label to the foreign key in the Orders table, **CustID**.

  You may not use the SQL *table.fieldname* syntax here, only the name of the field. If you need to distinguish between ambiguous field names, define an SQL alias in the query for the ambiguous field name and refer to the alias instead.

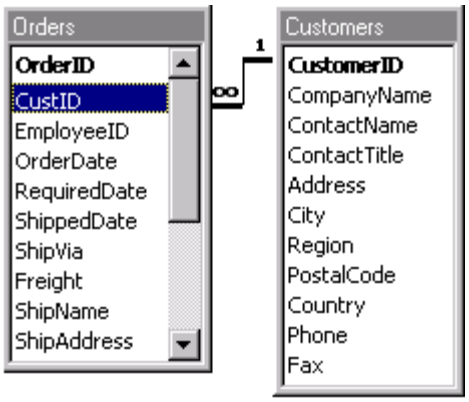

Figure 1. Sample database relationship between foreign key, CustID, and primary key, CustomerID

- **Key Field Type:** The data type of the primary key field.

- **Field Count:** The number of fields from the database to display in the list.

- **Field Name:** For each pair of Field Name and Field Label properties, specify the name of the field as it exists in the SQL query or database table. This is the name the DBList component will use to find the field in the result set, so make sure to enter the field name exactly as it exists in the query. You may not use the SQL *table.fieldname* syntax here, only the name of the field. If you need to distinguish between ambiguous field names, define an SQL alias in the query for the ambiguous field name and refer to the alias instead.

- **Field Label:** For each pair of Field Name and Field Label properties, specify the name of the field as you want it to appear in the column header at the top of the list. The name you enter here can be different than the Field Name. Typically, this property is used to give more readable names for fields in the database that may not have very presentable names. For example, if the Field Name is **SSN**, the Field Label can display **Social Security Number** so that it is more presentable to site visitors.

Requirements

- The DBList component requires that a DBQuery component exist on the page with a valid SQL query.
- When published, this component copies its JScript resource file, `MSDBList.inc`, into the `assets\lib` subdirectory of the root publishing directory.

Example

The DBList component has a variety of features for displaying lists of records in many formats. In addition to its display capabilities, DBList also plays an integral part in constructing interesting and useful views of data for site visitors, giving them a great deal of control over what data they will see. This example will illustrate how to construct a common view of data.

The example data view involves two pages - the list page, which displays lists of records via a DBList component, and the detail page, which uses DBDynaFields to display detailed information about a single record. The intention is that a site visitor views the list page, selects a hyperlinked record in the list, and detailed information about the chosen record is displayed via the detail page.

Consider the example of displaying a list of customers on the list page and a detailed view of a chosen customer on the detail page. Figure 2 shows the field list from the Customers table including its primary key field, CustomerID, which uniquely identifies each record in the table. Knowing the primary key or keys for each table is important when setting up the relationship between list and detail pages.

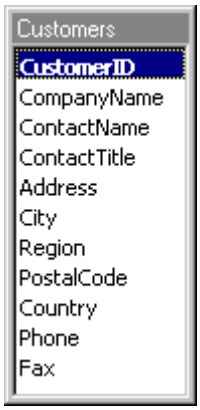

Figure 2. A sample database table.

This page requires a DBConnection component (or a single global DBConnection component placed anywhere in the same site), a DBQuery component configured to select records from the Customers table, and a DBList component. A DBQuery component configured with an SQL query such as **select * from Customers where city='Seattle' order by ContactName** will retrieve all customers who live in Seattle and will sort the results by the ContactName field. The DBList component displays the results in the same sequence defined by its associated DBQuery component, so the resulting list will also be sorted by ContactName when displayed to site visitors.

To complete the list page, configure the DBList component as follows:

1) In Page view, display the page where the DBList component is to be placed.

2) Select the "ASP Database List" tool from the Microsoft ASP toolbar.

3) Place the component anywhere within the Layout. Do not place it in a MasterBorder.

4) Stretch the DBList component borders to the approximate size required. While the width of the object will roughly equal the width you define by stretching the component, the length is defined by how many rows of data are displayed. However, it is a good idea to stretch out an approximate length to allocate space on your layout for the component. The component divides the total width you define by the number of display columns and renders all columns in the table with equal width.

5) Set the parameters for the component.

    a) Name: **CustomerList**. Give the component a name that is unique among all other components on the same page.

    b) Query Component: **CustQuery**. Enter the name of the DBQuery component you wish to use as your data provider.

    c) Label Font properties: These properties will control the appearance of the column names in the resulting HTML table. Typically, you will want to highlight the column headers by making them bold and possibly changing their color.

    d) Data Font properties: Set the font properties to control the appearance of all data displayed in the table. These properties will override any site style settings.

    e) Border Size: **0**. For this example, eliminate the table borders around the list by setting the border size to zero.

    f) Hyperlink Page: **Customer Detail**. Use the URL chooser to select the detail page where users will be sent when they click on a record to view detailed information. This example assumes the page is named **Customer Detail**.

    g) Field Count: **3**. The only three fields from the query that will be used in this list are ContactName, CompanyName, and Phone. When you increment the Field Count to 3, some extra properties appear that allow you to specify which fields you want to display.

    h) Field Name 1: **ContactName**. This value must be the exact name of the field as it appears in the query.

    i) Field Label 1: **Customer Name**. This string will appear at the top of the table as the column header.

    j) Field Name 2: **CompanyName**. This value must be the exact name of the field as

it appears in the query.

k) Field Label 2: **Company**. This string will appear at the top of the table as the column header.

l) Field Name 3: **Phone**. This value must be the exact name of the field as it appears in the query.

m) Field Label 3: **Phone #**. This string will appear at the top of the table as the column header.

n) Hyperlink Field: **ContactName**. After you have entered fields that will be displayed in the list, you will see that the drop-down list for this property contains each of those field names. Select ContactName to make that field a hyperlink in the list. When site visitors click on a ContactName in the list, they are sent to the Customer Detail page along with the primary key information for that record so the Customer Detail page can identify which record they wish to view.

o) Key Field Count: **1**. The Customers table has one primary key field, CustomerID.

p) Key Field Name 1: **CustomerID**. The value of this field is sent to the Customer Detail page to uniquely identify the record when a site visitor clicks on a hyperlinked field. With this information, the DBQuery component on the Customer Detail page finds the desired customer record.

q) Key Field Label 1: **CustomerID**. This field is most useful when the DBQuery component on the detail page retrieves records from a different table than the table on the list page, and the foreign key is called by a different name than is used in the list table. Generally, the same value as the Key Field Name property is used here.

r) Key Field Type 1: **number**. The CustomerID field is defined as an integer. Any type of number -- integer, float, etc. -- uses the number data type.

6) In Site view, choose the Page tab in the Properties window, choose Custom Names, and enter **.asp** as the File Extension. IIS will not execute the generated server-side JScript code if you do not change the file extension of the page to .asp.

With a fully configured DBList component on the list page, the final step is to construct the Customer Detail page. The Customer Detail page must contain a DBConnection component (or a global connection somewhere in the site), a DBQuery component with "Use Initial Page" property is set to **true**, DBDynaFields to display field data, and possibly a DBNav component to allow browsing beyond the individual record chosen. Refer to the DBDynaField component description for more information on constructing this page.

DBNav

Version 2.0

The DBNav component navigates through a record set one record at a time. Typically, it is used in combination with DBDynaField components or a DBList component to allow users to browse individual records or groups of records in a result set. Often, you will want to set up master-detail pages to show both a navigable list of records and their navigable detail or single record view. DBNav requires a DBQuery component on the same page.

DBNav allows both forward and backward navigation through the result set. You can set the number of records to navigate each time a site visitor clicks the next or previous buttons. When used with DBList components on list pages, you can also restrict the number of records that display at one time for easier viewing, and allow site visitors to navigate through groups of records. For detail pages, visitors will usually only want to navigate one record at a time.

The DBNav component navigates through records contained in one of two result sets. If its associated DBQuery component has its "Use Previous Query" property set to **false**, then DBNav navigates the result set defined by the DBQuery component's select statement. If the "Use Previous Query" property is set to **true** then DBNav does one of two things:

- If the user arrives at the DBNav page by clicking on a hyperlink from a DBList component on a previous page, DBNav navigates the result set displayed in the DBList component from that previous page. In this case, DBNav ignores its own DBQuery component and instead navigates the result set defined by the DBQuery on the previous page. This gives the site visitor the impression of navigating records in the list they just came from.
- Otherwise, DBNav uses the result set defined by the DBQuery component on the same page as the DBNav component.

To use the component, you should be a site designer with information design expertise. You will also need to be familiar with the concept of primary keys and be reasonably familiar with relational databases.

Note: Only one DBNav component may be used per page.

Properties

- **Name:** A name for this instance of the DBNav component. The name must be unique among all other ASP components on the page. Valid names must start with a letter, and can only contain letters, numbers and underscores (_).
- **Query Component:** The case-sensitive name of the DBQuery component on the page.

- **Navigation Type:** Specifies whether the component navigates a list of multiple records or a detailed view of a single record.

- **Records to Move:** The number of records the cursor advances with each click of the next or previous buttons. Also defines how many records will be displayed per page when used in combination with a DBList component

- **Previous Image:** An image that users will click on to move to the previous record in the record set. This image defaults to an image of a black circle with an arrow pointing to the left. The image used can be .GIF or .JPG.

- **Next Image:** An image that users will click on to move to the Next record in the record set. This image defaults to an image of a black circle with an arrow pointing to the right. The image used can be .GIF or .JPG.

- **Up Image:** An image that users will click on to return to the page that sent them to the page that contains the DBNav component. This image defaults to an image of a black circle with an arrow pointing up. The image used can be .GIF or .JPG.

- **This Page:** The name of the current page, so the DBNav component knows which page it resides on.

- **Up Page:** The page in the site that sends the site visitor to this page. Usually, this page contains a DBList component with a hyperlinked field pointing to this page.

- **Key Field Count:** Indicate the number of fields that comprise the primary key for the data being displayed. The Key Field Count is used in conjunction with the Key Field Name, Key Field Label and Key Field Type properties. Combined, their purpose is to uniquely identify a record when navigating the record set.

  For example, if the primary (unique) key of a table called Employee is a string field called "SSN," set the Key Field Count to 1. When you do, the Key Field Name, Key Field Label and Key Field Type properties are created for this primary key field. In this example, the correct entries for each field are **SSN** for the Key Field Name, **SSN** for the Key Field Label and choose **string** for the Key Field Type.

- **Key Field Name:** The name of the key field as it exists in your query. With most databases this name is not case-sensitive, but it must be spelled exactly the same as the field name or SQL alias name. You may not use the SQL *table.fieldname* syntax here, only the name of the field. To distinguish between ambiguous field names, define an SQL alias in the query for the ambiguous field name and refer to the alias instead.

- **Key Field Type:** The data type of the primary key field.

Requirements

- The DBNav component requires that a DBQuery component exists on the page with a valid SQL query that returns a data set to be navigated.
- If a DBList is to be navigated, the DBList must have its Nav Component property set to the name of the DBNav component.
- When published, this component copies its JavaScript resource file, `MSDBNav.inc`, into the `assets\lib` subdirectory of the root publishing directory.

Example

In a typical web database application, site visitors will want to browse individual records in the database. On a page that displays a detailed view of any single record with DBDynaFields, the DBNav component provides record this record-by-record browsing ability.

DBNav requires a DBQuery component on the same page, and it's not very useful without a DBList or some DBDynaFields to display the data as the user navigates. To construct a page that allows users to browse through records in a Customer table (see Figure 1), the DBQuery component must be configured in one of two ways. It can be configured to use the query from some previous list page, or it can always use its own query parameters to define a SELECT statement such as **select * from Customers where city='Seattle'**. The DBDynaFields are used to display data from the DBQuery component's result set.

Once the other elements on the page are configured, configure the DBNav component as follows:

1) In Page view, display the page where the DBNav component will reside.

2) Select the "ASP Database Record Navigator" from the Microsoft ASP toolbar.

3) Place the component anyplace within the Layout. Do not place it in a MasterBorder.

4) Set the parameters for the component.

    a) Name: **CustomerNav**. Give the component a name that is unique among all other components on the same page.

    a) Query Component: **CustQuery**. Use the exact name of the DBQuery component on the page.

    b) Navigation Type: **Detail**. Select the type of view the records are in. These records are a single record detail view.

    c) Records to Move: **1**.

d) Previous Image: <default>. If you select a different image, the component will automatically resize itself on the page to fit the size of the image. You will not see the selected image until after you publish and run the web application.

e) Next Image: <default>.

f) Up Image: <default>

g) This page: In the URL chooser, select the **Current Page** option to tell the DBNav component what page it resides upon.

h) Up Page: Typically, users arrive at a detail view page containing a DBNav component by clicking on a hyperlink in a DBList component on another page. In such a scenario, set this parameter to point to the page containing the DBList.

i) Key Field Count: **1**. The Customers table has one primary key field, CustomerID.

j) Key Field Name 1: **CustomerID**. This name must exactly match the name of the key field as defined in the query.

k) Key Field Type 1: **number**.

5) In Site view, choose the Page tab in the Properties window, choose Custom Names, and enter **.asp** as the File Extension. IIS will not execute the generated server-side JScript code if you do not change the file extension of the page to .asp.

After configuring the DBNav, DBQuery, and DBDynaField components, the detail view page is ready to be published along with the rest of the pages in your ASP application.

DBQuery

Version 2.0

The DBQuery component defines a SQL query that all other data-aware objects on a page must use to obtain data. Any valid SQL SELECT statement can be used to query the database, including multi-table joins. The resulting set of database records, the result set, is available for display or manipulation by the other database components. As with DBConnection, this component produces no visible HTML representation.

Note: Only one DBQuery component can be used per page.

To use this component, you should be a site designer with some experience writing SQL SELECT queries and some familiarity with databases. If you are not familiar with these concepts, ask your database administrator for help when writing the queries. The DBQuery component combines the values entered for the Select, From, Where and Order By properties into a single SQL SELECT statement such as **select * from Customers where city='Seattle'**.

Properties

- **Name:** A case-sensitive name for the DBQuery component. Valid names must start with a letter and can only contain letters, numbers and underscores (_).

- **Use Previous Query:** This option determines how the query gets its result set. Set the parameter to **false** if you want this component to always use its default query. This is the most common setting. Set this option to **true** if you want the component to use record set from the query on a list page the user just came from, rather than its default query. A default query must always be defined, even if this parameter is set to true. The default query is used whenever site visitors reach the page without coming from a prior list page (for example, if they bookmarked the page in their browser and returned directly to it).

  Note: This property should always be set to **false** unless used in combination with a DBNav component. When used with a DBNav component, it may be set to either **true** or **false**. See the description of the DBNav component for more details on the how to use this property.

- **Select:** The SELECT portion of the query, enumerating the fields that the result set will contain. Enter * to select all fields, or enter the desired fields by name, separated by commas. Refer to a SQL reference guide for your database for proper syntax requirements and options.

- **From:** The FROM portion of the query. Generally, a single table name or a list of table names separated by commas is entered here.

- **Where:** (optional) A valid WHERE clause. The statements specified here are criteria that determine which records are chosen from the tables specified in the From property. Double quotes are not allowed in this field.

- **Order By:** (optional) A comma delimited list of field names, specifying the sequence in which the result set will be ordered.

Requirements

- All pages containing database components must have their extension changed to .asp in NetObjects Fusion.
- The DBQuery component requires that either one DBConnection component on the same page, or that a single global DBConnection be present on any page within the site.
- When published, this component copies its JScript resource file, MSDBQuery.inc, into the `assets\lib` subdirectory of the root publishing directory.

Example

The DBQuery component must exist on any page that will display data from the database, perform modify or delete operations or navigate a result set. It is not useful by itself since it does not actually execute any queries until asked to do so by one of the visual components. For this example, suppose that you want to display a list of customers from a table called **Customers** (see Figure 1). To do so, you need a DBConnection component on the same page (or a global component somewhere in the same site) to establish a connection to the database. You also need a DBList component on the page to display the results of the SQL query defined in the DBQuery component.

To configure the DBQuery component, follow these steps:

1) In Page view, display the page where you want to place the DBQuery component.

2) Select the "ASP Database Query" from the Microsoft ASP toolbar.

3) Place the component anyplace within the Layout. Do not place it in a MasterBorder.

4) Set the parameters for the component. For this example, you require a list of customer data. This implies a result set of data, which must be defined by some SQL SELECT statement. To display information about all customers who live in Seattle, the DBQuery component might be configured as follows:

    a) Name: **CustQuery**. This name must be unique among all components on the page.

b) Use Previous Query: **false**.

c) Select: *****. In this example, all records are selected. However, for performance reasons or to perform aggregation (e.g. count, sum, etc.), specifying individual fields or aggregate functions can be advantageous.

d) From: **Customers**. Generally this is a comma-delimited list of tables, but it may contain any valid SQL FROM clause.

e) Where: **city='Seattle'**. Limit the results to only those customer records where the value in the **city** field is equal to the string **'Seattle'**. This value is optional. Note that single quotes are used; double quotes are not allowed.

Note: Some databases might conduct case-sensitive queries when comparing strings as in the above Where clause and others might not. Be sure to read your database documentation to find out if it conducts case-sensitive queries. You may be able to override the default behavior with special commands in your SQL statement. Refer to your vendor's documentation for more details.

---

f) Order By: **LastName, FirstName**. Sorts the results by the values in the LastName field. If there are any duplicate values in LastName, those records will be additionally sorted by the FirstName field.

5) In Site view, choose the Page tab in the Properties window, choose Custom Names, and enter **.asp** as the File Extension. IIS will not execute the generated server-side JScript code if you do not change the file extension of the page to .asp.

DBSearch

Version 2.0

The DBSearch component executes a search using user-specified criteria and shows the results on a specified result page. It obtains the name of the fields to search and their respective values from the editable DBDynaFields on the page.

To use this component, you should be a site designer. Typically, designers will create search page that includes one search button and one or more DBDynaFields. The field name property of each DBDynaField should be set to one of the fields in the query on the result page. Matching should be set to *exact match* if any of the supporting DBDynaFields data types are set to number, date or boolean. When using multiple DBDynaFields the search criteria are compounded (or joined with AND operators). Because of these things, placing multiple DBDynaFields on a search page yields best results if all are string fields with matching types of either *starts with* or *contains.* An alternative to having a separate search and results page is to create one page that contains a DBSearch component for conducting a search and also contains DBQuery and DBList components for displaying results.

It is important to understand that the DBSearch component does not actually perform a search itself. It merely sends the search criteria and data entered by a user to the result page, and the DBQuery component on the result page uses the search criteria to generate a new result set.

Properties

- **Name:** Enter a name for this instance of the DBSearch component. This is a case-sensitive name must also be unique among all other ASP components on the. Valid names must start with a letter, and can only contain letters, numbers and underscores (_).

- **Button Type:** Specifies the type of button you wish to display - *image* or *text*.

- **Button Image:** (optional) Specifies the image to be used as the button icon. This property requires the Button Type to be *image*. Setting the Button Image will automatically cause the Button Type property to change to *image*. The image used can be .GIF or .JPG

- **Button Text:** (optional) The text that appears on the face of the update button. This property requires the Button Type to be set to *text*. Entering a Button Text value will cause the Button Type property to automatically change to *text*.

- **Matching:** The type of matching to perform. Set to *exact match* if any of the supporting DBDynaFields data types are number, date or boolean. Placing multiple DBDynaFields on a search page yields best results if all are string fields, and Matching is set to either *starts with* or *contains.*

- **Result Page:** Specifies the page that will display the results of the search. The result page must have a DBQuery component to define the data set that will be searched, and generally contains either a DBList component for displaying the results of the search or DBDynaFields.

Requirements

- The DBSearch component requires that one or more DBDynaFields exist on the same page.
- The result page must point to a page with a DBList and a DBQuery. The result page can be the same page as the search page.
- You must perform one extra step to use this component. In NetObjects Fusion, you must designate some area within the layout to be a form area. This allows you to have more than one form area on the page, if you wish. The DBSearch component must also be placed within a form layout area on the page, just like any other HTML form element. The only difference is that you set the form's action attribute using the component's Result Page property rather than setting the form's action attribute directly. You must also be sure that any corresponding DBDynaField elements are located within the same form area as DBSearch.

Example

As described above, the DBSearch component is used in combination with one or more DBDynaField components to initiate a search. The DBSearch component does not actually perform the search. Instead, it simply submits data entered by a user in DBDynaFields to a result page, which you create. The submitted data acts like search criteria and narrows down the result set defined by a query component on a result page.

For example, to conduct a search a Customer table by customer name requires two pages. The first page, the search page, contains one DBSearch component and one editable DBDynaField corresponding to the customer name field. When site visitors click on the DBSearch button or image, its data and the customer name is sent to a second page, the result page. DBQuery components are designed to "listen" for this search information and use it to search for the values in the result set defined by its default query.

Therefore, a result page must contain a DBConnection component (or a global DBConnection somewhere in the site), a DBQuery component to process the search information and produce a new result set, and a DBList component to display the results of the search. For this example, the result page needs to contain a DBQuery component to conduct a SQL query on a table called Customers, for example **select * from customers**.

To configure the DBSearch component for the search page:

1) In Page view, display the page where you want to place the DBSearch component.

2) Select the "ASP Database Search" tool from the Microsoft ASP toolbar.

3) Place the component anywhere within the Layout. Do not place it in a MasterBorder.

4) Set the parameters for the component. For this example, the set the parameters as follows:

    a) Name: **CustSearch**. This name must be unique among all other components on the same page.

    b) Button Type: **Text**.

    c) Button Text: **Find Customers**

    d) Matching: **contains**. This search will look for any values that simply contain the string entered by the site visitor anywhere within the customer name. The customer name field is of string type (as defined by the DBDynaField on the same page as this component and by the database). This makes use of a "contains" search possible.

    e) Result Page: **List Customers**. This is a pointer to a page in the same site called List Customers that contains a DBConnection, DBQuery and DBList component as described above. This page displays the search results.

5) In Site view, choose the Page tab in the Properties window, choose Custom Names, and enter **.asp** as the File Extension. IIS will not execute the generated server-side JScript code if you do not change the file extension of the page to .asp.

DBUpdate

Version 2.0

The DBUpdate component updates one record in a table. Working together with DBDynaField components on the same page, the DBUpdate component provides facilities to add, modify and/or delete records. The DBUpdate component obtains the name of the fields to be updated and their respective values from the editable DBDynaFields on the page.

To use this component, you should be a site designer. In a typical scenario, the designer sets up a detail page with DBConnection, DBQuery, DBNav and DBDynaField components. This page allows the user to browse the data specified by a single table query. Including two DBUpdate components, one with the Button Action property set to **modify** and the other to **delete,** allows site visitors to modify and/or delete the current record. A DBUpdate component that performs an **add** operation is typically set up on a page by itself, rather than in combination with other **modify** or **delete** versions of the component. The add page should not have a DBQuery component, and its DBDynaFields should all have their "Use Query" property is set to **false**. This way, added records will start with blank field values instead of values found in other records of the table.

Behind the scenes, when the site visitor clicks the DBUpdate button the component silently sends the user to the MSDBUpdateHandler.asp page, which contains special server-side JScript code. This page constructs the INSERT, UPDATE, or DELETE SQL statement, passes it on to the selected database for execution, and redirects the site visitor to the success or error page depending upon the success or failure of the operation. Due to software limitations, the update component can provide very little information to the site developer regarding the exact errors originating from the database. Blank key fields or required fields, mismatched data types, referential integrity violations, and other database-enforced errors will result in visits to the error page. Therefore, it is important to include as much helpful information as possible on the error page regarding what things might have caused an error. If any untrappable errors occur once the site visitor has been sent to the handler page, they can be presented with an intermediate error page containing detailed error information, which can be reported to a server administrator.

Note: When using the Update component, you must place the DBConnection component in the same form as the DBUpdate component in order for the components to connect to the database properly.

Properties

- **Name:** A name for this instance of the DBUpdate component. This is a case-sensitive name and also must be unique among all other ASP components on the page. Valid names must start with a letter, and can only contain letters, numbers and underscores (_).

- **Query Component:** The name of the DBQuery component that will supply data to the DBUpdate component. This is a case-sensitive name, and the DBQuery component must be on the same page.

- **Button Action:** The type of update to be performed on the table. Choose to add a new record, modify or delete an existing record.

- **Button Type:** The type of button you wish to display - image or text.

- **Button Image:** An image to be displayed for the button. This property requires the Button Type to be set to *image*. Choosing a Button Image for a button already set to type *text* will change the Button Type to type *image* automatically.

- **Button Text:** The text to be displayed on the face of the update button. This property requires the Button Type to be set to *text*. Changing the Button Text for a button already set to type *image* will change the Button Type property to type *image* automatically.

- **Table Name:** The name of the table to update. All editable DBDynaFields on the page performing the update must have valid field names in the specified table.

- **Key Field Count:** Indicate the number of fields that comprise the primary key for the data being displayed. The Key Field Count is used in conjunction with the Key Field Name and Key Field Type properties to uniquely identify the record updated in the table.

  For example, if the primary (unique) key of a table called Employee is a string field called "SSN," set the Key Field Count to 1. When you do, the Key Field Name and Key Field Type properties are created for this primary key field. In this example, the correct entries for each field are **SSN** for the Key Field Name and choose **string** for the Key Field Type.

- **Key Field Name:** The name of the key field as it exists in your query. With most databases this name is not case-sensitive, but it must be spelled exactly the same as the field name or SQL alias name. You may not use the SQL *table.fieldname* syntax here, only the name of the field. To distinguish between ambiguous field names, define an SQL alias in the query for the ambiguous field name and refer to the alias instead.

- **Key Field Type:** The data type of the primary key field.

- **Success Page:** The page to which the site visitor is automatically directed upon successful update of the table. This page can contain a success message and links to other pages.

- **Error Page:** The page to which the user is automatically directed upon failure of the update or insert operation. This page might contain messages suggesting possible problems with their data or instructions on how to receive technical support, as well as links to other pages.

Requirements

- The DBUpdate component requires that either one DBConnection component exists on the same page, or that a single global DBConnection be present on any page within the site. It also requires editable DBDynaFields for each field value you wish to add or modify. The DBDynaField components can get their default values from either an active query on the page or their own Default Value property. The update component will take the current value for each editable DBDynaField and update the table value.
- For modify and delete operations, a DBQuery component is also required so that the DBUpdate component may obtain primary key values and uniquely identify the record to be updated.
- When published, this component copies its JScript resource file, MSDBUpdate.inc and the UpdateHandler.html file, into the `assets\lib` subdirectory of the root publishing directory.
- In NetObjects Fusion 3.0, you must designate some area within the layout to be a form area. This allows you to have more than one form area on the page, if you wish. The DBUpdate component must also be placed within a form layout area on the page, just like any other HTML form element. DBUpdate sets the form's action attribute automatically. You must also be sure that any corresponding DBDynaField elements are located within the same form area as DBUpdate.

Example

The DBUpdate component can be used to add, modify or delete a single record. In all three cases, it is used in a similar fashion. A typical location for the DBUpdate component is at the bottom of a data entry page that contains DBDynaFields. After changing or entering in the DBDynaFields, the site visitor clicks on the DBUpdate button or icon to perform appropriate operation on the database server.

This example illustrates how to use the DBUpdate component to modify an existing record in the database. In this scenario, the DBUpdate component is added to a data entry page comprised of a DBConnection component (or a global DBConnection component elsewhere in the site), several DBDynaFields to display data from a record in the database, and a DBQuery component to provide the DBDynaFields with their data.

Suppose you want to update information for existing customers, such as their address and phone number. First, design your page as you would any page that displays detailed information about a single record. Refer to the DBDynaField description and example for more information on configuring those components properly.

With the DBQuery component set up to retrieve data from the Customer table and the DBDynaFields to display the data, configure the DBUpdate component to modify the data. Set the property values as follows:

1) In Page view, display the page where you want to place the DBUpdate component.

2) Select the "ASP Database Table Update" tool from the Microsoft ASP toolbar.

3) Place the component anyplace within the Layout. Do not place it in a MasterBorder.

4) Set the parameters for the component. For this example, set the parameters as follows:

   a) Name: **CustModify**. Give the component a name that is unique among all other components on the page.

   b) Query Component: **CustQuery**. Enter the name of the DBQuery component on the page. If this DBUpdate component was intended to perform an add operation, the Query Component property would be left blank.

   c) Button Action: **modify**.

   d) Button Type: **image**. A DBUpdate component can use an HTML form button or an image of your choice. However, due to limitations in the NetObjects Fusion API, you will not see the selected image until you publish and run your application.

   e) Button Text: Do not modify this value, since it only applies to text button types.

   f) Table Name: **Customers**.

   g) Key Field Count: **1**. The Customers table has one primary key field, an integer field called CustomerID.

   h) Key Field Name 1: **CustomerID**. Enter the exact name of the primary key field as it exists in the query. The DBUpdate component needs to know the name of the primary key field so it can uniquely identify which record you wish to update.

   i) Key Field Type 1: **number**.

   j) Success Page: Select a page in the site that users will see if the modification was successful.

   k) Error Page: Select a page in the site that users will see if the modification was unsuccessful.

5) In Site view, choose the Page tab in the Properties window, choose Custom Names, and enter **.asp** as the File Extension. IIS will not execute the generated server-side JScript code if you do not change the file extension of the page to .asp.

After publishing the site, use this DBUpdate component to modify a record in the Customers table.

Adding a Record

When using the DBUpdate component to add a record, the above scenario changes only slightly. When adding a record, site visitors generally do not require data from an existing record in the database. Therefore, there is no need for a DBQuery component, and the DBDynaField components do not have to be data aware. However, you must always use a DBConnection component with a DBUpdate component, or there must be a global DBConnection component somewhere in the site.

Supported Data Types

Several properties of the components described require that you choose a data type for a field in the database. There are four basic data types you may choose from: **string**, **number**, **date** and **boolean**. Those data types correspond to several possible data types available from your database server. Below are descriptions of how each component data type will handle data types from your database server. Using data types other than those described below may result in unexpected behavior.

string: Any string or character data type less than 255 characters in length. Large text data types, such as the Microsoft Access Memo data type, are not supported in Version 2.0.

number: Any integer, float or double data type may be used.

date: Any date data type will work. Time data types are not supported.

Date-time combination data types are only supported for database servers that allow only the date portion of the date-time to be inserted, updated or compared. The components always ignore the time portion of date-time data types. When inserting or comparing date-time values, the time portion will generally default to midnight (00:00:00), but may vary depending upon how your server is configured. These components are known to support date and date-time data types for Microsoft Access and Microsoft SQL Server.

Your web site users must enter date values in a specific format. Regardless of database type, all date values entered by users in DBDynaField components must be entered in an **mm-dd-yyyy, mm-dd-yy, mm/dd/yyyy** or **mm/dd/yy** format. Unexpected results may occur if users enter dates in any other format or (because of JScript and ASP limitations) enter dates before 1970 and after 2037.

If you allow users to enter date values in DBDynaFields for searches or updates, you should tell them about the required format on your web page. Values passed in from DBDynaFields are first interpreted by the components, validated and then translated into your database server's native format. This automatic translation makes data entry more convenient for your users since they do not need to know all of the different date formats for the different database servers you may be using.

By contrast, when configuring a DBQuery component, you must use your database server's native date format instead. For example, if you need to use a date value in the WHERE clause of your SQL query, you must use your database server's native date format because the DBQuery component passes the date value directly to your database server, without interpretation. Consult your database server documentation for information on proper date formats.

boolean: Boolean, logical, yes/no or single-bit data types will work with this data type.