BBFind

Version 4.0

Developers Reference

"The Ultimate Query Experience for 4th Dimension"

By Tony Ringsmuth
of
Business Brothers Inc.

Also known As
BBI and BBSP

Revision 11/25/2008

Contents

Preface

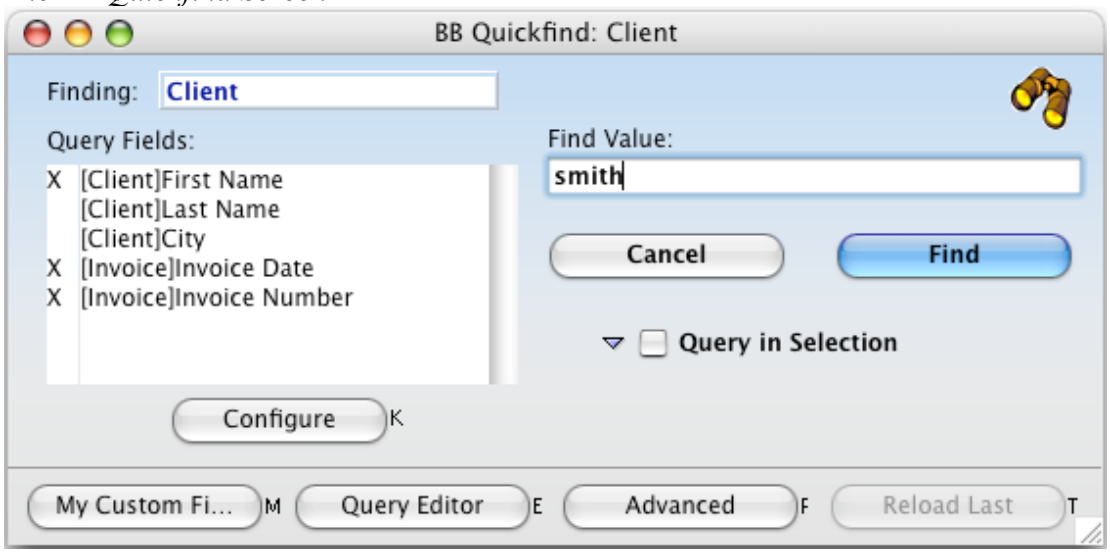
The BBFind component is a 4th Dimension component that enhances the ability to query records in your database. BBFind is designed to be easy to integrate into any 4th Dimension database. The entire BBFind mechanism can be called with as little as one short line of code. BBFind is also designed to be flexible. Therefore, BBFind offers a number of ways to quickly customize it to work best with your database. Some of the ways that you can easily customize BBFind using the 4th Dimension Language include:

- Configuring dynamic field names
- Configuring what fields are or are not visible to the user
- Adding Virtual fields
- Adding or removing logical relationships between tables
- Specifying your own dynamic choice lists for fields
- Modifying BBFind forms

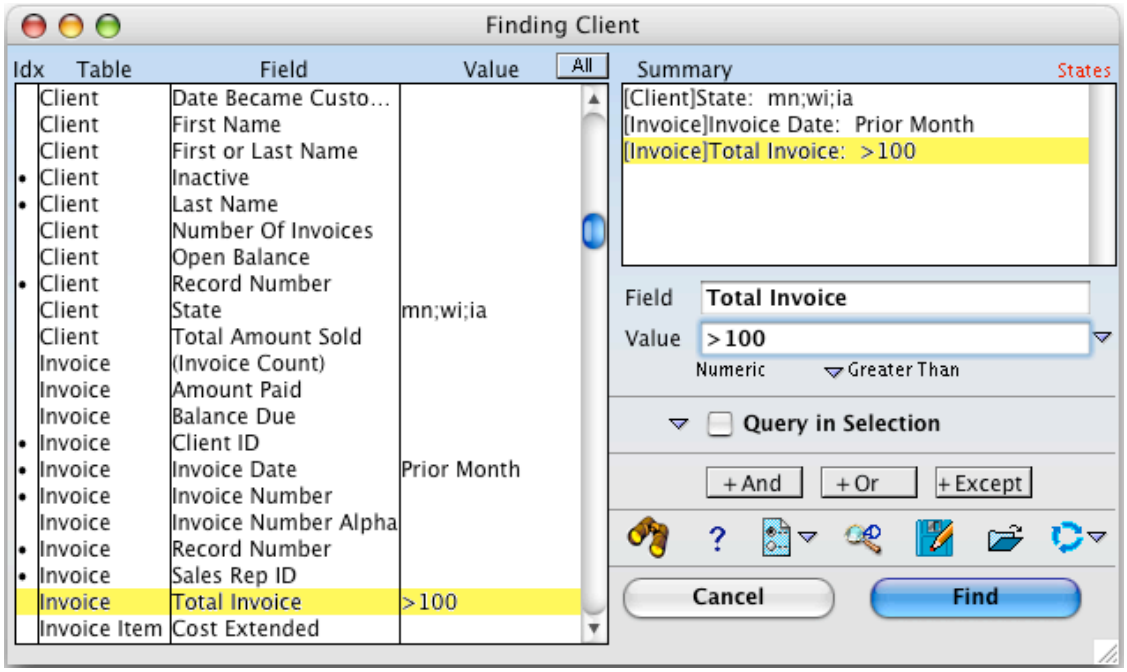
The flexibility and power of BBFind make it the ideal query tool for almost any 4th Dimension database.

Screen Shots of BBQuickFind and BB Advanced Find

The BB Quickfind Screen



The BB Advanced Find Screen



About This Manual

This manual is intended to give the developer everything he or she needs to work with BBFind in the development environment.

BBFind has many project methods that are not documented in this manual. All methods that are intended for the Developer to modify or call are documented in this manual. Most other methods are not documented here. However, most of these undocumented methods do have at least basic documentation within the method it's self.

This manual uses the following standards:

BBFind Project Method Names will be shown in **bold blue** letters.

Optional Parameters will be enclosed in {Curley Brackets}.

1
Installation
BEFORE YOU INSTALL: IMPORTANT

Before you install BBFind into your database, it is important that you first make a backup copy of your

BEFORE YOU INSTALL: IMPORTANT

FOR 4D V2004 AND PRIOR DATABASES ONLY: Before you install BBFind into your database, it is important that you first make a backup copy of your database. If something goes wrong, you want to be able to recover.

The BBFind component requires BBBase Component. BBBase is a component of general utility routines for array management, message displaying, record locking management and the likes.

FOR 4D V11+ DATABASES: BBFind is fully compatible with 4D V11.1 to V11.3. To install BBFind, just put BBComponents into the Components folder of your database. BBFind is provided with both Unicode and non-unicode compiled components: Be sure to use the file that corresponds to the Unicode setting of your database.

Tables contained in BBFIND and BBBASE

The BBFind and BBBase components for 4D V2004 each contain one database table. It is necessary for any 4D component which has Forms to also contain at least one Table. However, this table does NOT necessarily need to be installed into your database. **BBSP recommends that you create a table in your database similar to the specifications below and then "link" the tables in BBFind and BBBase to your table**. This table is generic and multi-purpose. It will be used to store dialogs and record for BBFind and BBBase. The exaxt names or order of the fields is not important.

The structure of [BB_Misc] is as follows:

Name	Type	Notes
ID	Alpha 80 (or more)	Indexed
Alpha1	Alpha 80(or more)	

Alpha2	Alpha 80(or more)	
Alpha3	Alpha 80(or more)	
Alpha4	*Alpha 80*(or more)	*Currently Not Used*
Text1	Text	
Time1	Time	
Date1	Date	
Longint1	Longint	Indexed
Blob1	*Blob*	*Currently Not Used*

The BBBase component provides you with the method "**BB_Base_StructureDeclarations**" which allows you to link fields in your database structure to the BBFind component. See also the section on **BB_Base_StructureDeclarations**

Reinstalling BBFind: IMPORTANT

When upgrading form versions of BBFind prior to 1.4, you must first un-install BBFind. You can then install BBBase and BBFind.

If you are reinstalling BBFind, 4D Insider will prompt you for each method that you have modified if you want to update the method with the new method from the component.

Naming Conventions

All objects of the following types are prefixed with "BB_"
Forms, Lists, Variables, Named Selections, Picture Library Pictures, Project Methods, Semaphores, Sets, Style Sheets, Tables, Tips

When installing the component into your database, if there are any naming conflicts between BBFind and your database, 4D Insider will alert you, abort the operation, and log the conflicting items in a log file. In this case, you must rename the items in your database so that they do not conflict with the BBFind component. After installing the component, you can then rename conflicting items back to their original name.

INSTALLATING BBFind INTO YOUR DATABASE

For 4D 2004:

1. Make a backup of your database structure. (This is recommended before installing any component)

2. Using 4D Insider version 6.8.1 or greater, open your Database. (BBFind is designed to work in 4D 6.8.x and 4D 2003.x and 4D 2004.x)

3. Under the "Component" menu, select "Install/Update"

4. Select and install the component BBBase. Be sure to Link the table BB_Base_Dlogs to a table in your database (do NOT copy the table).

5. Select and install the component BBFind. Be sure to Link the table BB_Find_Dlogs to a table in your database (do NOT copy the table).

5. Close 4D Insider.

For 4D V11:

Put the BBFind component into the “Components” folder of your database

2
Integrating BBFind into your database

The integration of BBFind falls into 3 categories
- Structure Linking: Telling BBFind where to find its records
- Invoking BBFind: How to call the BBFind screens for your users
- Customization: How to customize BBFind for your database

This section will cover each of these categories.

Structure Linking

Telling BBFind where to find its dialogs and records

You must call the method **BB_Base_StructureDeclarations** to tell BBFind where to go for dialogs and record storage. **BB_Base_StructureDeclarations** must be called before you use BBFind. It is recommended that you call **BB_ Base _StructureDeclarations** during database startup.

BBFind is designed that the dialogs can be on one table and its records can be stored in another table. If you choose to link BBFind to a table in your database, you should select the table where you want the forms will be stored. You can then link the table and fields for record storage to a different table if so desired.

Project Method: **BB_ Base _StructureDeclarations**
Function: Link the BBFind component to the desired table in your database

Parameters
$1: Longint: Reserved: Pass 0
$2: Text: Reserved: Pass “”
$3: Pointer to the table that contains all BBFind Dialogs.
$4: Pointer to the table that contains any saved queries or preferences Generally $3 and $4 are the same table
$5: Pointer to Utility ID field: should be 40 to 80 characters, indexed
$6: Pointer to Utility Alpha field 1: should be 40 to 80 characters
$7: Pointer to Utility Text field
$8: Pointer to Utility Longint field
$9: Pointer to Utility Alpha field 2: should be 40 to 80 characters
$10: Pointer to Utility Date field
$11: Pointer to Utility Time field
$12: Pointer to Utility Alpha field3: should be 40 to 80 characters

Example:

```
BB_Base_StructureDeclarations (0;"";->[BB_MISC];->[BB_MISC];->[BB_MISC]ID;
  ->[BB_MISC]A1;->[BB_MISC]T1;->[BB_MISC]L1;->[BB_MISC]A2;->[BB_MISC]D1;-
>[BB_MISC]H1;->[BB_MISC]A3)

BB_Base_StructureDeclarations (0;"";->[MyDialogTable];->[MY_UTILITY_TABLE];
  ->[MY_UTILITY_TABLE]ID;->[MY_UTILITY_TABLE]A1;->[MY_UTILITY_TABLE]T1;
  ->[MY_UTILITY_TABLE]L1;->[MY_UTILITY_TABLE]A2;->[MY_UTILITY_TABLE]D1;
  ->[MY_UTILITY_TABLE]H1;-> [MY_UTILITY_TABLE]A3)
```

Invoking BBFind

How to call the BBFind screens for your users

BBFind has two main user interface screens: BB Quickfind and BB Advanced Find. From BB Quickfind, the user can always go to BB Advanced Find. The syntax of each method that invokes each of the find screen is similar.

To invoke BB Quickfind, call
$iFindResult:=**BB_FindQuick**({ ->[QueryTable]} ;{ Options})

To invoke BB Advanced Find, call
$iFindResult:=**BB_Find**({ ->[QueryTable]} ;{ Options})

$0: $iFindResult: **BB_FindQuick**, and **BB_Find** each return a Longint result
1 = the user executed a Find
0 = the user cancelled, without executing a find

$1: ->[QueryTable]
QueryTable is the table in which the user will find a selection of records. This parameter is optional. If no parameter is passed, BB Find will use the current form table as the query table.

$2: Options:
This is an optional bitwise, Longint parameter. The bitwise portions of the parameter are as follows:
+1 (bit 0) = Force a Query In Selection
+2 (bit 1) = Open a new window for the BB Quickfind Dialog
+4 (bit 2) = Suppress Query Progress Window
+8 (bit 3) = Suppress BBFind auto inclusion of Related Record Count queries
+16 (bit 4) = Auto Load a query, of ID = BBFind_QueryID
+32 (bit 5) = For AutoLoad Query, if bit 5, then immediate accept
+64 (bit 6) = Allow Searches by Record Number

Examples:
BB_FindQuick Invokes BB Quickfind, for the current form table, in the current window

BB_FindQuick(->[Invoice];1) Invokes BB Quickfind for the Invoice Table, in the current window, and forces a Query within the current selection of records

BB_FindQuick(->[invoice];1+2) Invokes BB Quickfind for the Invoice Table, in a new window, and forces a Query within the current selection of records.

BB_Find uses the same parameters a **BB_FindQuick**

SUBTABLES

How to integrate the subtables of your database into BBFind

If your database does not use subtables, or you do not wish to have them available in BBFind to your users, than you can skip this section.

While most developers are avoiding the use of subtables, many legacy systems still have them. Because of this, BBSP went the extra mile to provide basic subtable support in BBFind.

There are some features that are not supported with subtables. These include: Choice List Pop-ups. Also, only one level of subtables is supported.

Because there is no 4D native way to get a list of subfields in a subtable, BBFind uses a free pluggin product by Rob Laveaux called API Pack. You only have to use this pluggin just once, while running in source code. After that, all subtable information is stored in a BBFind List called "BB_SubTableInfos". You can then remove API Pack if you like.

Here's what you need to do:

- After you have installed the BBFind component into your database
- Put the pluggin API Pack into your Mac4DX or Win4DX folder. (You only need it in the appropriate folder for the platform that you are running this process on)
- Start your database, and go to the BB Advanced Find Screen
- Click on the "More" dropdown list
- Select "Integrate Subtables"

After this, you can remove the pluggin. If you add or modify subtables or subfields, you should run this process again.

The information stored in the "BB_SubTableInfos" list is in the following format: One list item for each subfield
TableNumber;FieldNumber;SubFieldNumber;Type;Length;Index;Invisible;Name

You can manually construct or modify this list if you like.

API Pack can be downloaded from www.Pluggers.nl

Technical notes about querying subfields

Queries on subfields are given special attention when doing a “Not Equal To” query. BBFind converts a “Not Equal To” query to an “Except” query. Here is the difference:

Given that a record has 2 subrecords, where [person]child’name contains “Adam” and “Bob” in the same record:

A “normal” 4D query *Query([person];[person]child’name # “Adam”)* would return our example record in the subsequent selection, because one of the subrecords (“Bob”) was not equal to “Adam”.

An “Except” query, *Query([person];#;[person]child’name = “Adam”)* will NOT return our example record. The query returns all [person] records, except where [person]child’name = “Adam”.

This issue is NOT covered in the users reference, because it is deemed too technical for the grasp of most users.

The 4D Language reference briefly mentions the *except* operator, but does explain what it does. See the section on Queries.

Customization
How to customize BBFind for your database

While BBFind is an open-source component and can therefore be directly modified, it is designed that you can customize it's behavior through a variety of callback methods and other forms of customization. We recommend that you modify the component as little as possible so that if you choose to upgrade your BBFind to a future version, you will not loose your customizations. Using the technique of BBFind callbacks and the likes, you will retain your customizations during an upgrade to BBFind.

There are several ways that you can programmatically customize BBFind. Some specifics include:

- Changing and or hiding fields from the user
- Adding your own "Virtual" fields
- Managing the behavior of choice list action when users click on a field in BB Advanced Find
- Adding your own code to the form methods in BBFind
- Creating your own front end and using the BBFind engine to execute your queries
- Pre-populating values in the BB Advanced Find Screen, which the user can see or modify
- Forcing a "Query in Selection"
- Executing a programmed query on the results of the BBFind Query

The BBFind Demo database has some good examples of ways that you can do this.

You may also modify any of the forms within BBFind. However, BBSP may modify or add features to these forms which you will want to install into your database which would overwrite your customization to the forms.

The following subsection documents each method that is intended to be used by the developer to allow programmatic customizations to BBFind.

BB_Find_SetFieldTitles

BB_Find_SetFieldTitles(->Table;"TblName";->ArrayFieldNames;->ArrayFieldNumbers)

Parameters:
$1 = Pointer to a 4D Table
$2 = Table name that the user should see. Leave blank to use native 4D table name
$3 = Pointer to Array of field names that the user should see
$4 = Pointer to Array of field numbers for the field names

This method is a companion to the 4D Command "**SET FIELD TITLES**" If you want BBFind to use your dynamic table names and field titles, you must call **BB_Find_SetFieldTitles.**

If you exclude any fields in Table from ArrayFieldNames and ArrayFieldNumbers, then those fields will not be displayed in BBFind.

If ArrayFieldNames and ArrayFieldNumbers have no elements, then BBFind will revert to using the full list of field names, as stored in the database structure.

You must call **BB_Find_SetFieldTitles** for each table that you want to specify dynamic field names. **BB_Find_SetFieldTitles** configures field names across all processes. The scope **BB_Find_SetFieldTitles** of is inter-process.

Call **BB_Find_SetFieldTitles** before you invoke BBFind. A good place to call **BB_Find_SetFieldTitles** is on database startup.

Example

```
ARRAY TEXT(aMyFieldNames;3)
ARRAY INTEGER(aMyFieldNums;3)

aMyFieldNames{1}:="First Name"
aMyFieldNums{1}:=3  `Field number 3 in [Client]

aMyFieldNames{2}:="Last Name"
aMyFieldNums{2}:=2  `Field number 2 in [Client]

aMyFieldNames{3}:="City"
aMyFieldNums{3}:=5  `Field number 5 in [Client]

BB_Find_SetFieldTitles(->[Client];"My Client";->aMyFieldNames;->aMyFieldNums)
```

BB_Find_SetCallback

BB_Find_SetCallback (CallBack Name;CallBack Expression)

BBFind allows a great deal of customized behavior to your database via callback methods. This allows you to tell BBFind a name of your database's method to call for particular activity.

For V11, make sure that any callback methods that you specify are marked as "Shared by Host Database and Component". Your callback methods will NOT work in V11 if this is not set.

Description
The method BB_Find_SetCallBack is one the primary methods for customizing various BBFind. BB_Find_SetCallback installs your own custom methods or expressions as the action to execute upon certain specific BBFind Events.

Parameters
$1: Text: CallBack Name: This is the descriptive name of the callback event which you install.
$2: Text: CallBack Expression: This is a method name of a method that you create to handle various event. BBFind passes various parameters, such as the main query table to your method to give you the information you need to handle the event.

Example:
BB_Find_SetCallback ("Configure_Relationships";"My_BBFind_ConfigRelations")

Following is a list of each BBFind Callback:

Callback: AF_FieldClick

(Modified in v1.4.1)
- This callback allows you to configure what happens when the user clicks on a field name in Advanced Find.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table
$2: Longint: Table Number of field clicked on
$3: Longint: Field Number of field clicked on
$4: Longint: Subfield Number of field clicked on
$5: Text: Current value of query field
$0: Text: what your method passes back to BBFind

This callback functions in one of two ways:
- Method 1: You can present your own custom pop-up-menu or choices dialog when a user clicks on a field name in the Advanced Find screen, in place of the standard choice list that BBFind presents when the user clicks on a field with an attached choice list. You pass back in the value that the user selects in $0. When doing this, if the user cancels the selection of any item, you should pass back in $0, the contents of $5 (the prior selected value in the Advanced Find Screen). If you do not, and the user has previously entered a value, it will be lost.

Method 2: you can pass back to BBFind a list of values and let BBFind treat those values as a choice list. To do this, your method must pass the string "ListValues;" and then the list of values, separated by semi-colons. (semi-colons, or whatever multi-value delimiter that you may have specified with **BBFind_SetParameter**)

```
 `MyBB_Find_AF_FieldClick(iMainTbl;iTbl;iField;iSubField;CurrentValue)->ReturnVal
 `An example method for using the BBFind Callback "AF_FieldClick"

C_LONGINT($1;$iQueryTable;$2;$iTable;$3;$iField;$4;$iSubField)
C_TEXT($4;$CurrentValue;$0)
C_POINTER($pField)

$iQueryTable:=$1
$iTable:=$2
$iField:=$3
$iSubField:=$4
$CurrentValue:=$5

$0:=$CurrentValue  `So that the user won't lose their prior selection if they cancel

If (($iField >0)&( $iField <=Count fields($iTable)))  `Beware, it could be a Virtual field, or a Record Number field (0)
 $pField:=Field($iTable;$iField)
 $iType:=Type($pField->)
```

```
Case of
  : ($iType=Is Boolean )
     `I will manage the selection dialog here and pass back the results
    $iPop:=Pop up menu("Yes;No")
    Case of
      : ($iPop=1)
        $0:="Yes"
      : ($iPop=2)
        $0:="Yes"
    End case

  : ($pField=(->[Product]Type)
     `Let BBFind Manage the selection dialog for me
    $0:="ListValues;Fruits;Vegetables;Canned"

 End case

End if
`
```

Callback: AF_FormMethod

Place code here to further control the form [BB_Misc] "BB_Find" (BB Advanced Find Screen). This code will be executed during each form event for the form. The Callback method will be executed after the built-in Form Method.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

Example:

```
  `MyBB_Find_AF_FormMethod
C_LONGINT($1;$iMainTable)
$iMainTable:=$1

If (Form event=On Load )
  DisableAllMenus
End if
```

Callback: AF_Load_Virtual_Fields

Use this callback to insert you own “Virtual fields” (fields that do not exist in the database structure) into the Advanced Find screen.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table
$2: Longint: Table Number of the currently loading table.

This callback is called once for EACH table that is loaded into the Advanced Find screen. By testing $1 and $2, you can know which is the main query table ($1), and the currently loading table($2). You add the virtual fields to the currently loading table ($2).

Use the method BB_Find_AF_AddVirtualField to insert virtual fields. The method BB_Find_AF_AddVirtualField should only be used within the context of AF_Load_Virtual_Fields callback, and nowhere else.

When the user executes the query, BBFind will invoke another callback, “AF_Query_Virtual_Fields”, once for every virtual field that the user has populated to allow you to handle the query operation of the virtual fields.

Method: **BB_Find_AF_AddVirtualField**
Function: Act as an API for the developer to add their own “Virtual” fields whatever name and type to the Advanced Find Screen

$1: Text: Field name that you want the user to see
$2: Longint: 4D Table Number that you want this field to appear in
$3: Longint: Your own Virtual field number.
$4 Longint: Your own Virtual Subfield number.
$5: Longint: Data type of field
$6: Boolean: Should the field appear as indexed?

Example:

```
`MyBB_Find_AF_LoadVirtualField
C_LONGINT($1;$iMainTable;$2;$iLoadTable)
$iMainTable:=$1
$iLoadTable:=$2

Case of
  : ($iLoadTable=Table(->[Client]))
    BB_Find_AF_AddVirtualField ("First Or Last Name";Table(-
>[Client]);1024;0;Is Boolean ;False)

End case
```

Callback: AF_Load_Query_Form_Method

Place code here to further control the form [BB_Misc] “BB_FindLoad”. This code will be executed during each form event for the form. The Callback method will be executed after the built-in Form Method.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

Example:

```
  `MyBB_Find_AF_LoadQueryFM
C_LONGINT($1;$iMainTable)
$iMainTable:=$1

If (Form event=On Load )
  DisableAllMenus
End if
```

Callback: AF_Preload_Values

Place code here to pre-load values into the advanced find screen. The user can see and modify these values in their query.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

Method: BB_Find_PreLoadQueryValue

Function: Give the developer a way to populate query fields values before the Advanced Find is presented to the user.

PARAMETERS:
$1: Table Number
$2: Field Number
$3: Subfield Number (or zero)
$4: The Textual value that you want to populate

This method should be used within the context of the AF_Preload_Values and AF_Query_Load callback method.

Example:

```
If($iMainTable=table(->[client]))
  BB_Find_PreLoadQueryValue (table(->[client]);field(->[client]inactive);0;"False")
End if
```

Callback: *AF_PreQuery*

This callback method is called after the user has completed the BB Advanced Find screen, but before the BBFind engine executes the query. You may limit the selection of records for this table prior to the execution of the BBFind engine. The BBFind Engine will only query within the selection that you create for QueryTable.

The intention here is that you may SPEED UP the user's query by first eliminating records (via an indexed query of your own) that the user is not allowed to access.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

This method is only called prior to an Advanced Find, NOT a Quickfind
Example:

Example:

```
  `MyBB_Find_AF_PreQuery
C_LONGINT($1;$iMainTable)
$iMainTable:=$1

Case of
  : ($iMainTable=Table(->[Client]))
     QUERY ([Client];[Client]Inactive=False)
End case
```

Callback: *AF_Query_Load*

This is invoked by BB Advanced Find screen, any time a previously executed (recent queries or saved queries) query is loaded into the AF Screen.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

Example:

```
If($iMainTable=table(->[client]))
  BB_Find_PreLoadQueryValue (table(->[client]);field(->[client]inactive);0;"False")
End if
```

Callback: AF_Query_Virtual_Fields

This callback allows you to insert your own code to handle the querying of Virtual fields that you have inserted into the Advanced Find screen.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table
$2: Text: Field Name (as seen by user)
$3: Longint: Query Table Number
$4: Longint: Field Number (this is the virtual field number that you assigned)
$5: Longint: SubField Number
$6: Text: Operator
$7: Text: Value

Example:

```
`MyBBFind_QueryVirtualFields

C_LONGINT($1;$iMainTable)
C_TEXT($2;$FieldName)
C_LONGINT($3;$iQueryTableNum)
C_LONGINT($4;$iQueryFieldNum)
C_LONGINT($5;$iSubFieldNum)
C_LONGINT($6;$Operator)
C_TEXT($7;$QueryValue)

$iMainTable:=$1
$FieldName:=$2
$iQueryTableNum:=$3
$iQueryFieldNum:=$4
$iSubFieldNum:=$5
$Operator:=$6
$QueryValue:=$7


Case of
    : ($FieldName="Phonetic Name")

    $PhoneticValue:=phonetic($QueryValue)
    QUERY SELECTION([People];[People]PhoneticName=$PhoneticValue)

End case
`
```

Callback: *Configure_Relationships*

Add or remove tables from the scope of a Find. See Also the section on **BB_Find_AddRelatedTable** and **BB_Find_RemoveRelatedTable**

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

Example

```
`MyBB_Find_ConfigRelations(iMainTable)
`a Demo Method

` See Also: MyBBFind_Initialization, BB_Find_SetCallback

C_LONGINT($1;$iMainTable)

$iMainTable:=$1

Case of
  : ($iMainTable=Table(->[Client]))
    `If querying  [Client] , add [Product] to the query by their
    `  relationship with [Invoice_item]
   BB_Find_AddRelatedTable (->[Product]Product_ID;->[Invoice_Item]Product_ID)


  : ($iMainTable=Table(->[Invoice]))
    `If querying  [Invoice] , add [Product] to the query by their
    `  relationship with [Invoice_item]
   BB_Find_AddRelatedTable (->[Product]Product_ID;->[Invoice_Item]Product_ID)


  : ($iMainTable=Table(->[Invoice_Item]))
    `If querying [Invoice_Item], remove [Sales_Rep] from the query.
    `      [Sales_Rep] is related through [Invoice]
   BB_Find_AddRelatedTable (->[Sales_Rep]Sales_Rep_ID;->[Invoice]Sales_Rep_ID)

End case
`
```

Callback: My_Custom_Query

If you are using BB Quickfind or BB Advanced Find as your primary query screen, you may still have the need for the user to perform other queries which you need to specifically program.

BBQuickFind provides a "Custom" button which is disabled and set to invisible by default. If you specify a My_Custom_Query, BBFind will enable and set to visible the Custom Button (and an associated hot key tip). The object name of this button is "CustomQuery_Btn", and the keyboard hint is named "CustomQuery_HotKey". You can rename the button from "Custom" to any name that you want, using the QF_FormMethod callback.

If you specify a My_Custom_Query callback, and when the user clicks on this button, then BBFind will exit the current BBFind query form (Quickfind or Advanced Find) and execute the installed My_Custom_Query callback method.

Keep in mind, that if the user has selected "Query in selection", "Add found record to selection" or "Remove found records from selection", BBFind will still attempt to enforce these options after ***My_Custom_Query*** has completed.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

Callback: *Post_Query*

Put your own code here to do any kind of selection management prior to the user being presented with their selection of records. This method is executed AFTER the BB_Find Query Engine has executed its query. This method will be called after BB Quickfind and BB Advanced Find.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

Example:

```
Case of
  : ($iMainTable=Table(->[Client]))
    QUERY ([Client];[Client]Inactive=False)
End case
```

Callback: QF_FormMethod

Place code here to further control the form [BB_Misc] “BB_FindQuick” (BBQuickfind Screen). This code will be executed during each form event for the form. The Callback method will be executed after the built-in Form Method.

List of parameters that BBFind passes to your callback method:
$1: Longint: Table Number of main query table

Example:

```
  `MyBB_Find_QF_FormMethod
C_LONGINT($1;$iMainTable)
$iMainTable:=$1

If (Form event=On Load )
   `It’s preferable to reference BBFind object by object name instead of Variable names so that if you re-install BBFind, there will not be variable naming conflicts
  BUTTON TEXT(*;”CustomQuery_Btn”;”XYZ Find”)
End if
```

BB_Find_SetParameter

BB_Find_SetParameter (Parameter Name;Parameter Value)

Description
This method is very similar to BB_Find_SetCallback, except that a BBFind Parameter is a literal text string, NOT an expression. Parameters are used to specify specific ways that BBFind will work.

BB_Find_SetParameter is inter-process in scope. Generally, you should call **BB_Find_SetParameter** once, on database startup.

Parameters
$1: Text: Parameter Name: This is the descriptive name of the parameter which you set.
$2: Text: Parameter Value: This is a alpha-numeric value.

Table of Parameter Names

Parameter Name	Notes
AF_AllowUserIndexing	Values "Yes" (for ON) or "" (for OFF). (the default of off). If you pass a "Yes", then, the user can contextual click on the Index column of the Advanced Find Screen, and will receive a pop-up menu with an option to index fields. BBFind does not allow the user to un-index a field.
AF_FirstEntryObject	Values: "Field" or "Value". This configures the behavior of the BB Advanced Find Screen, which entry gets the cursor first: The "Field" name entry or the "Value" name entry
AutoWildCard	Values: "OFF" or "" (blank). This command allows you to turn off BBFind's auto wild card feature for alpha fields. The default for BBFind is that Alpha fields query with an implied wild character at the end unless the user enters the "!" at the end of their value. You can turn this feature OFF by passing the value "OFF" for this parameter.
Default_AdvFind_WinType	Pass in string form, any valid 4D Window type. This value will set the default window type that BBFind will use for the Advanced Find screen.
Default_QuickFind_WinType	Pass in string form, any valid 4D Window type. This value will set the default window type that BBFind will use for the Quick Find screen.
MultiValue_Delimiter	By default, BBFind uses a semi-colon as a delimiter between values for a multi-value query. You can change this by specifying a different value for "MultiValue_Delimiter". The delimiter may be as long as 7 characters. Example: **BB_Find_SetParameter** ("MultiValue_Delimiter";**",**")
Record_Number_Field_Tag	Value: {the name users will see for "Record Number"}: If you allow your users to query by Record Number (Pass a +64 in the options parameter of BB_Find), then be default the Record Number field for each table appears as "Record Number". However, you can set this to read as whatever you want with this parameter. The name of this psudo field will be displayed the same for all tables.
User_Name	If not set, this parameter will default to the value of the 4D Function "Current User". Populate the User_Name parameter with a unique identifier for the

	current system user. This value will be used to save the users preferences and saved queries. **BB_Find_SetParameter** (“User_Name”;**Current User**)

BB_Find_AddRelatedTable

BB_Find_AddRelatedTable (->[RelatedTable]Id_Field;->[RelatingTable]Id_Field)

Description

This method will add a table and its relationship the tables and fields available in BBQuickfind and BB Advanced Find. You should call this method from within a ***Configure_Relationships*** callback method.

Parameters

$1: a pointer to the relating field in the table that you wish to add into the relationship
$2: a pointer to the relating field to which you will add a related table

Example

```
If (BB_Find_GetQueryTable=(->[Client]))
  BB_Find_AddRelatedTable (->[Product]Product_ID;->[Invoice_Item]Product_ID)
End if
```

For more information, see the example database structure below.

BB_Find_RemoveRelatedTable

BB_Find_RemoveRelatedTable(->[table1]Field;->[table2]Field)

Description
This method removes the relationship between the two fields, so that it will be ignored by BB_Find. It does not matter which direction the relationship points: **BB_Find_RemoveRelations** removes both One to Many and Many to One relationships.

You should call this method from within a ***Configure_Relationships*** callback method.

Parameters
$1: a pointer to the relating field in the table that you wish to remove the relationship
$2: a pointer to the related field of which you will remove related table

Example: Both of the following examples achieve the same results

```
If (BB_Find_GetQueryTable=(->[Client]))
 BB_Find_RemoveRelatedTable (->[Invoice_Item]Invoice_Number;->[Invoice]Invoice_Number)
End if

If (BB_Find_GetQueryTable=(->[Client]))
 BB_Find_RemoveRelatedTable (->[Invoice]Invoice_Number;->[Invoice_Item]Invoice_Number)
End if
```

BB_FindQuick_SetFieldList

BB_FindQuick_SetFieldList(opts;->MainTable;{->ArrayFields};{ArraySelect})

Description
Allow the programmer to programmatically specify the list of fields that will appear in the BB QuickFind screen.

The list you specify is for the current process only, and is not saved into data.

The list of fields that you specify will override any list that the user may have configured.

If you specify no fields, then the user's configured list will be used.

The list of fields that you specify will also be used for any subsequent calls to BB_Find_Engine_Execute for the given process.

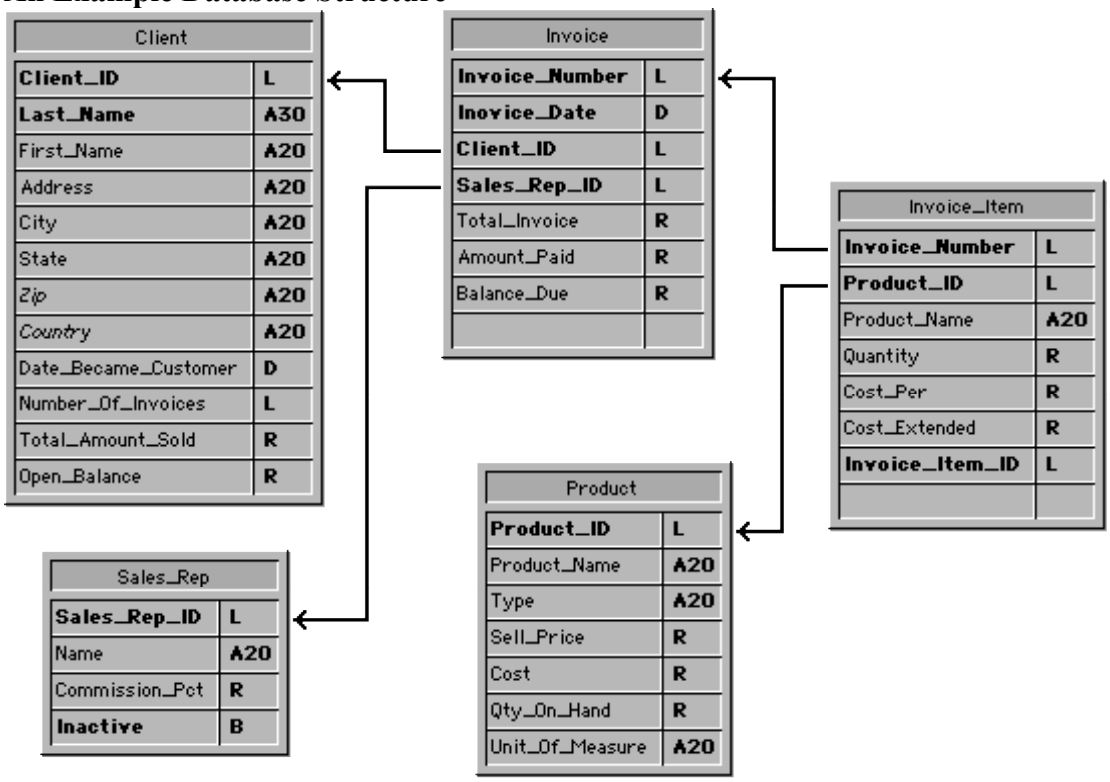
Parameters:
$1: Longint: Bitwise options (reserved)
$2: Pointer: a pointer to the table for which you are configuring the list of fields to query
$3: {optional} Pointer to a pointer array of fields: If $3 and above are not passed, then any configured list for this table will be removed and disregarded.
$4: {optional} Pointer to a numeric or Boolean array: This array is parallel to the array of fields. Values of 1 or true configure the field to be selected for querying. Values of zero or false configure the field to be deselected for querying. If $4 is omitted, then all fields are set to "Selected".

Example:

```
Array Pointer(aFieldList;3)
aFieldList{1}:=->[Client]Name
aFieldList{2}:=->[Invoice]Number
aFieldList{3}:=->[Invoice]Date
```

BB_FindQuick_SetFieldList **(0;->[invoice];->aFieldList)**

An Example Database Structure



BBFind will automatically find all related one tables and all related many tables related to your query table. In the example above:

if [Invoice] is your query table, BBFind will add [Client] and [Invoice_item]

if [Client] is your query table, BBFind will add [Invoice] and [Invoice_item]

if [Invoice_Item] is your query table, BBFind will add [Product], [Invoice], [Sales_Rep] and [Client].

When BB_Find is loaded each time, it find all tables related to the Query table, through the closest relationship, multiple levels up or down. You do not have to do anything additional to define relationships that are already drawn in your database structure.

If a circular relation exists, BBFind ignores the final relation in the relationship.

You may want to add to or remove from the relationships that have been drawn in your database structure. The methods "**BB_Find_AddRelatedTable**" and "**BB_Find_RemoveRelatedTable**" have been created to do this. See the documentation of each of these methods in the preceding pages.

API Calls

This sections lists additional Methods that are used as API (application programmer interface) calls, either to get information from BBFind, or to pass information to BBFind.

BB_Find_GetLastQuery

BB_Find_GetLastQuery(TableNumber;{ProcesNumber};{Options}) ->Text

Description
This method returns in Tab – Carriage Return delimited text, the information that comprises the last Advanced Find performed by the user. One possible use for this information would be to print out on a report.

Parameters
$1: Longint: Table Number of the table for which you want the last query
$2: Longint: Optional: If specified, and if greater than zero, then only return the most recent query from the specified process.
$3: Longint: Optional: Bitwise Options
+1: Return field Names, instead of Table & field numbers
$0: Tab, Carriage Return Delimited Text.

If you pass 1 in options, the format of $0 is:
[tablename]fieldName <tab> QueryData <cr>

if +1 is NOT passed in options, the format of $0 is:
TableNumber <tab> fieldNumber <tab> QueryData <tab> subfieldNumber <cr>

Example:
vMyReportTitle:=BB_Find_GetLastQuery(table(current form table);current process;1)

Using the BBFind Engine without the BBFind Interface

Let's say that you want to develop your own custom interface: a Query by Example perhaps. BBFind provides you good tools to develop you forms while still using some of the features of BBFind. Here are some of the features that you can use while using your own form.

- Date keyword searching (Today, Current month, etc, etc)
- Multi values in a single entry field (Value1;Value2;Value3)
- Range searching within a field (Value1 to ValueN)
- Entering operators within the data (=, >=, etc)
- Accelerated query speed across table relationships

This section documents the methods that enable you to accomplish this.

BB Advanced Find engine: To use the BB Advanced Find engine, you will use 3 API Calls:
BB_Find_Engine_Init: To initialize the call
BB_Find_Engine_PopulateFld: Once for each field to populate
BB_Find_Engine_Execute: To execute the query

BB_Find_Engine_Init

BB_Find_Engine_Init (->[QueryTable])

Description
This method initializes the BBFind Advanced Find engine to receive query values without the BBFind user interface

Parameters
$1: Pointer to the Query Table

Example
This example is taken from the BBFind Demo Database, in the form [Client] QueryByExample form method.

BB_Find_Engine_Init (Current Default Table)

BB_Find_Engine_PopulateFld (->[Client]Last_Name;vLastName)
BB_Find_Engine_PopulateFld (->[Client]First_Name;vFirstName)
BB_Find_Engine_PopulateFld (->[Client]Address;vAddress)
BB_Find_Engine_PopulateFld (->[Client]City;vCity)
BB_Find_Engine_PopulateFld (->[Client]State;vState)
BB_Find_Engine_PopulateFld (->[Client]Total_Amount_Sold;vTotalSold)
BB_Find_Engine_PopulateFld (->[Client]Open_Balance;vClientOpenBalance)
BB_Find_Engine_PopulateFld (->[Invoice]Invoice_Date;vInvoiceDate)

BB_Find_Engine_Execute

BB_Find_Engine_PopulateFld

BB_Find_Engine_PopulateFld(->QueryField;QueryValue)

Description
This method populates the value for Query Field. Use this method to use the BBFind engine independently of the BBFind Forms.

Parameters
$1: Pointer to a Query Field
$2: Text: The value to query in Query Field

Example:
See the example in **BB_Find_Engine_Init.**

BB_Find_Engine_Execute

BB_Find_Engine_Execute(QuerySelectionOptions)

Description
This method uses the BBFind engine to execute a query. You must first initialize the query with **BB_Find_Engine_Init**, and populate the query with **BB_Find_Engine_PopulateFld**. Use this method for your own Query by Example screens, or other programmed queries where you don't want to use the BBFind Screens.

Parameters
$1: Longint: Query Selection Options.
0 = Query all records
1 = Query in Selection
2 = Add found records to current selection
3 = Remove found records from current selection

Example:
See the example in **BB_Find_Engine_Init.**

BB_FindQuick_Engine_Execute

BB_FindQuick_Engine_Execute(->Table ;Query Value;{Group Name};{Selection Options};{General Options})

Description
This method is an API to call the BB QuickFind engine. Calling this method performs a query, just as if the user had performed a query using the BB QuickFind screen. This is the only method that you need to call in order to invoke an interfaceless QuickFind.

Parameters:
$1: Pointer: Main table to query
$2: Text: the Query Value
$3: Text: Group/Configuration name: Can be one of the following values:
"Default" or "": Loads whatever is specified as the default configuration for this user
"All_Users_Config": Loads the configuration for All Users. If not found, then the MyGroup configuration will be loaded (if exists).
"My_Group_Config": Loads the configuration for this users specified group.
"Personal_Config": Loads the configuration for this user's personal configuration.
{Group or configuration Name}: Loads the QF configuration for the named Group by this name.
For each of the above configuration names, if the configuration is not found, then BBFind will load load the All Users configuration.
$4: Longint: Selection Options
0: Query all records
1: Query in selection
2: Add found records to selection
3: Remove found records from selection
$5: Longint: General Options: a Bitwise Parameter
+4 (bit 2) = Suppress BBFind Progress windows when running query

Example:
BB_FindQuick_Engine_Init (->[invoice];"ABC Company";"Accounting";0;4)
`Execute a QuickFind on table Invoice,
` querying for the data "ABC Company",
` using the "Accounting" group BBQuickFind configuration,
` querying all records,
` and suppressing the BBQuickFind progress messages.