
epi info™
Guide Series
Epi Info Guide to Check Code

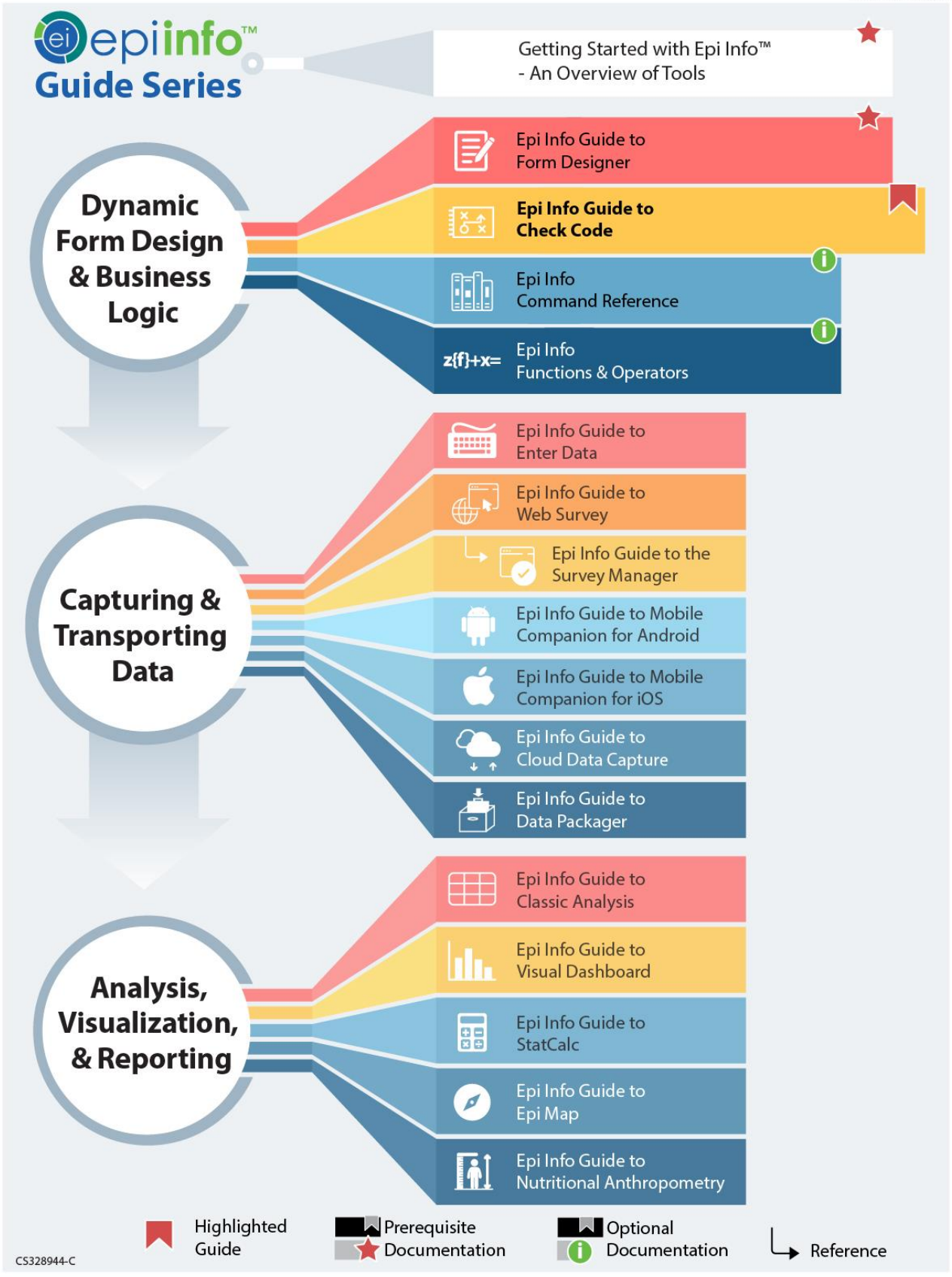
EPI INFO GUIDE TO CHECK CODE

Preface

Epi Info™ is a public domain suite of interoperable software tools designed for the global community of public health practitioners and researchers. Perform data analysis with epidemiologic statistics, maps, and graphs. Build data entry forms, construct a database, and customize statistics applications. Physicians, Epidemiologists, and public health officials without a technical background can easily work with critical data using Epi Info™ tools.

Use this guide alone or as a supplement to other Epi Info™ guides. On the next page, view the complete collection of helpful guides available for Epi Info™ tools.

The diagram shows how this guide highlighted with the bookmark icon fits into the big picture of the Epi Info™ suite. Additional guides may offer prerequisite information for the tool you're working with. For example, if you're working with Epi Info™ Check Code (EICC), then you are using code to customize fields in Form Designer. Therefore, you may need to begin by consulting the Epi Info™ Guide to Form Designer and Epi Info™ Guide as indicated by the prerequisite documentation icon ( ). Additionally, you may find helpful other guides that are indicated by the optional documentation icon ( ).


epiinfo™
Guide Series
Getting Started with Epi Info™
- An Overview of Tools
Dynamic Form Design & Business Logic
Epi Info Guide to Form Designer
Epi Info Guide to Check Code
Epi Info Command Reference
Epi Info Functions & Operators
Capturing & Transporting Data
Epi Info Guide to Enter Data
Epi Info Guide to Web Survey
Epi Info Guide to the Survey Manager
Epi Info Guide to Mobile Companion for Android
Epi Info Guide to Mobile Companion for iOS
Epi Info Guide to Cloud Data Capture
Epi Info Guide to Data Packager
Analysis, Visualization, & Reporting
Epi Info Guide to Classic Analysis
Epi Info Guide to Visual Dashboard
Epi Info Guide to StatCalc
Epi Info Guide to Epi Map
Epi Info Guide to Nutritional Anthropometry
Highlighted Guide
Prerequisite Documentation
Optional Documentation
Reference
CS328944-C

This page has been intentionally left blank.

TABLE OF CONTENTS

This Page has been intentionally left blank.

Check Code: Customizing the Data Entry Process

Introduction

The Check Code tool allows users to customize the data entry process. Use Check Code to check for errors during data entry, conduct automatic calculations in survey fields, and to skip parts of the survey if they meet certain conditions. Protect collected data against many common errors by creating rules for data entry. Check Code operations run each time a participant enters data or when triggered by certain values. Check Code is optional but advantageous. It provides a customized data entry process.

Check Code operations include:

- Displaying messages that appear to be part of the questionnaire
- Calculating fields from mathematical operations
- Checking one or more fields for relationships (e.g., making sure birth date is earlier than current date)
- Checking fields for inconsistencies (e.g., vaccination date is entered but response stated that record was not vaccinated)
- Displaying error messages because of improper entries in a field
- Complex statistics or other operations, written in other languages
- Field automatic indexing for faster searching
- Automatic searches during data entry

Accessing Check Code Program Editor

Click the Check Code **button** to navigate to the Check Code **Program** Editor in the **Form Designer** tool bar after opening your Epi Info™ 7 project.

Check Code button

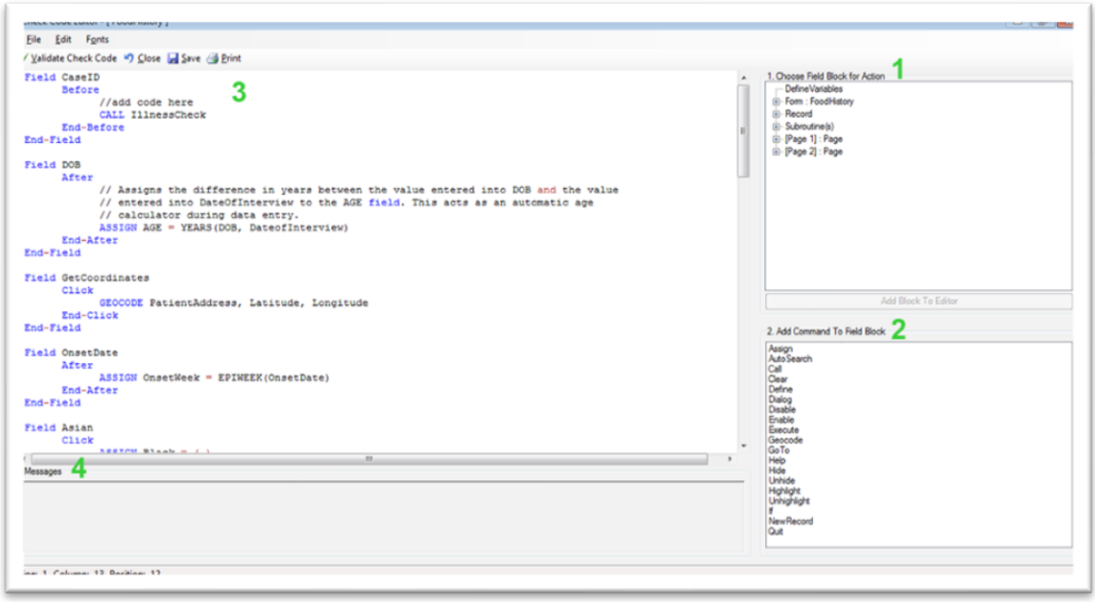
You can also select **Tools > Check Code** Editor from the Form Designer navigation menu.

Form Designer menu

Navigating Check Code Program Editor

The Check Code Editor window contains four sections:

- Choose Field Block for Action
- Add Command to Field Block
- Program Editor
- Messages



Check Code Program Editor

Sections:

1. **Choose Field Block for Action** tree — allows you to select form fields and establishes when Check Code Commands occur during data entry.
2. **Add Command to Field Block** window — displays all current Check Code commands in the **Form Designer** program.
3. **Program** Editor window — displays the code generated by commands created from **Choose Field Block for Action** or **Add Command to Field**

Block window. Users can also type directly and save the code into the **Program** Editor.

4. **Message** window — alerts the user of any check command problems.

There are several options on the toolbar at the top left, allowing you to **save, edit, validate, and change fonts** in **Program** Editor.

Close Check Code **Program** Editor and return to your form by clicking on the X button at the top right of your screen. You can also click on the **Close** button or press the **F10** key.

Check Code Commands

Check Code commands must be within a field-name block. Command blocks begins with a field, page or form name and end with the word **End**. Commands in a block, activate before or after entering data into a field. For some field types, blocks activate when clicking on the control (i.e.: checkboxes and command buttons). Usually, commands activate after the user presses the **Enter** key, or when the cursor has left the field. Change command activation by placing commands in **Before** blocks. Below is a list of typical Check Code formatting:

```
Field <Variablename>
After
—Check Code syntax inserted here—
End-After
End-Field
```

- **Field** parameter establishes a field name and the beginning of a corresponding Check Code block.
- **After** parameter specifies when actions occur. An **After** event executes as soon as the cursor leaves the field. **Before** and **Click** are the other two events supported. A **Click** event executes when the user clicks on the field and is only supported for Legal Values, Comment Legal, Checkboxes, and Command buttons. The **Before** event executes as soon as the cursor moves into the field.
- **End-After** parameter specifies the end of the command execution of Check Code**.** Code is placed between the **After** and **End-After** sections and code gets executed after the cursor leaves the field.
- **End-Field** parameter closes a block of commands for the corresponding field.

In the example below, we have incorporated an **After** event for a field called **DOB**. The Check Code block executes a value assignment to the field **Age**, using the **YEARS** function. The **YEARS** function calculates the difference between two date fields and provides the results in number of years.

```
Field DOB
After
ASSIGN AGE = YEARS(DOB, SYSTEMDATE)
End-After
End-Field
```

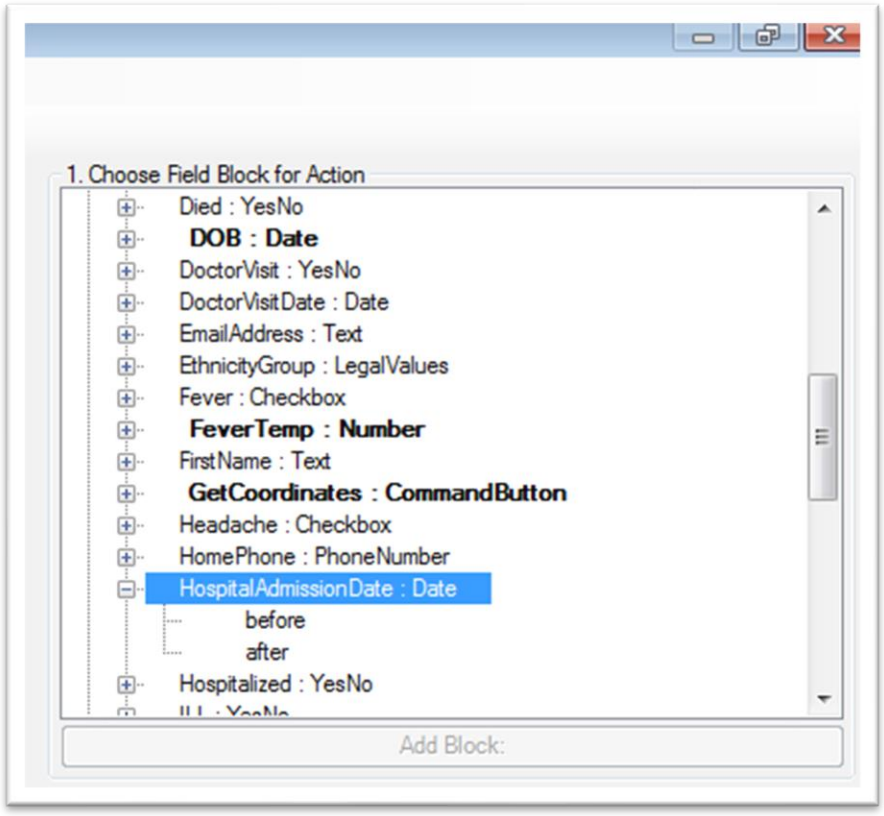
Basic Check Code Command Rules

1. Always place Check Code commands in a block corresponding to a variable/field in the form. Execute commands in provided sections, before or after you display a form, page, or record.
2. Place comments preceded by two forward slashes ("//") within blocks of commands. Comments are ignored during the execution of Check Code.
3. Block commands activate:
    - Before or after you make an entry in the field or when you click on the field (if Legal Value, Comment Legal, Checkbox, or Command button).
    - After completing an entry with **Enter, PgUp, PgDn**, or **Tab.**
    - When a command moves the cursor out of the field (e.g., **GOTO**).
4. Form field records, contain Check commands.
5. User interaction with the dialog boxes triggers Check commands. Syntax is displayed in the Check Code Editor. You can edit and save text in the **Program** Editor.
6. Insert **Before** and **After** commands into forms, fields, pages, or records.

Create a Check Code Block

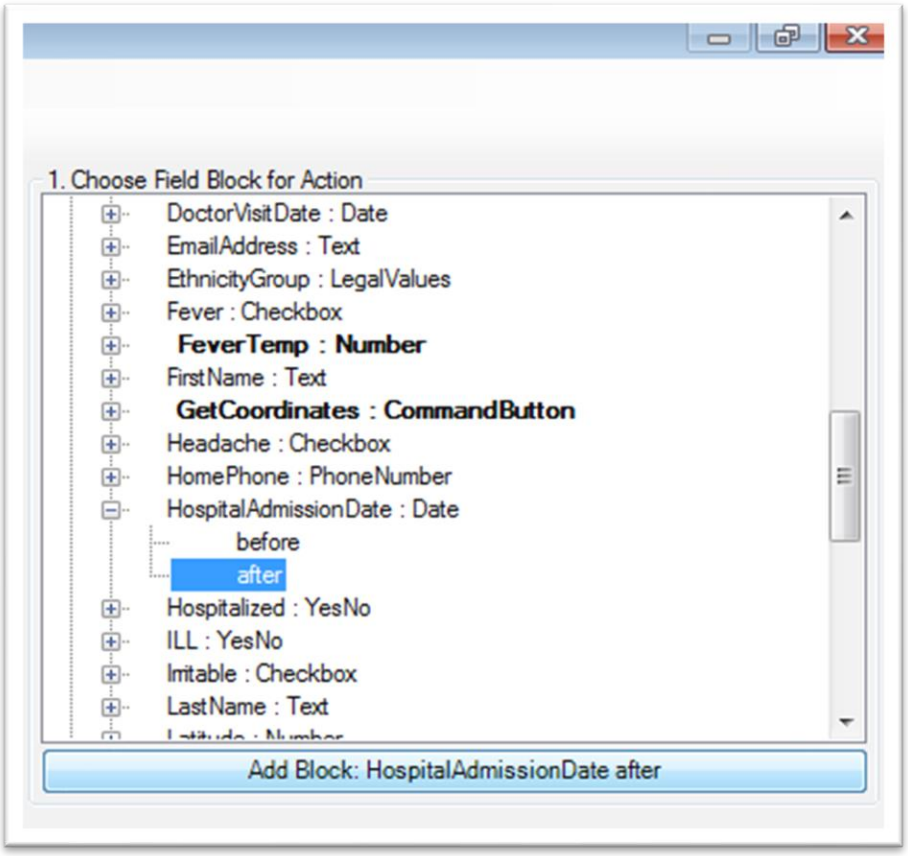
Steps:

1. Select and expand the form, page, record, or field that will receive the commands. Expand by clicking the **+** sign.



Choose Field Block for Action: Date

2. Select command execution. (i.e.: **before** or **after** data entry into the form, page, record, or field. View the example using **after**, below).



After event

3. Double-click on the **after** event or click on the **after** event and then click the **Add Block** button to insert the block. The Check Code block appears in the Check Code editor.

```
Field HospitalAdmissionDate
      After
            //add code here

      End-After
End-Field
```

Code Block added

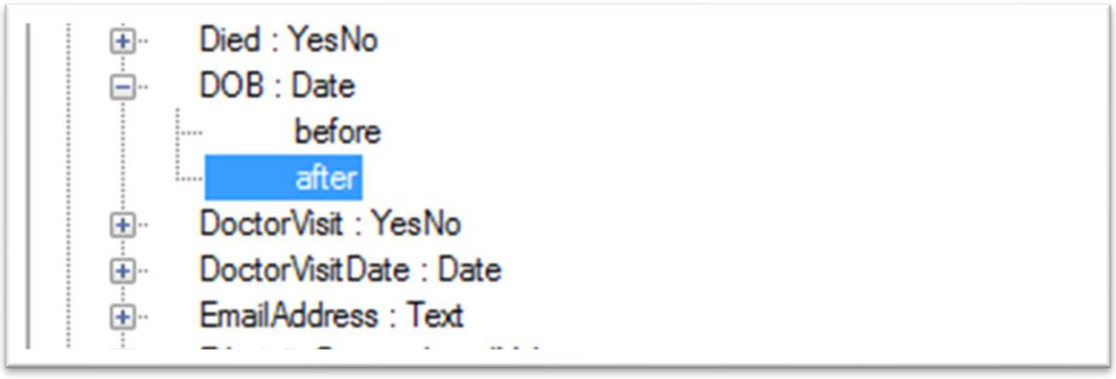
Insert commands within a new Check Code block using the **Add Command to Field Block** section.

Create a Skip Pattern with GOTO

Create skip patterns by changing tab order and setting new cursor sequences in forms. Try creating Check Code using the **GOTO** command. Create skip patterns for answers to questions using **IF/THEN** statements. In the following example, added Check Code moves the cursor to the **Ethnicity** field after participants enter data into the **DOB** field and skipping the **AGE** field.

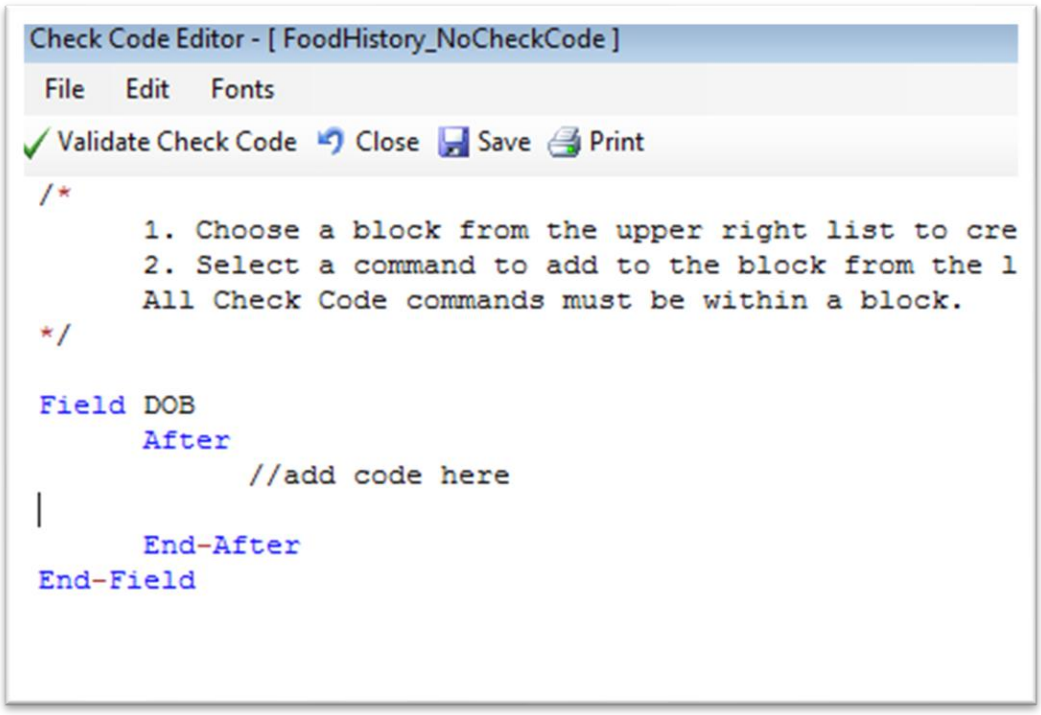
Steps:

1. Open the **Ecoli.prj** project from **Form Designer**,
2. Double click on the **FoodHistory_NoCheckCode** form.
3. Click Check Code or select **Tools >** Check Code Editor. The Check Code Editor opens.
4. Expand the node for Page 1, from the **Choose Field Block for Action** section and view the various fields on the first page.
5. Expand the node for the field **DOB**.



Dialog Block for action After DOB

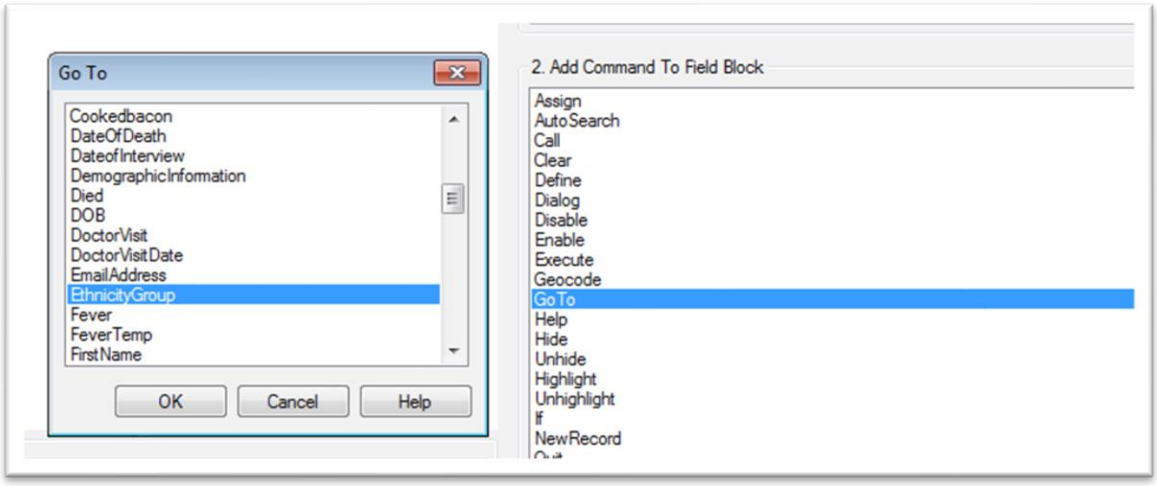
6. Double click on the **after** event.
7. A block of code for the **DOB** field displays in the Check Code editor.



Check Code Block for Action

Click **GOTO** from the **Add Command To Field Block** list box. The **GOTO** dialog box opens.

Select the **EthnicityGroup** field for cursor placement, after an entry in the **DOB** field. The code will run after the cursor leaves the field.



Skip Pattern GoTo dialog box

Click **OK**. The code appears in the Check Code Editor.

```
Field DOB
      After
            //add code here
            GOTO EthnicityGroup

      End-After
End-Field
```

Skip Pattern Check Code Command

Follow the next series of actions to verify your syntax:

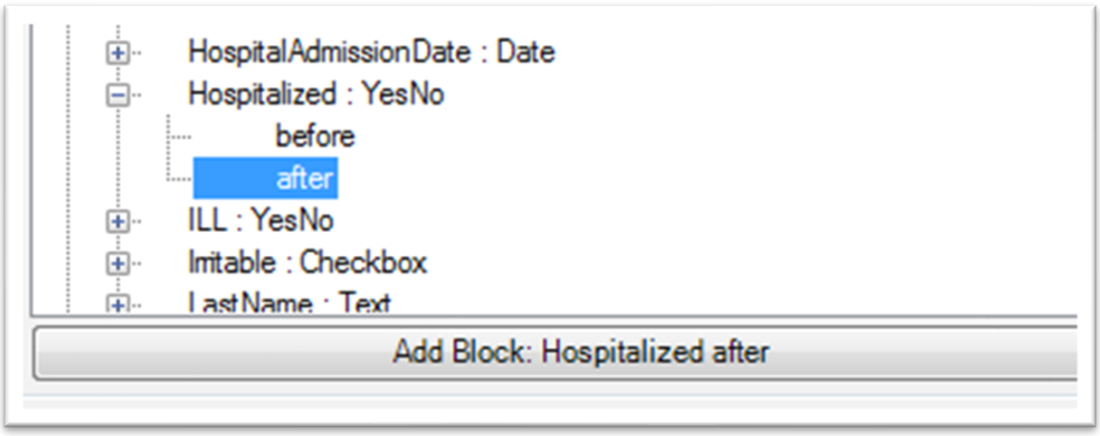
- Click **Verify** Check Code from the Check Code Editor.
- Click **Save** from the Check Code Editor.
- Click **Close** to return to the form.

Test the skip pattern by opening the form in the **Enter Data** tool. Move to the Data of Birth field and press the tab key to execute the **GOTO** command.

Create a skip pattern using **IF/THEN** and **GOTO.**

Use **IF/THEN** statements to create skip patterns based on answers to questions in the form. This example creates code for a participant who answers "No" for the **Hospitalized** field, then the cursor jumps to the field: **Was the patient treated with antibiotics?** Then, skips the **Hospital Admission date** field.

1. From **Form Designer**, Open the **Ecoli.prj** project.
2. Double click on the **FoodHistory_NoCheckCode** form.
3. Click Check Code or select **Tools >** Check Code Editor. The Check Code Editor opens.
4. From the **Choose Field Block for Action** section, expand the node for the first page and view the various fields.
5. Expand the node for the **Hospitalized** field.
6. The action occurs **after** a respondent fills in the **Hospitalized** field.
7. Double click on the **after** event.
8. Check Code Editor displays a block of code for the **Hospitalized** field.



Dialog Block for Action After Hospitalized

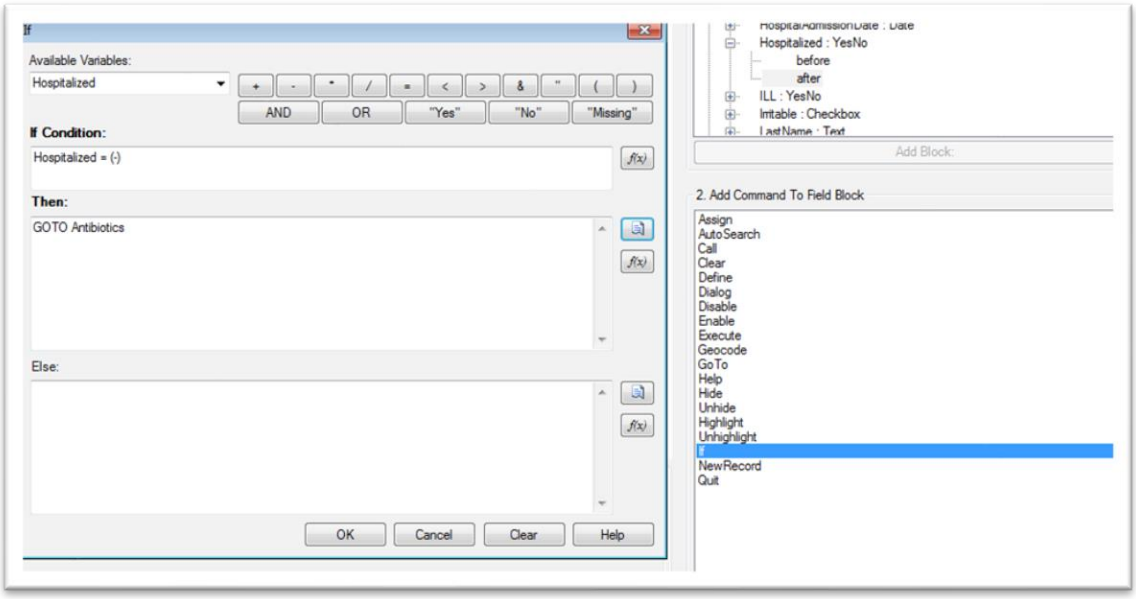
1. The Check Code Editor displays the code.

```
Field Hospitalized
      After
            //add code here

      End-After
End-Field
```

IF/THEN Block Code After Hospitalized

1. Click **If** from the **Add Command to Field Block** list box. The **If** dialog box opens.
2. Select the field from the **Available Variables** drop-down list to contain the action. For this example, select **Hospitalized**. The selected variable appears in the **If** Condition field.
3. Click the **=** operator from the operators list,
4. Click **No f**rom the operators list. The If Condition field will read Hospitalized **= (-)**.
5. Click the code snippet button in the Then section. A list of commands appears.
6. From the command list, select **GOTO**. The **GOTO** dialog box opens.
7. Select the field for cursor placement if the participant answers, **No**, from the list of variables. For this example, select **Antibiotics**.

8. In the **GOTO** dialog box, click **OK** to return to the **If** dialog box.



IF/THEN dialog box

Click **OK**. The Check Code editor displays the code. The example code appears:

```
Field Hospitalized
      After
            //add code here
            IF Hospitalized = (-) THEN
                  GOTO Antibiotics
            END-IF

      End-After
End-Field
```

IF/THEN Check Code Command

9. Click the **Verify** Check Code button from the Check Code Editor.

10. Click the **Save**, from the Check Code Editor.

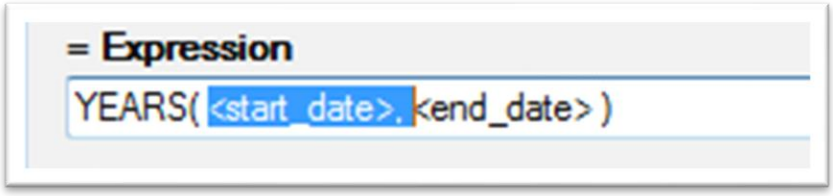
Using Functions with a Date Field

To program a mathematical function, use the **Program** Editor and **ASSIGN** command. For example, Check Code calculates the age of a participant based on date of birth and the form completion date (calibrated to the system date).

This example uses the **Date Of Birth (DOB)** and **Age** fields. (i.e.: from the **FoodHistory_NoCheckCode** form in the **E. coli project**). The example demonstrates the **ASSIGN** command and the function, **YEARS**.

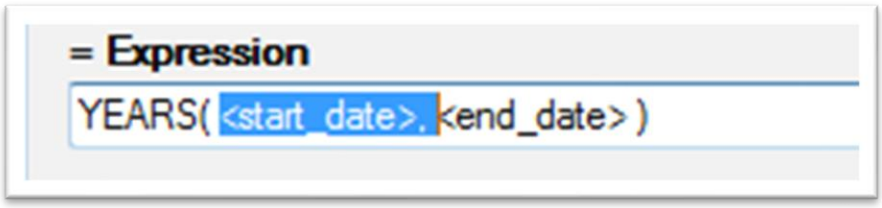
1. Click Check Code or select **Tools** > Check Code Editor. The Check Code Editor opens.
2. Expand the node from the **Choose Field Block for Action** section, where the **DOB** field is located (i.e. Page 1)
3. Expand the node for the date of birth field called **DOB**.
4. Double click on the **after** event.
5. The Check Code Editor displays a block of code for the **DOB** field.
6. Click **Assign**, from the **Add Command to Field Block** list box. The **Assign** dialog box opens.
7. Select the field where the calculated value should appear.
8. Select the **Age** field from the **Assign Variable** drop-down list.
9. Click on the functions button.
10. Select the **Date Functions** option.
11. Click on the **YEARS** function.

12. Double click on the **<start_date>** parameter. This highlights the command section.



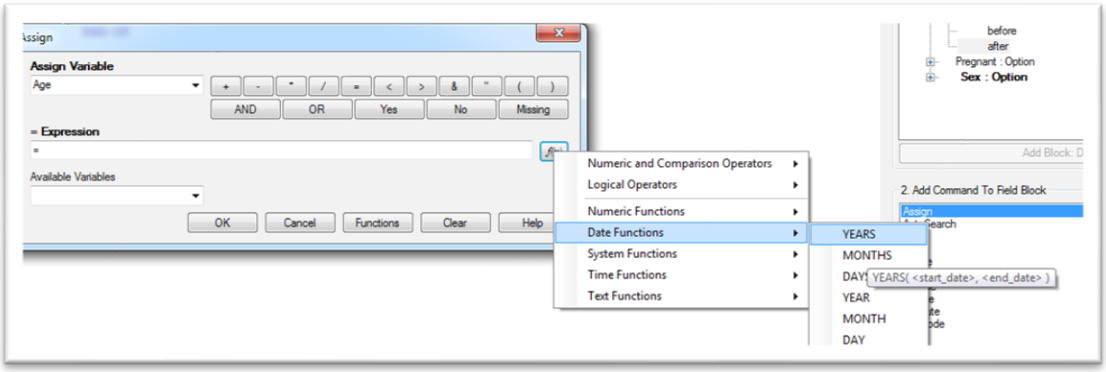
Start Date Parameter

1. Select **DOB** from the list of variables
2. Double click on the **<end_date>** parameter. This action highlights the command section.

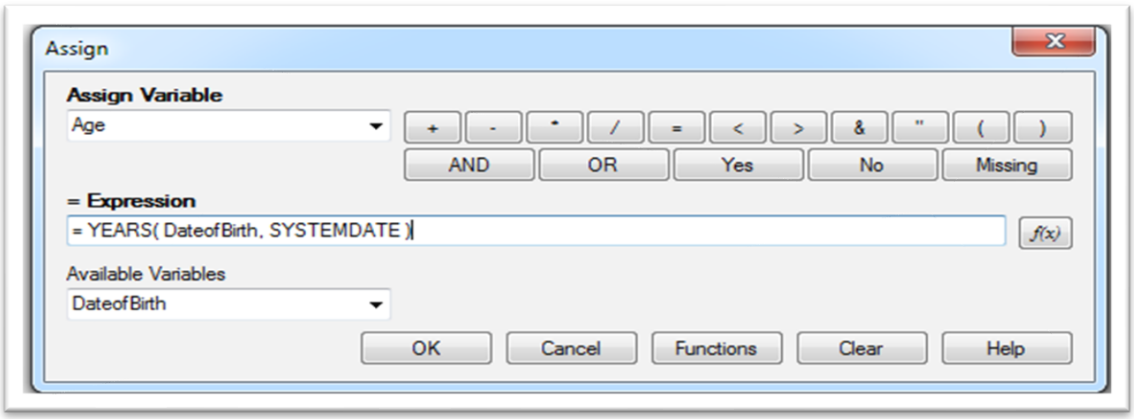


End Date Parameter

3. Click on the functions button that looks like 
4. Select the **System Functions** option.
5. Click the **SYSTEMDATE** function. Once completed, the inserted syntax appears in the dialog window.

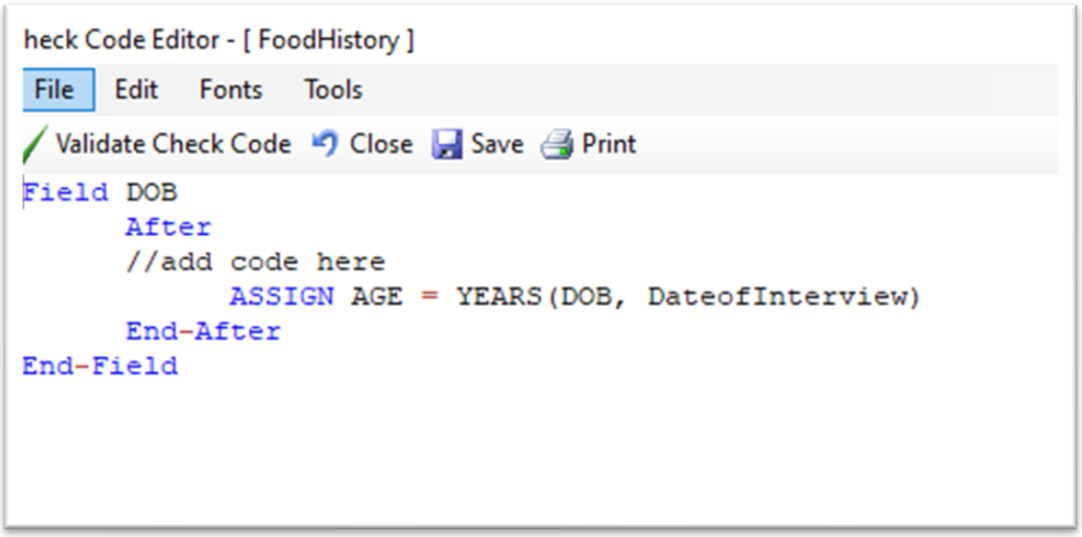


Date Function options



Assign dialog box

6. Click **OK**. Check Code appears in the Check Code Editor.
7. Click the **Verify** Check Code button in the Check Code Editor.
8. Click the **Save** button in the Check Code Editor.



Assign Check Code Command

When entering a date of birth into the form, the **Age** field populates.

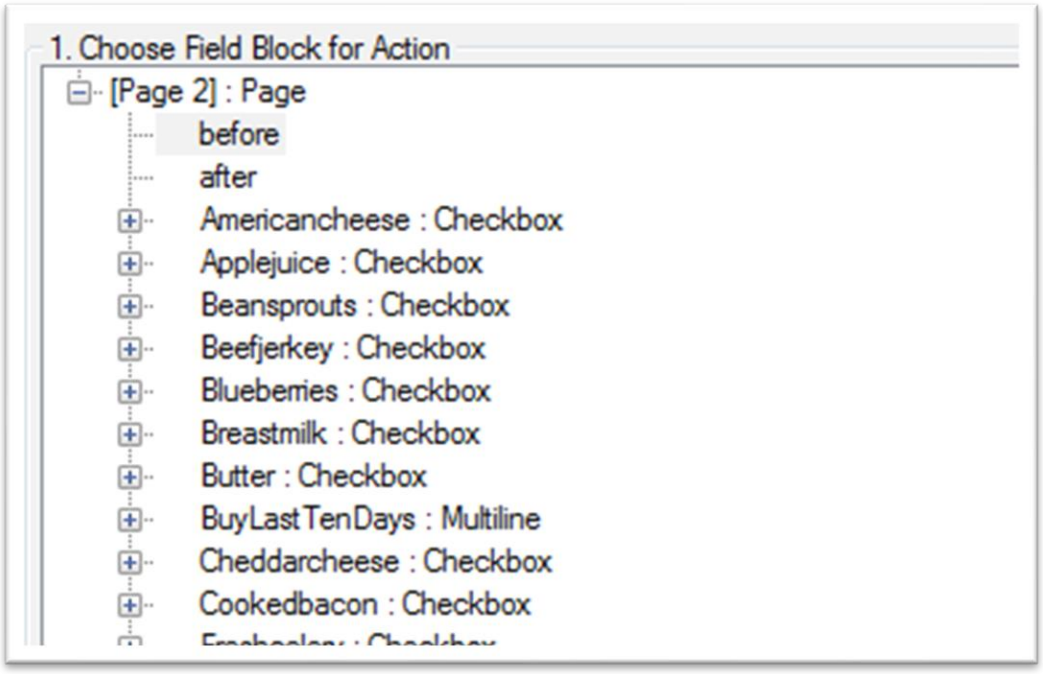
Assignment of Age in Enter Data

Interact with Users using the DIALOG Command

Interact with data entry personnel from within the program, using the **DIALOG** command. Dialogs display information, requests and receives input, and generates helpful lists, assisting survey respondents. In the following example, the **DIALOG** command creates a reminder that respondents must complete all fields on page two.

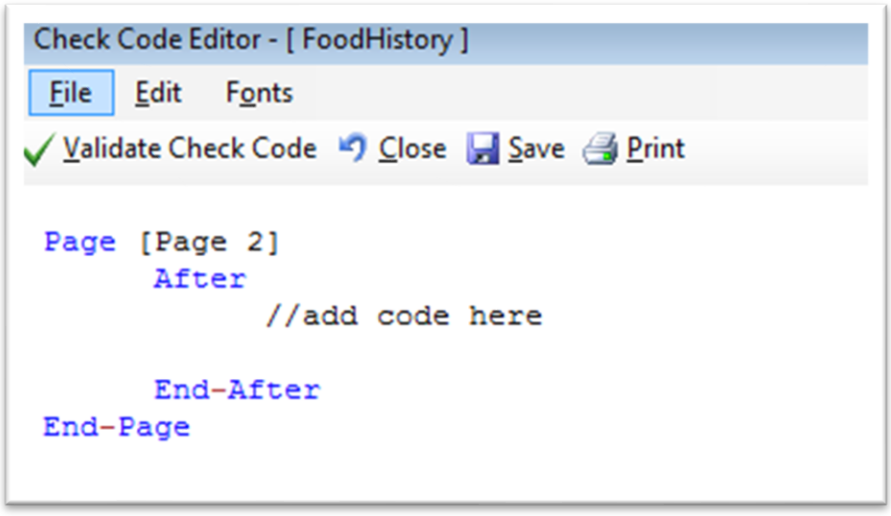
1. From Form Designer, Open the **FoodHistory_NoCheckCode** form in the EColi.prj.

2. Click Check Code or select **Tools > Check Code** Editor. The Check Code Editor opens.
3. Select **page 2** from the **Choose Field Block for Action** list box. The action occurs before the page loads.
4. Select **Before** from the Before or After section.
5. Click the **Add Block** button.



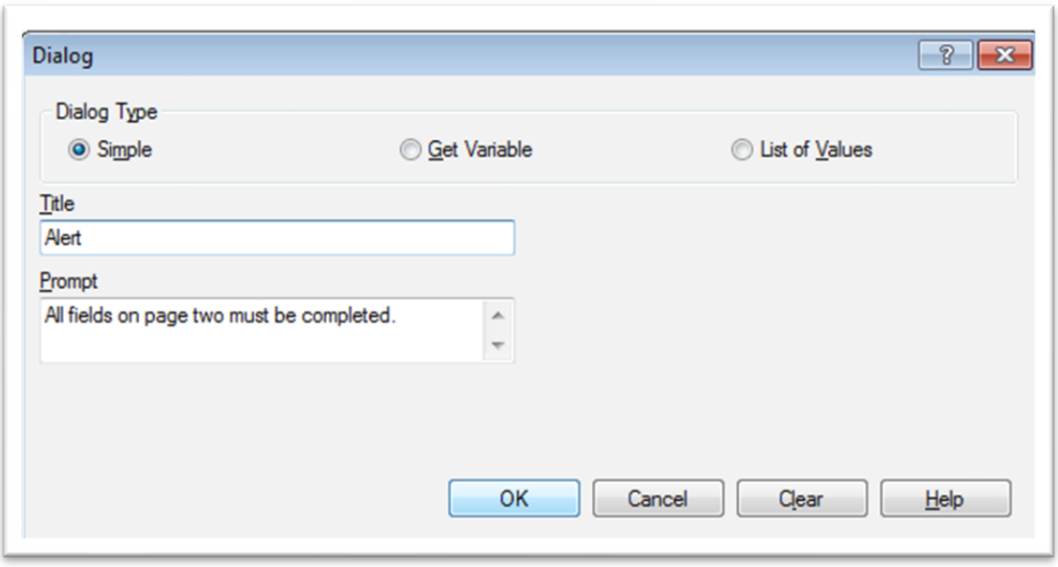
Before action

6. The code appears in the Check Code Editor.



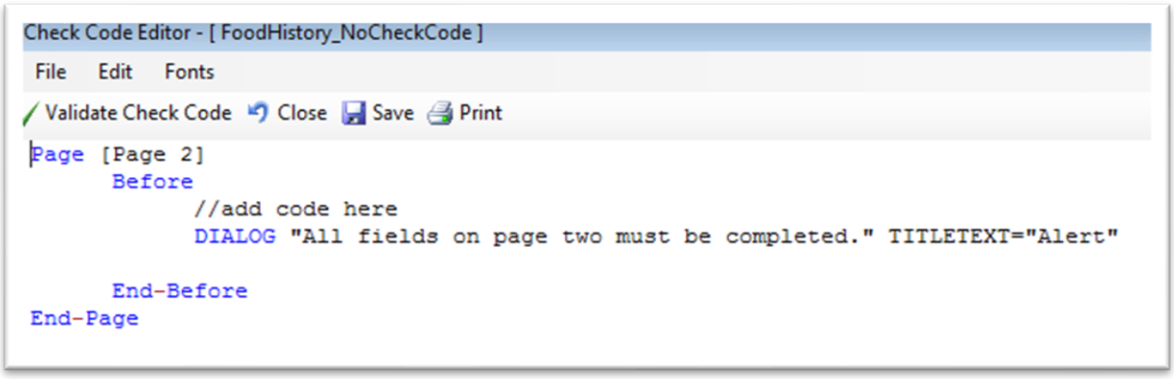
Dialog Block Code

7. From the **Add Command to Field Block** list box, select **Dialog**. The **Dialog** box opens.
8. Select **Simple** from the Dialog Type radio button.
9. In the Title field, type "Alert."
10. In the **Prompt** field, type: "All fields on page two must be completed."



Dialog commands box

1. Click **OK**. The code appears in the Check Code Editor.



Dialog Check Code Command

2. Click the **Verify** Check Code button from the Check Code Editor.
3. Click the **Save** button in the Check Code Editor.

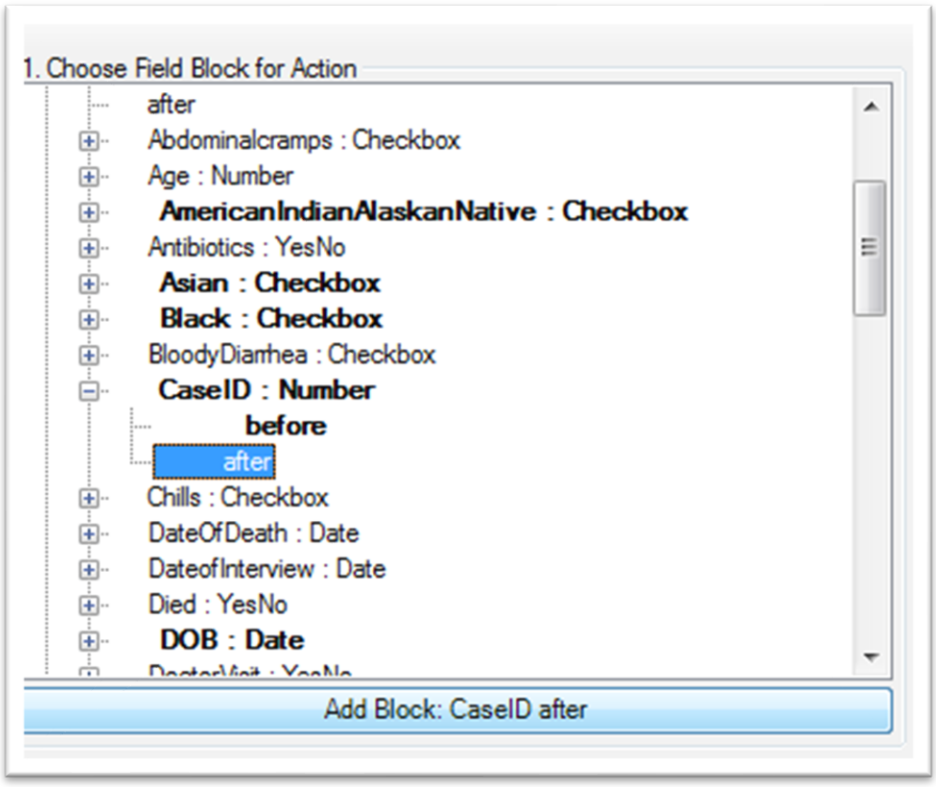
Searching for Records – Using the AUTOSEARCH Command

The **AUTOSEARCH** command searches for existing records matching entered values and notifies the user. You can edit the matching record or continue entering data in a new record. The **AUTOSEARCH** command detects and prevents duplicate records. View the example using **AUTOSEARCH** on the **CaseID** field.

1. From Form Designer, Open the FoodHistory_NoCheckCode form in the EColi.prj.
2. Click Check Code or select Tools > Check Code Editor. The Check Code Editor opens.
3. From the Choose Field Block for Action section, expand the node for page 1 and locate the CaseID field.
4. Expand the node for the CaseID field .

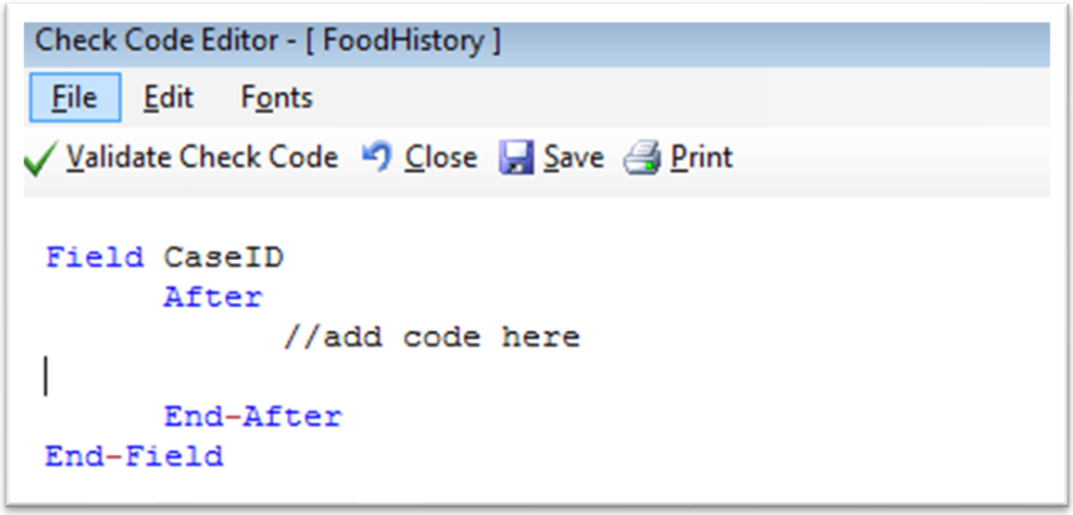
5. Double click on the after event.

6. Check Code Editor displays a block of code for the CaseID field.



AUTOSEARCH Block for Action

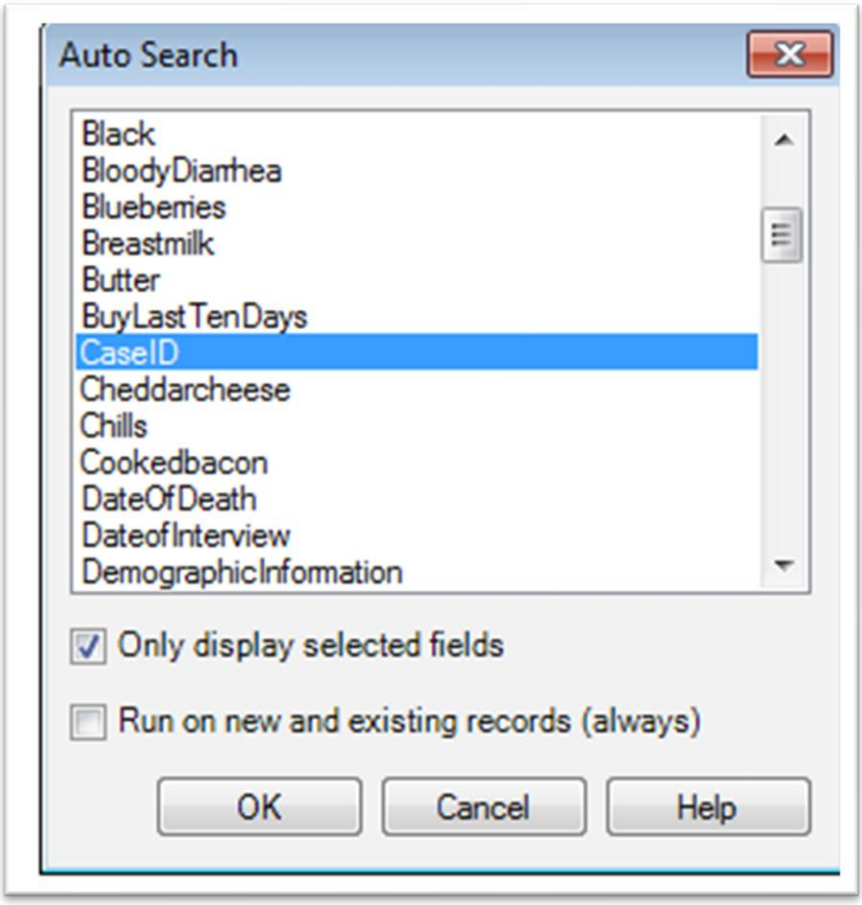
1. **Check Code Editor** displays the code.



AUTOSEARCH Block Code

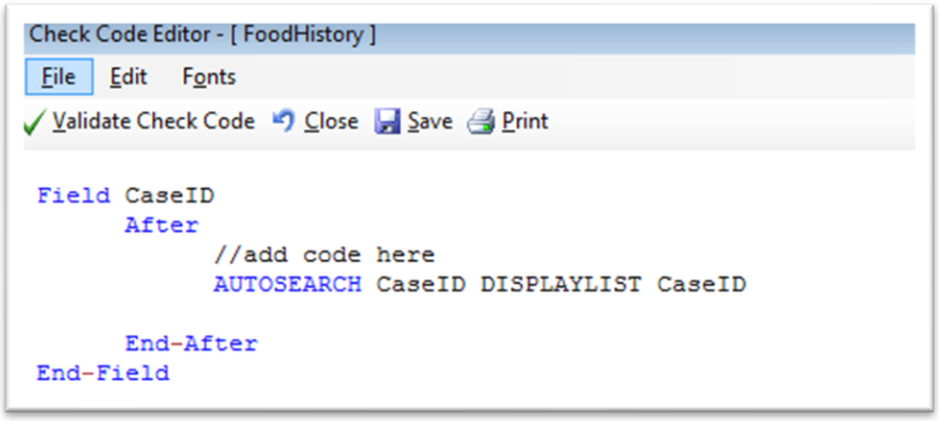
From the **Add Command to Field Block** list box, click **AUTOSEARCH**. The **AUTOSEARCH** window opens.

Select the search variable(s) during data entry. In this example, select **CaseID**.



AUTOSEARCH dialog box

Click **OK**. The code appears in the Check Code Editor window.



AUTOSEARCH Check Code Command

Follow the next series of actions to verify your syntax:

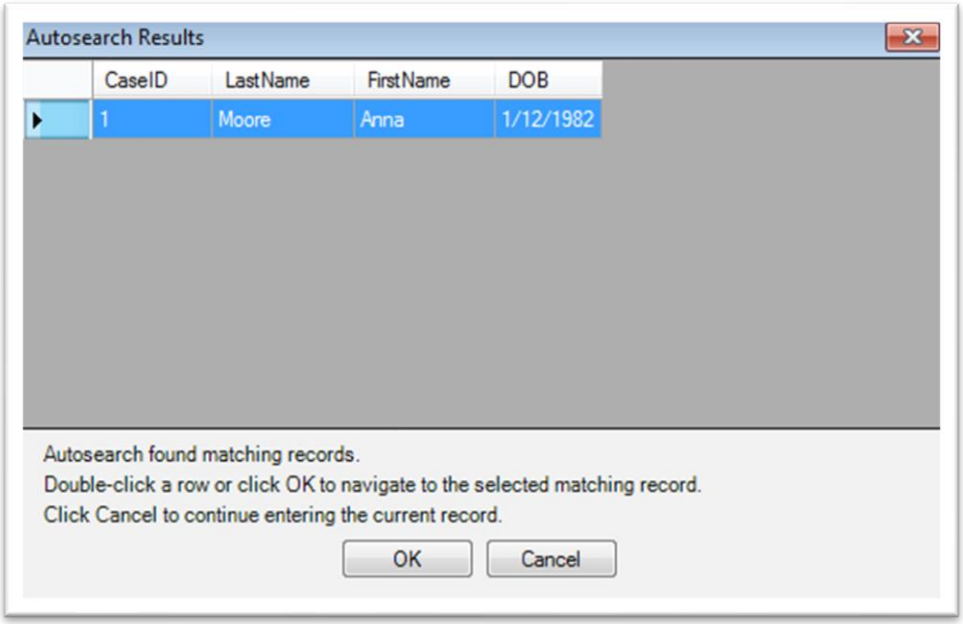
- Click **Verify** Check Code from the Check Code Editor.
- Click **Save** from the Check Code Editor.
- Click **Close** to return to the form.

The **AUTOSEARCH** dialog box opens with all the matching records when a duplicate record is entered from the **Enter Data** tool. View and clear the duplicate record by double-clicking the arrow next to the record. Otherwise, click **Cancel** to remain on the current record and accept the duplicate value. To display other matching record variables, add the variable names after the **DISPLAYLIST** parameter (i.e., Last Name, First Name and Date of Birth as shown below).

Field CaseID
    After
    //add code here

```
        AUTOSEARCH CaseId DISPLAYLIST CaseID LastName FirstName
DOB
    End-After
    End-Field
```

In the Check Code above, **AUTOSEARCH** will find matching records on the field **CaseID**. Detected matching records display the following fields on the grid: CaseID LastName, FirstName and DOB.



AUTOSEARCH Results box

Note: For more information on using **AUTOSEARCH**, please see the **AUTOSEARCH** topic in the Command reference section.

Copy Field Values from Main Forms to Related Forms

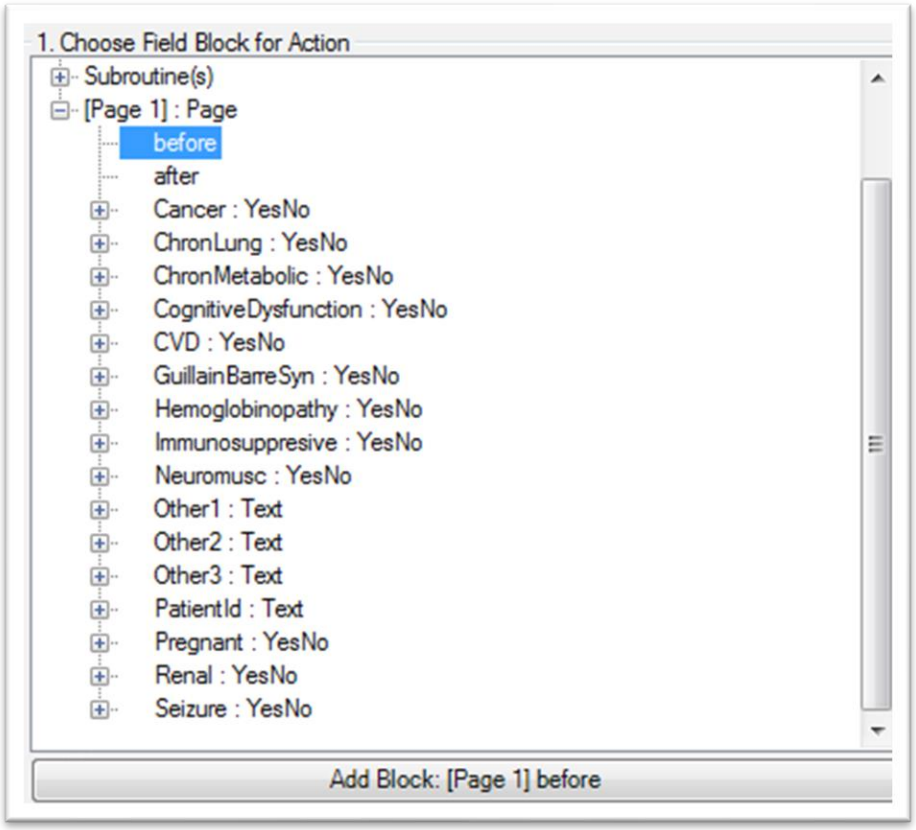
After building a relational database using Epi Info™ 7, best practices suggest transferring field values in the parent form (i.e., core demographics) to the child form (i.e., visits information). Accomplishing this requires existing Check Code from

field values in both parent and child forms. (i.e., Visible Case ID Number or Patient's Name must exist in parent and child forms).

The following instructions assume the parent and child forms already exist. Let's name the parent form **Surveillance** while the child form is called **Hepatitis**. A field in **Surveillance**, must be present in **Hepatitis**. Otherwise, users need to create it. Afterward, use Check Code in **Hepatitis** (i.e., **LastName** in parent form is transferred to **LastName** in child form). Let's call the parent form field name: **PatientId.** Let's copy **PatientID** values to a child form.

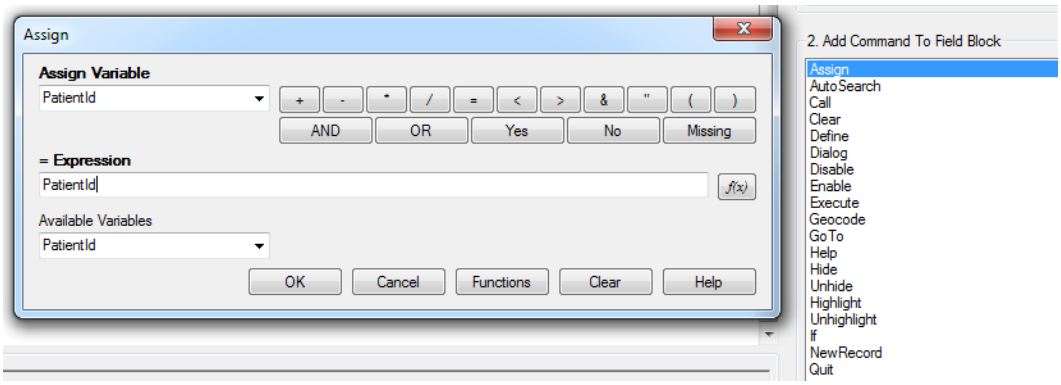
Open your project from **Form Designer** and click on the child form from the **Project Explorer** tree.

2. Create a new field. The new field must be the same field type as the field being copied from the parent form. For this example, use **PatientId**.
3. Select the **Read Only** option.
4. Click **OK**. The new field appears in the form. A value from the parent form will be assigned and displayed during data entry on the child form.
5. Click Check Code or select **Tools >** Check Code Editor. The Check Code Editor opens.
6. From the **Choose Field Block for Action** list box, select the page corresponding to the location of the **PatientId** field. Let's assume this field is on page 1. For the page, select the **<before>** from the before or after section.
7. Click the **Add Block** button.



Choose Field Block for Action

1. From the **Add Command to Field Block** list box, click **Assign**. The **Assign** dialog box appears.
2. From the **Available Variables** drop-down list, select the new variable, **PatientId**.
3. In the = **Expression** area, type the field name from the parent form. In this case, use **PatientId**. Add a prefix followed by a period for the field name, in **Assign** expression (**parentformname.**). The prefix is the parent form name. Modify as described after the command writes to the **Program** Editor:



Copy Value- Assign dialog box

4. Click **OK**.

Check Code Editor displays the code. If parent and child forms have the same field name, distinguish the field name during the copy operation. Always prefix the field name with the parent form name.

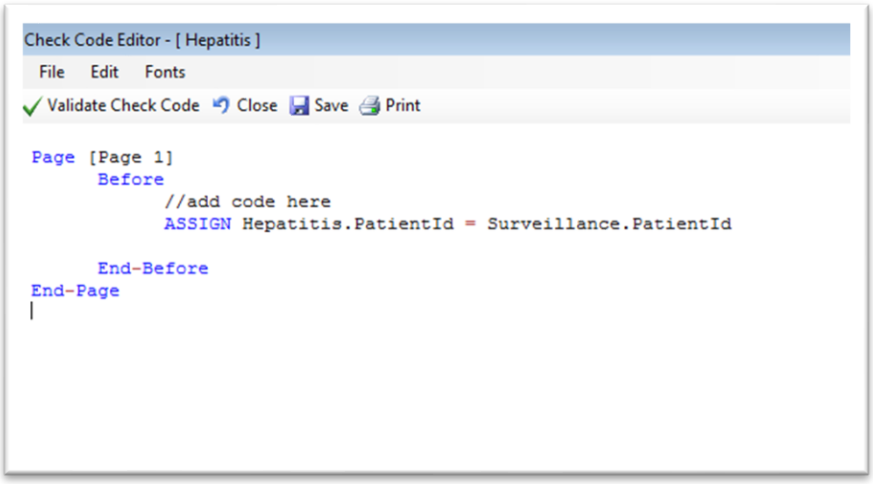
Steps:

- Determine if the child field and the parent field have the same name
- Add a prefix to the field name.
- Add a period in Assign expression.

Note: Syntax: **<parentformname.childfieldname>**

View the example in the image below.

Hepatitis is the name of the child form; **Surveillance** is the name of the parent form. If parent and child forms do not have the same field name, only use the field name in the syntax. Follow the syntax example in the image below.



Copy Value Check Code Command

Click the **Validate** Check Code and correct any issues. Then click **Save**.

Concatenate Fields

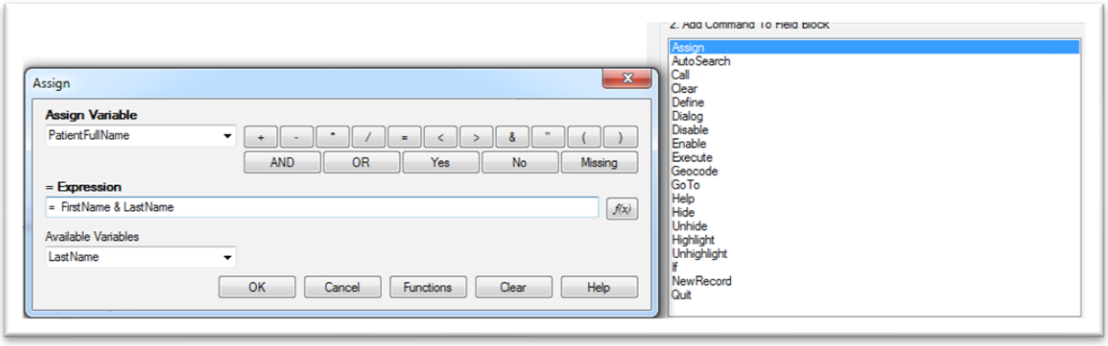
Check Code syntax will work for new records with concatenating fields. Check Code syntax won't work retroactively. Your code won't execute on old records with concatenating fields. Instead, use concatenation commands for previous records. Refer to the **Classic Analysis** section of the manual.

Concatenate Fields with the Ampersand '&' Operator

This example illustrates how to join data from two fields and assign it into a third field using the **'&'** operator. The example shows how **PatientFullName** is assigned to the concatenation of **FirstName** and **LastName**.

1. Open your form in **Form Designer**.
2. Click Check Code, the Check Code Editor opens.
3. Select **LastName** from the **Choose Field Block for Action** list box.

4. Select **After** from the **<Before>** and **After** Section.
5. Click the **Add Block** button.
6. Go to the **Add Command to Field Block** list box, click **Assign**. The **Assign** dialog box opens.
7. From the **Assign Variable** drop-down list, select the field for the concatenated value.
8. Create the **=** expression using the **&** operator. In this example, **Assign PatientFullName = FirstName & LastName**.



Concatenate - Assign dialog box

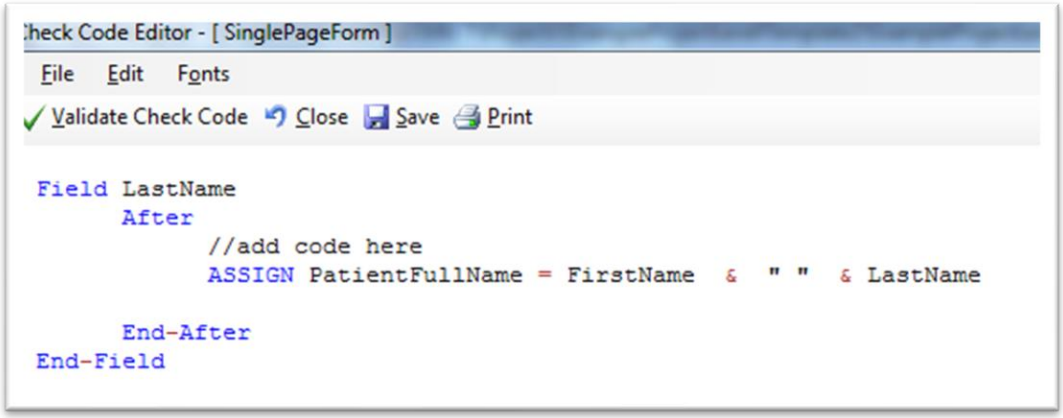
9. Click **OK**. Check Code appears in the Check Code Editor.

```
Field LastName
      After
            //add code here
            ASSIGN PatientFullName = FirstName & LastName

      End-After
End-Field
```

Concatenate No Space Check Code Command

10. Click the **Validate** Check Code button and correct any issues and click **Save**.

For example, using the **Enter Data** tool, add Carl for **FirstName** and Gao for **LastName**. The result for **PatientFullName** would be CarlGao. Add a space between CarlGao by changing the **ASSIGN** statement. Add a blank space in quotes between the first and last names.



Concatenate Include Space Check Code Command

Concatenate Fields with the Substring Function

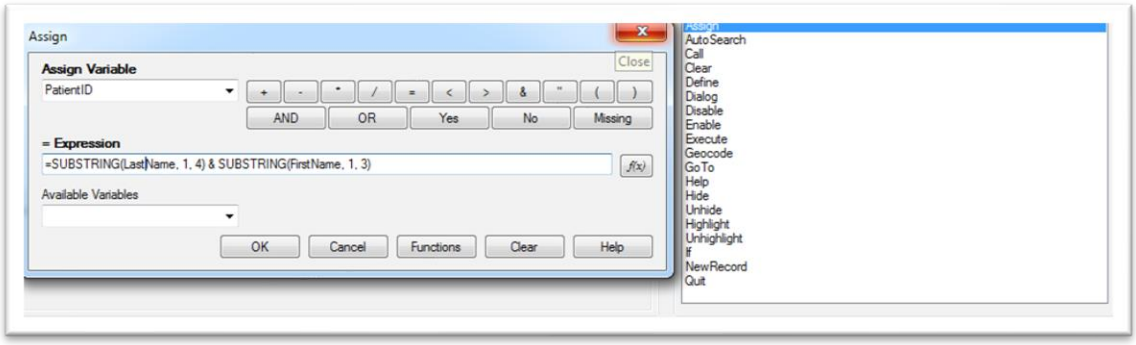
This example illustrates how to join segments of two variables to create a unique text ID. In this example, create a **Patient ID** made up of segments of the patient's last and first name. Use the ampersand (**&**) operator and join the two segments together.

Enter Form - Name

Steps:

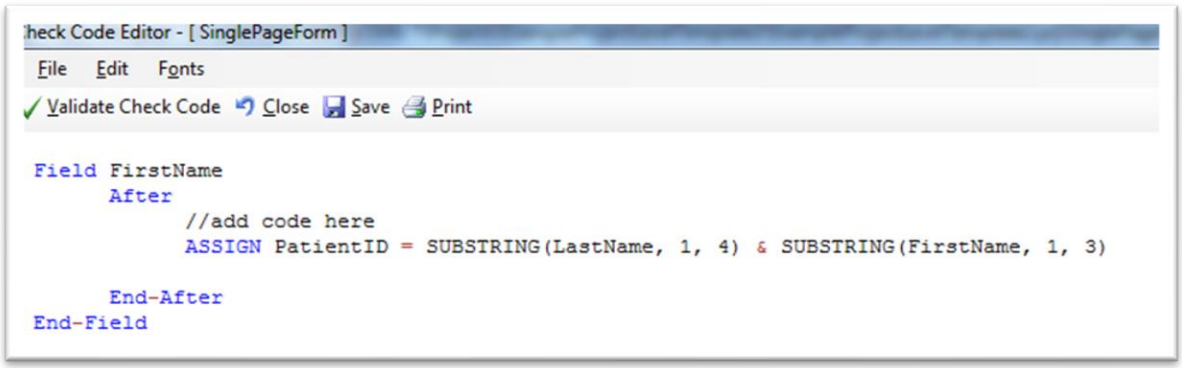
1. Click Check Code from the **Form Designer**. The Check Code Editor opens.
2. Select **FirstName**, from the **Choose Field Block for Action** list box.
3. Select **After** from the **Before** and **After** Section.
4. Click the Add Block button.
5. Click **Assign**, from **Add Command to Field Block** list box The **Assign** dialog box opens.
6. Select the field to contain the concatenated value from the **Assign Variable** drop-down list. In this example, select **PatientID**.
7. Create the **=** expression using the **SUBSTRING** syntax.
    - **SUBSTRING(<variable>, position #, #characters)**, whereas the expression **<variable>** is the field or variable.
    - **position #** is the position of the first character extracted from the variable
    - **#characters** is the number of characters to extract

In this example, the **PatientID** variable contains a combination of the first position and four characters of the last name plus the first position and three characters of the first name.



Concatenate with Substring Assign dialog box

8. Click **OK**. Check Code Editor window displays the code.

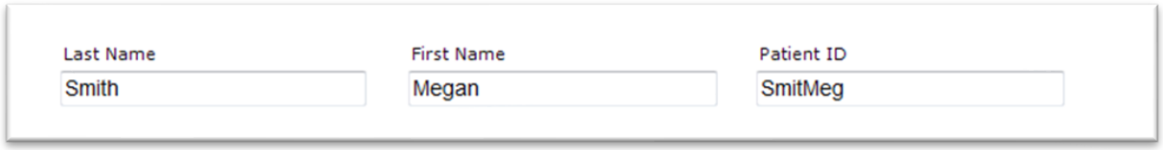


Concatenate with Substring Check Code Command

9. Click **Validate** Check Code button and correct any issues, and then click **Save**.

The example functions as described below.

- Last Name: Smith
- First Name: Megan
- Patient ID: SmitMeg
- The **Patient ID** field being calculated is read only.

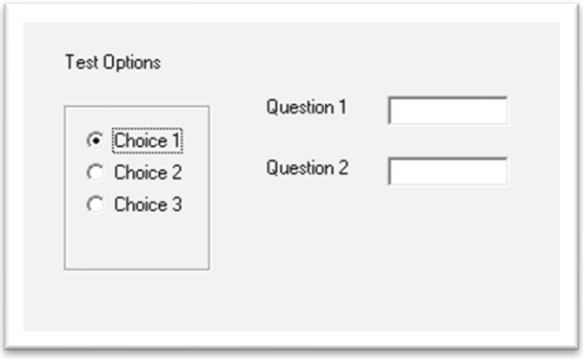


Concatenate with Substring Enter Form

Create Check Code for Option Box Fields

Add Check Code to option box fields. Add Code to any line/choice in the option boxes. Use the Check Code Editor and create complex Check Code for option box fields.

The Check Code **GOTO** command works in the following scenario. If the answer to **Test Options** is **Choice 1**, the cursor will jump to **Question 2**. If the answer to **Test Options** is **Choice 2** or **Choice 3**, the cursor will jump to **Question 1**. Check the tab order before creating the Check Code to ensure accuracy.

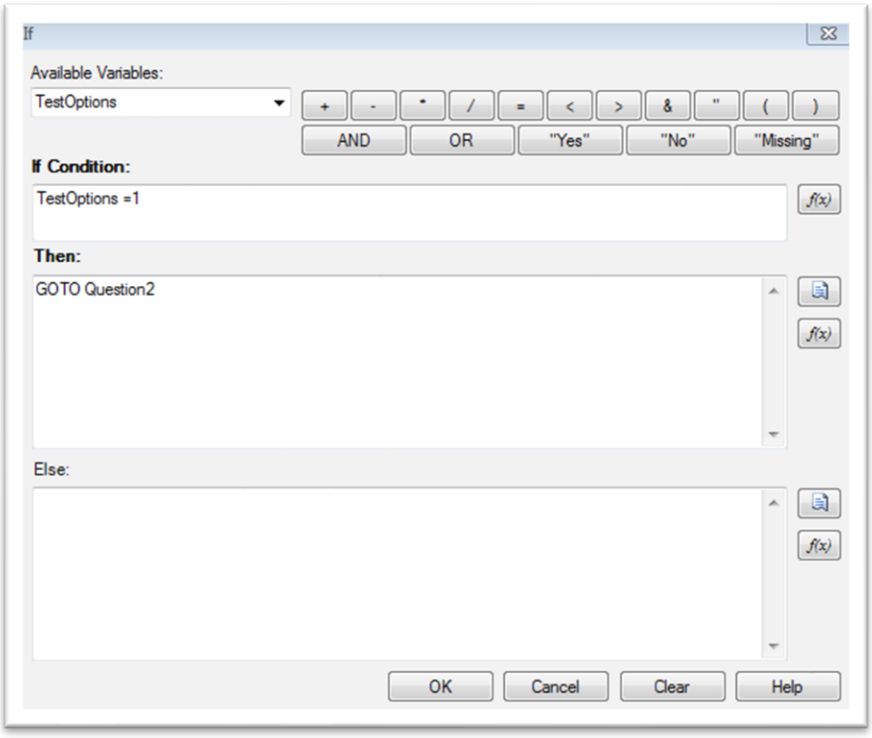


Text Options box

1. Create an **Option Box** in a form. Note the name of the variable.
    - The variable is: **TestOptions**.
    - Each text line represents a choice in the form represents a numeric position in the Check Code Editor.

- For example, there are three potential lines in the variable **TestOptions**. In the Check Code Editor, the choices are numeric, Choice 1 = position 0, Choice 2 = position 1, and Choice 3 = position 2.

2. Open the Check Code Editor.
3. Select **TestOptions**, From the **Choose Field Block for Action** list box.
4. Select **Click** from the **After** or **Click** Section.
5. Click the **Add Block** button.
6. Click If from the **Add Command to Field Block** list. The **IF** dialog box opens.
7. Type "Test Options = 1" from the **If Condition** field,
8. Remember that the number 1 represents a text value, Choice 2.
9. Click the **Code Snippet** button in the **Then** section. A list of commands appears.
10. Select **GOTO**, from the command list. The **GOTO** dialog box opens.
11. Select **Question2**.
12. Click **OK**.



If dialog box

13. Click **Validate** Check Code and correct any issues, then click **Save**.

Delete a Line of Code from the Check Code Editor

Steps:

1. Highlight the line(s) of code/text that you desire to delete.
2. Tap the Delete key on your keyboard.
3. Click Save from the Check Code Editor toolbar.

Note: Confirm all deletions before making them. Deletions are permanent.

Additional Check Code Commands

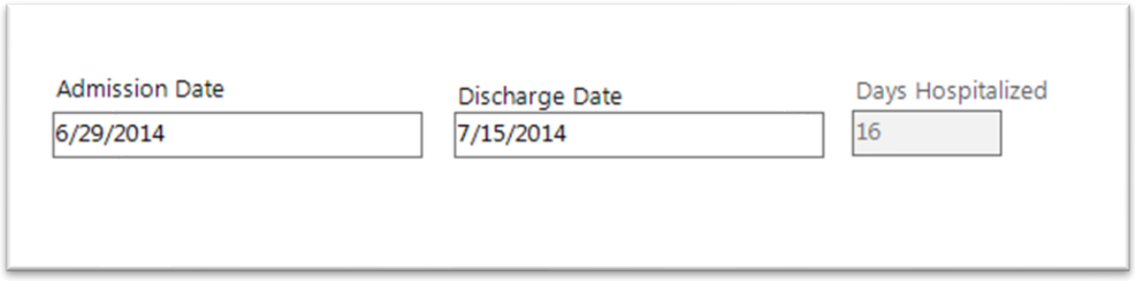
CALL

This command redirects to another command block in Check Code and returns after execution. Calls and subroutines work together. Subroutines act as a Check Code common unit, usually dependent on two variables. The benefit of using subroutines is that it allows maintenance of Check Code in one common location. Here is an example of a subroutine:

```
Sub CalculateDays
      ASSIGN DaysHosp = DAYS(AdmissionDate,DischargeDate)
End-Sub

Field AdmissionDate
      After
            CALL CalculateDays
      End-After
End-Field

Field DischargeDate
      After
            CALL CalculateDays
      End-After
End-Field
```

Check Code syntax for Subroutines

In the example above, determining the number of hospitalization days (**DaysHosp**) uses the **After** event. Place the **After** event after the **AdmissionDate** and **DischargeDate.** These expressions calculate the total number of days (**CALL** CalculateDays). Use Subroutines to update and maintain Check Code in one location. The **CALL** command executes a block of Check Code in multiple fields.

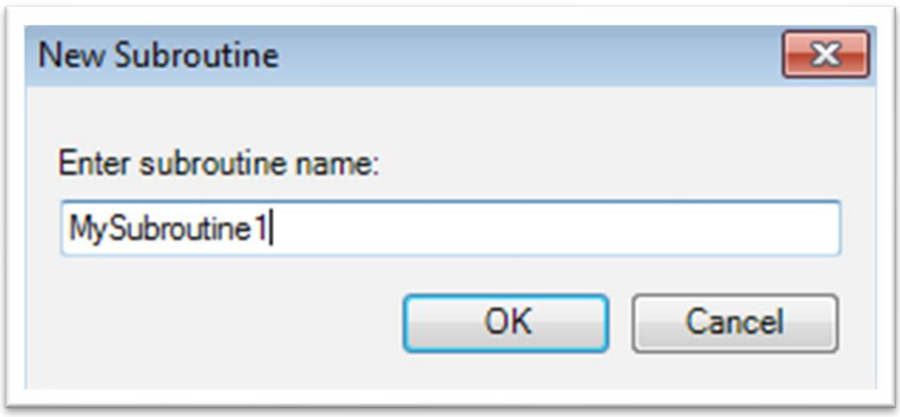


Calculation of Days Hospitalized

Create a Subroutine

Steps:

1. Open your form from **Form Designer**,
2. Click Check Code. The Check Code Editor opens.
3. Expand the subroutine item from the **Choose Field Block for Action** list
4. Double click on the **Add new item**. A **New Subroutine** window opens.
5. Assign a name to your subroutine (i.e., **MySubroutine1**) and click **OK**.



Enter New Subroutine box

```
Sub MySubroutine1
      //add code here


End-Sub
```

MySubroutine1 code block

A new block of code for the subroutine called **MySubroutine1** displays in the Check Code Editor. Determine the Check Code commands for the subroutine. Verify your code and save your work. The user will need to place the CALL command with the name of the subroutine in each desired field in order to execute the Check Code captured in the subroutine.

CLEAR

If a field entry is incorrect, **CLEAR** sets the field with a null value as if it were blank. It's useful when the entry in the field contains an error to use the **CLEAR** command with the **GOTO** command. Working in tandem, the commands return the cursor back into the field so the participant may try again. Below, **CLEAR** removes data from the **DateofInterview** field after flagging the user with a message, "Date of interview is greater than today's date." The entry is cleared, and the user can try again.

```
Field DateofInterview
      After
            //add code here
            IF DateofInterview > SYSTEMDATE THEN
                  DIALOG "Date of Interview is greater than todays' date.  Please verify." TITLETEXT="Alert"
                  GOTO DateofInterview
                  CLEAR DateofInterview
            END-IF
      End-After
End-Field
```

Check Code syntax for CLEAR command

DEFINE

This command creates a new variable. Check Code saves all user-defined variables in the **DefineVariables** section.

The proper syntax is:

- **DEFINE** `<variable name> {<scope>} {<field type indicator>}`
- **`<variable name>`** represents the name of the new variable. **`<variable>`** cannot be a reserved word. For a list of reserved words, see the List of Reserved Words Section of the User's Manual.
- **`<scope>`** is optional and is the level of visibility and availability of the new variable. This parameter must be a reserved word: STANDARD, GLOBAL, or PERMANENT. If omitted, the code assigns STANDARD as the parameter and omits, a **<field type indicator>**.
- **`<field type indicator>`** is the data type of the new variable and must be one of the following reserved words: NUMERIC, TEXTINPUT, YN, DATEFORMAT, TIMEFORMAT and DLLOBJECT.

```
DefineVariables
            DEFINE MYVARIABLE1 TEXTINPUT
            DEFINE MYVARIABLE2 GLOBAL NUMERIC
            DEFINE MYVARIABLE3 PERMANENT DATEFORMAT
End-DefineVariables
```

Check Code syntax for DEFINE command

Warning: If omitted, the data type infers the variable type of the first assigned value. Thereafter, the variable type is unchangeable, and results in an error.

Below is a description of the **SCOPE** parameter. **SCOPE** is an optional parameter used with the **DEFINE** command.

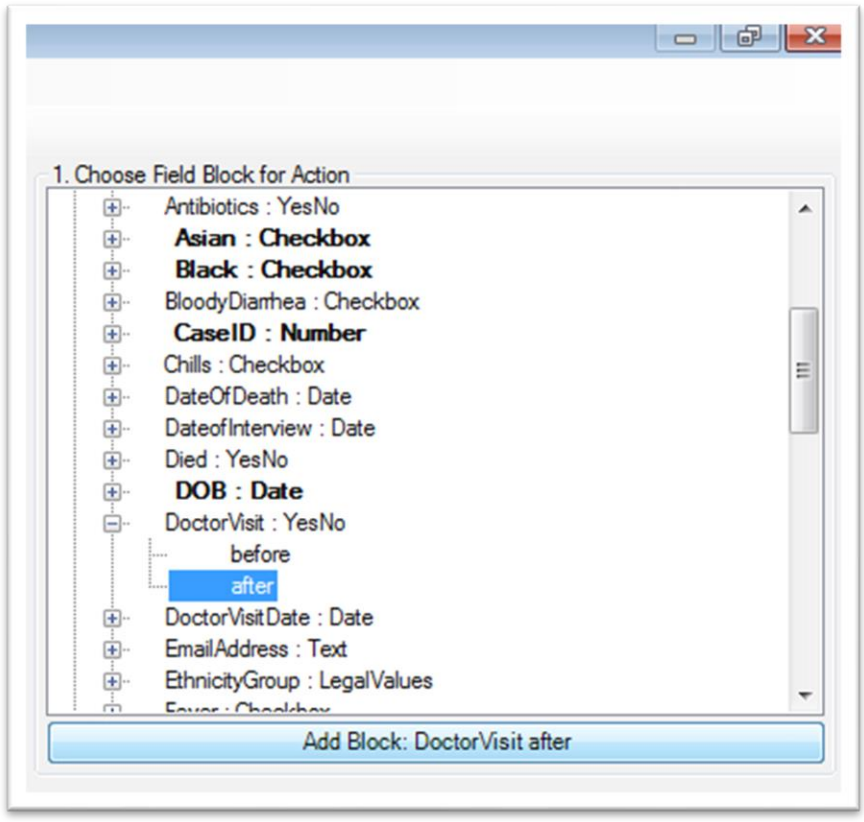
- **STANDARD variables** keep their values only within the current record. Values reset when loading new records. Standard variables work like temporary variables, behaving like other fields in the database.
- **GLOBAL variables** keep their values across related forms, even after opening a new form. Global variables persist for the duration of program execution. Global variables end when closing the **Enter Data** tool.
- **PERMANENT variables** keep any assigned values. Values do not persist for undefined variables or, if altered by another assignment. Epi Info™ modules (Enter, Classic Analysis, etc.), sharing permanent variables persists even if the computer shuts down.

Note: Find Permanent variables in the **EpiInfo.Config.xml** file located in the \Epi Info 7 > Configuration directory.

DISABLE

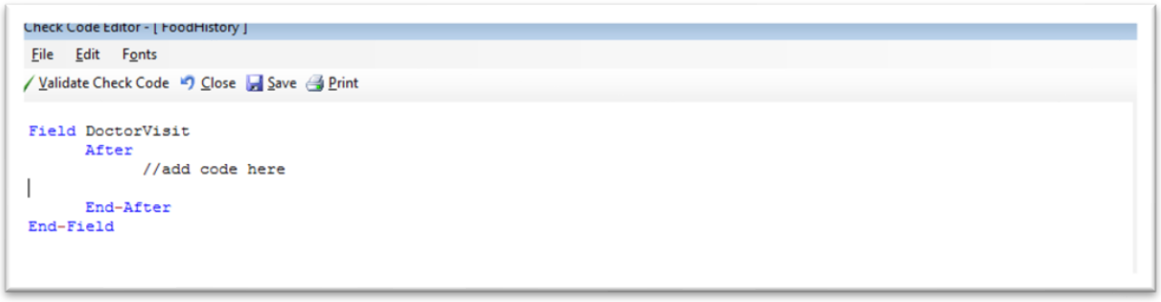
If a field isn't required for data collection, use the **DISABLE** Check Code command. Use If, Then, and Else conditionals Check Code commands to **DISABLE** a field. In this example, the field **DoctorVisitDate** will be disabled if the response to the **DoctorVisit** is No.

1. Open the FoodHistory_NoCheckCode form in the EColi.prj.
2. Click Check Code. The Check Code Editor opens.
3. Select the DoctorVisit from the Choose Field Block for Action list box.
4. Select after from the before or after Section.
5. Click the Add Block button.



Add Block: DoctorVisit after

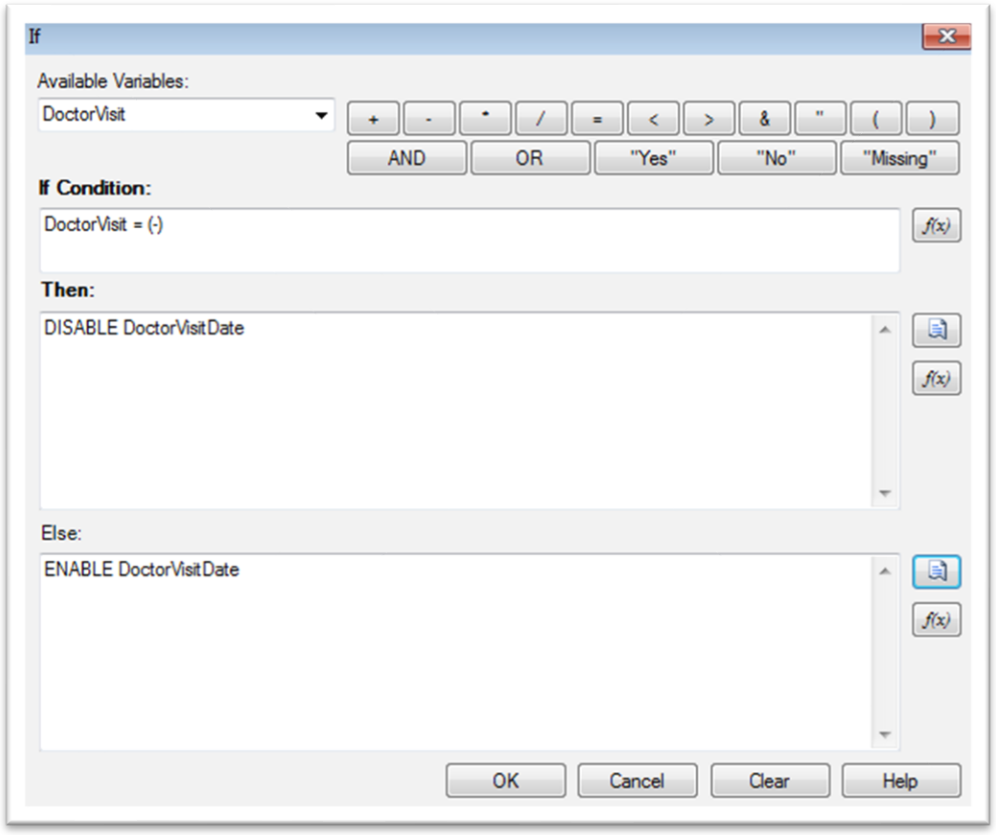
6. The code block appears in the Check Code Editor.



Disable Block Command

1. Click **IF** from the **Add Command to Field Block** list box. The **IF** dialog box opens.
2. From the **Available Variables** drop-down list, select the **field** to contain the action. For this example, select **DoctorVisit**. The selected variable appears in the **If Condition** field.
3. From the operators list, click **=**.
4. From the operators list, click **No**. **If Condition** field reads **DoctorVisit=(-)**.
5. Click the **Code Snippet** button in the **Then** section. A list of available commands appears.
6. From the command list, select **DISABLE**. The **DISABLE** dialog box opens.
7. Select to disable a field if the participant answers, "**No.**" Select **DoctorVisitDate** from the list of variables
8. Click **OK** to return to the **IF** dialog box.
9. Click the **Code Snippet** button in under the **Then** section. A list of available commands appears.
10. Select **Enable** From the command list. The **Enable** dialog box opens.

11. Select the field to enable based on a **"Yes"** answer from the list of variables. For this example, select **DoctorVisitDate**.
12. Click **OK** to in the **Enable** dialog box to return to the **If** dialog box.



DISABLE - Then dialog box

13. Click **OK**. The code appears in the Check Code Editor. View the code in the image below.

```
Field DoctorVisit
      After
            //add code here
            IF DoctorVisit = (-) THEN
                  DISABLE DoctorVisitDate
            ELSE
                  ENABLE DoctorVisitDate
            END-IF
      End-After
End-Field
```

DISABLE Check Code Command

Click:

- **Verify** Check Code**.**
- **Save.**
- **Close** to return to the form**.**

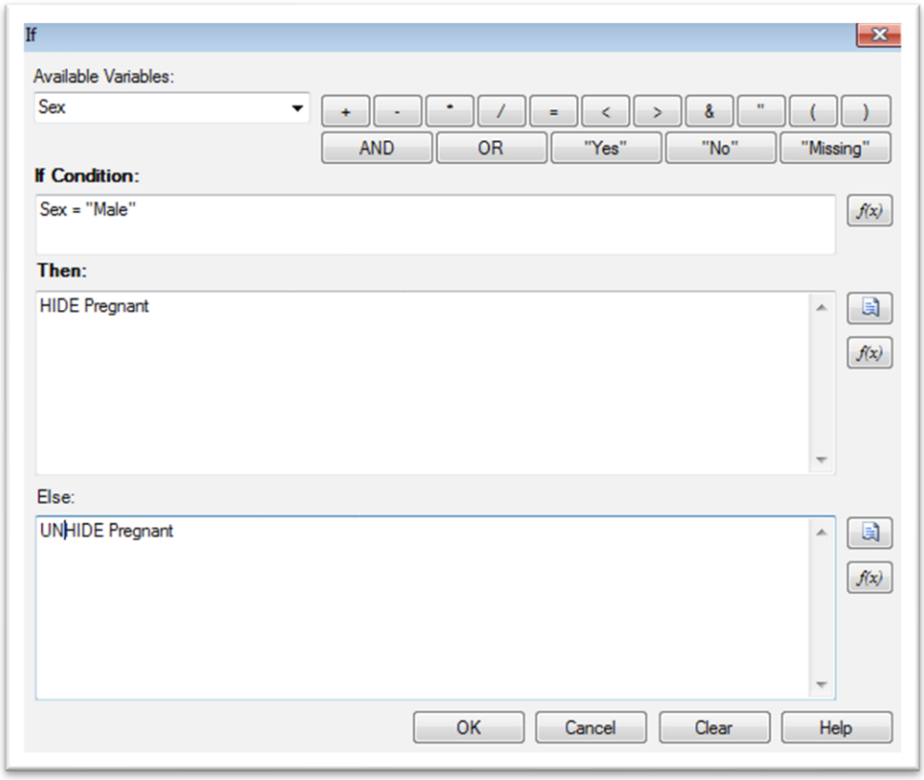
Note: Test DISABLE by opening the form in the Enter Data tool. Entering "No" for the DoctorVisit field will disable the DoctorVisitDate field.

HIDE/UNHIDE

Form designers can hide a field by using the **HIDE** command. **HIDE** works for removing fields that are non-applicable. View the following example. HIDE removes the Pregnant option box. The response for sex triggers the **HIDE** command, removing the *Pregnant* option box, following a respondent answer to their sex.

1. Click Check Code. The Check Code Editor opens.
2. Select **Sex** from the **Choose Field Block for Action** list box.
3. Select **after** from the **before** or **after** Section.
4. Click the **Add Block** button. The code runs after data entry.

5. Check Code Editor displays the code.
6. Click **If**, from the **Add Command to Field Block** list box. The **If** dialog box opens.
7. Select the field from the **Available Variables** drop-down list. Select **Sex** for this example. The **If Condition** field shows the selected variable.
8. Click **=** from the Operators list.
9. Type **Sex = "Male"** from the **If Condition field**. Remember that **Sex** is a text field, so always the value in quotes.
10. Click the **Code Snippet** button under the **Then** section. A list of commands appears.
11. Select **HIDE**, From the command list. The **HIDE** dialog box opens.
12. Select **Pregnant** and click **OK**.



Then - Else dialog box

13. Click the **Code Snippet** button in the **Else** section. A list of commands appears.
14. Select **UNHIDE** from the command list, The **UNHIDE** dialog box opens.
15. Select the field to unhide, if the participant answers, **"yes."** Select **Pregnant**.
16. Click **OK** in the **Unhide** dialog box, to return to the **If** dialog box.
17. Click **OK**, and then again, **OK**. Check Code Editor displays the code.

```
Field Sex
      After
            //add code here
            IF Sex = "Male" THEN
                  HIDE Pregnant
            ELSE
                  UNHIDE Pregnant
            END-IF
      End-After
End-Field
```

Hide Check Code Command

14. Click the **Save** button.

15. Click **Close** to return to the form.

Note: The Unhide command shows any hidden fields. You can select this command in the Add Command to Field Block section of Check Code Editor. Use a CLEAR command with the HIDE command to set to null, any information previously entered in the Pregnant field.

ENABLE/DISABLE

These commands can work in conjunction. The **DISABLE** command disallows data entry into a field while the **ENABLE** command allows data entry into a previously disabled field.

```
Field ILL
      After
            //add code here
            IF ILL = (-) THEN
                  DISABLE Symptoms
            ELSE
                  ENABLE Symptoms
            END-IF

      End-After
End-Field
```

Check Code syntax for ENABLE/DISABLE command

EXECUTE

Executes a Windows program — either explicitly named in the command or designated within the Windows registry. (i.e., A file with extension listed in a Windows registry). Using the **EXECUTE** command, Windows launches the default program(s) listed in its registry. The **EXECUTE** command accepts a series of paths, separated by semicolons like the example below:

EXECUTE c:\Users\MyPC\myfile.xls;d:\myfile.xls

EXECUTE attempts to launch files in succession. In Check Code, place the **EXECUTE** command in any command block. You can use it with a command button. Execute Check Code syntax when you click on the command button.

EXECUTE WAITFOREXIT **"<filename>"**
EXECUTE NOWAITFOREXIT **"<filename>"**

- The **<filename>** represents the path and program name for .exe files and .com files (DOS binary executables).
- The **<command-line parameters>** represent any additional, program acceptable command-line arguments.
- When using the **WAITFOREXIT** command (modal), the command runs and **Enter** pauses until the command has finished. Then, **Enter** runs subsequent commands. When using the **NOWAITFOREXIT** command (non-modal), **Enter** continues running any subsequent commands without waiting for previous commands to finish. **EXECUTE** (modal), writes permanent variables before the command runs, and reloads them again, afterward.

In the example below, the CDC website page opens when the users click on a command button called **OpenCDCWebsite**. A PDF file opens when the users click on a command button called **OpenPDFdocument**.

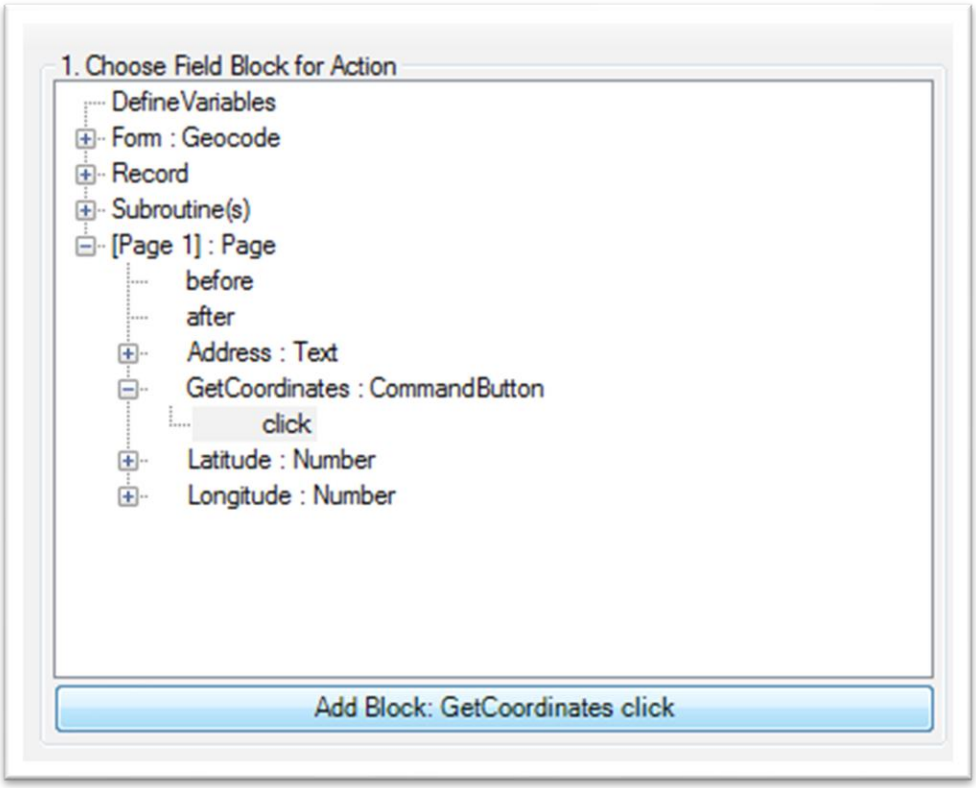
```
Field OpenCDCWebsite
      Click
            EXECUTE WAITFOREXIT "www.cdc.gov"
      End-Click
End-Field

Field OpenPDFdocument
      Click
            EXECUTE WAITFOREXIT "C:\Users\MyPC\Desktop\DownloadingEpiInfo7.pdf"
      End-Click
End-Field
```

Check Code example of EXECUTE command syntax

GEOCODE

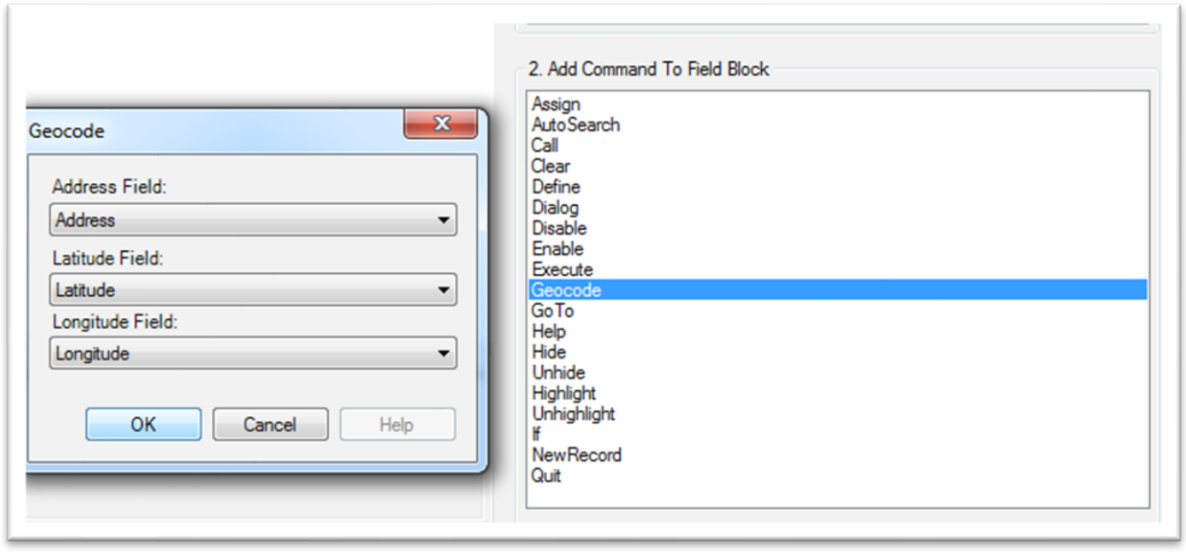
The Geocode command uses data entered in a text field which corresponds to an address. GEOCODE retrieves and populates Latitude and Longitude coordinates into the form. When using Desktop, this command will require an Internet connection.

1. Open the **FoodHistory_NoCheckCode** form, from **Form Designer**. Go to the form in the **EColi.prj**.
2. Click Check Code. The Check Code Editor opens.
3. Select **GetCoordinates** from the **Choose Field Block for Action** list box.
4. Select **Click**.
5. Click the **Add Block** button. This creates code to run after the **GetCoordinates** command button is clicked.



Geocode Block for Action

1. The code appears in the Check Code Editor.
2. Select **Geocode** from the **Add Command to Field Block** list box. The **Geocode** dialog box opens.
3. Select **Address** from the **Address Field** drop-down list.
4. Select **Latitude** from the **Latitude Field** drop-down list,
5. Select **Longitude** from the **Longitude Field** drop-down list.



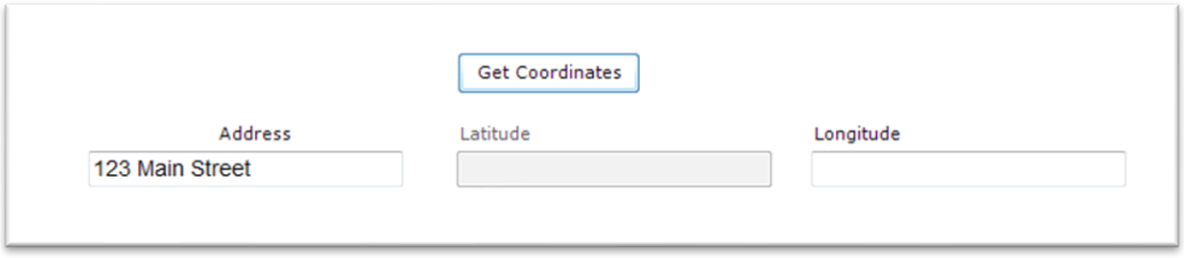
Geocode dialog box

1. Click **OK**.
2. Click **OK**. Check Code Editor displays the code. Click **Save**, then click **Close**.

```
Field GetCoordinates
      Click
            //add code here
            GEOCODE Address, Latitude, Longitude

      End-Click
End-Field
```

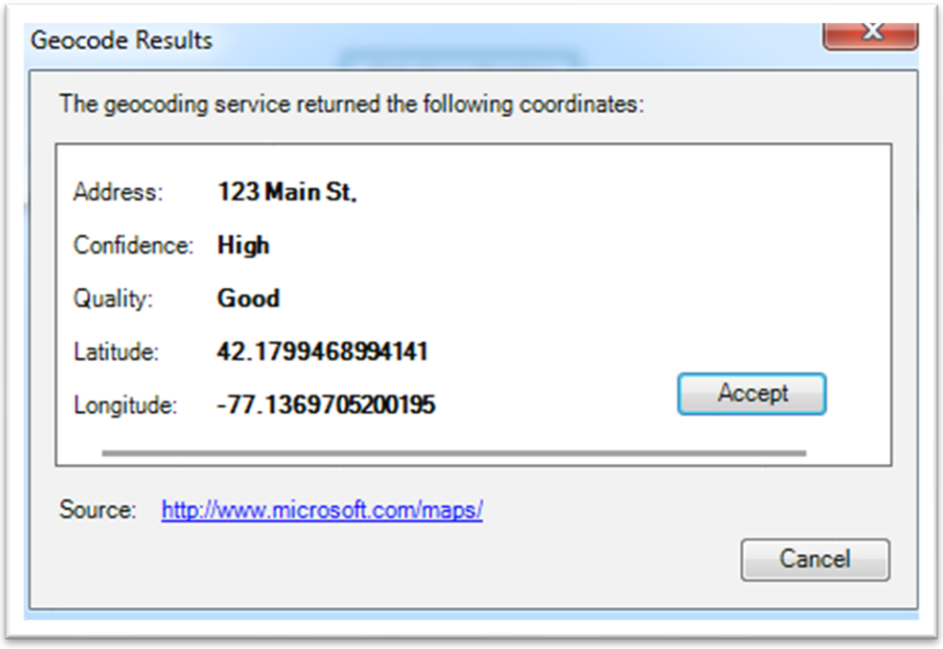
GEOCODE Check Code Command

3. When clicking the **GetCoordinates** command button after adding an address from the **Enter Data** tool, the geocode results appear.



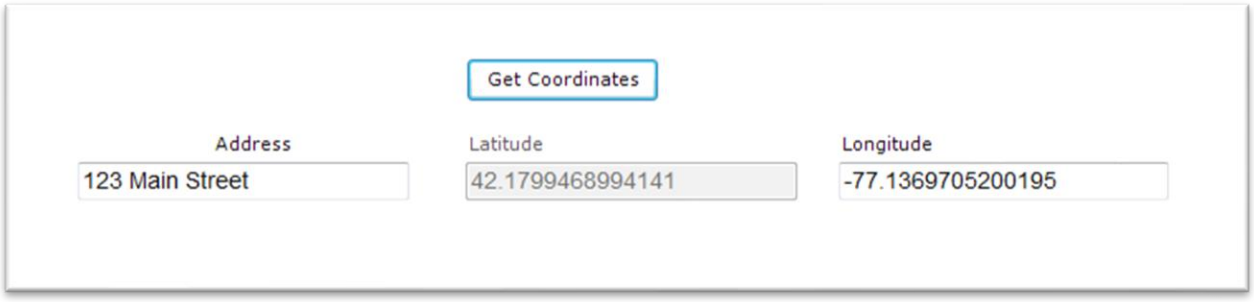
Geocode Enter Data Before

The **Geocode Results** dialog box contains the address, **Latitude** and **Longitude** coordinates. Also, you can view the confidence level of geocode service coordinates.



Geocode Results window

Click Accept in the **Geocode Results** dialog box. The geocode service coordinates fill the **Latitude** and **Longitude** fields in your form.



Geocode Enter Data After

HELP

Displays a pop-up help window containing a message. The help window works with large files by allowing users to move from one highlighted block of text to another. It works similarly to hypertext.

HIGHLIGHT/UNHIGHLIGHT

Highlight/Unhighlight emphasizes field locations by highlighting them in bright yellow. Highlighting is useful when errors are detected. Fields can be highlighted and unhighlighted. View the example below. The highlighted field **Emergency** is triggered by a response to the field**: Vaccinated = "No"**. So, the field: "**Did you visit the emergency room?"** is highlighted, skipping over the rest of the vaccination questions.

```
Field Vaccinated
      After
            IF Vaccinated = (-) THEN
                  GOTO emergency
                  HIGHLIGHT emergency
            ELSE
                  UNHIGHLIGHT emergency
            END-IF

      End-After
End-Field
```

HIGHLIGHT/UNHIGHLIGHT command

NEWRECORD

The NewRecord command saves the current records data and opens a new record for data entry.

```
Field FinishInterview
      After
            //add code here
            IF FinishInterview = (+) THEN
                  NEWRECORD
            END-IF
      End-After
End-Field
```

syntax for Field NEWRECORD command

QUIT

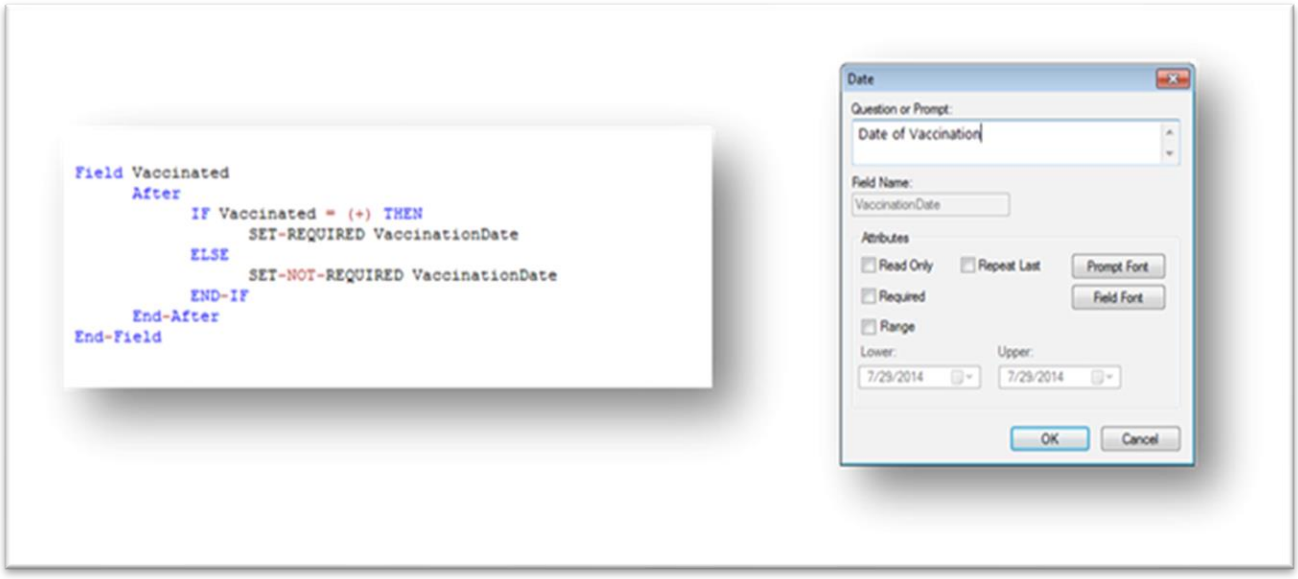
This command saves the current record and closes the Enter application.

```
Field DoneWithEnteringData
      After
            //add code here
            IF DoneWithEnteringData = (+) THEN
                  QUIT
            END-IF
      End-After
End-Field
```

Check Code syntax for QUIT command

SET-REQUIRED and **SET-NOT-REQUIRED**

You can set fields to **REQUIRED**, based on respondent criteria. Fields set to **REQUIRED** prevent records from being saved until the fields are filled in. Entering data allows participants to continue. In the example below, the field: **VaccinationDate** is set to **REQUIRED** if the participant answers, **"Yes"** to "have you received a vaccination?" View the **SET-REQUIRED** example below. The **REQUIRE** property is set with Check Code, and not applied during field creation. Entering a specific criterion resets the field to its original state by using the **SET-NOT-REQUIRED** command.



Check Code syntax for SET-REQUIRED command

Check Features Covered Elsewhere

You can setup the **Enter Data** application for complex operations not listed in this chapter. Examples include mathematical and logical operations using multiple fields, help window pop-ups and calls to field-content programs in various languages.

How to Use the Epiweek Function

Epidemiological weeks are complete weeks. Some worldwide ministries of health consider Sunday as the first epidemiological day of the week. Others consider the first epidemiological day of the week to be Saturday or Monday.

By default, Epi Info™ 7 marks Sunday, the beginning of the epidemiological week. Modify the parameter for the **EpiWeek** function, changing the first day of the epidemiological week.

Use the **EpiWeek** function if the year doesn't matter. Using **EpiWeek** creates a numeric value stored in a numeric field. **EPIWEEK** requires the date parameter. Weeks are calculated based on the date (i.e.: year). View the epidemiological week syntax below:

ASSIGN MyEpiWeek = **EPIWEEK**(**<start_date>, {<first_day_of_week>}**)

The Check Code shown below, calculates **EPIWEEK** into the **SurveillanceWeek** field. It requires a value entry in **OnsetDate**.

```
Field OnsetDate
      After
            //add code here
            ASSIGN SurveillanceWeek = EPIWEEK( OnsetDate)
      End-After
End-Field
```

Check Code syntax for EPIWEEK function

Change the Check Code syntax if the first day of the epidemiological week doesn't start on Sunday. In the example, set the epidemiological week to Monday (i.e., Sunday will be 1, Monday will be 2, Tuesday will be 3, etc.)

```
Field OnsetDate
      After
            //add code here
            ASSIGN SurveillanceWeek = EPIWEEK( OnsetDate,2)

      End-After
End-Field
```

EPIWEEK with beginning week parameter

Proper Check Code Syntax

Check Code syntax must match the field type. Here are some examples of proper syntax for a field type:

- Assign the 'Age' field (numeric field type) the value 24
  - **ASSIGN Age = 24**
- Assign the 'Ill' field (Yes/No field type) the value No
  - **ASSIGN Ill = (-)**
- Assign the 'AteChicken' field (checkbox field type) the value Yes
  - **ASSIGN AteChicken = (+)**
- Assign the 'DateOfInterview' field (Date field type) the value 5/5/2012
  - **ASSIGN DateOfInterview = 5/5/2012**
- Assign the 'CaseStatus' field (legal values field type) the value "Confirmed"
  - **ASSIGN CaseStatus = "Confirmed"**
- Assign the 'Sex' field (Comment Legal field) the value "F". The field code is M-Male, and F-Female.
  - **ASSIGN Sex = "F"**

This page has been intentionally left blank.


DEPARTMENT OF HEALTH & HUMAN SERVICES · USA
CDC
CENTERS FOR DISEASE CONTROL AND PREVENTION
CS330911-A