

# Using Satellite Forms™ with Microsoft® Access 97™ A Step by Step Tutorial

Satellite Forms™ 

Prepared By:  
Reema Mukhtar  
Last Edit: 8/24/99



## Using Satellite Forms™ with Microsoft® Access 97™ A Step by Step Tutorial

---

### Introduction

This document describes how to integrate your Puma Technology® Satellite Forms application with Corporate Data stored in Microsoft Access. This tutorial applies to using Satellite Forms, Standard Edition or Enterprise Edition with Microsoft Access 97. Although the description here takes advantage of some specific features of Access, many of the concepts are general and apply to other database management systems as well. In this document, when we refer to Corporate Data or Corporate Database, we will be referring to data stored in native Access tables on the PC Server. This data is distinct from data stored in Satellite Forms dBase 5 data transfer tables.

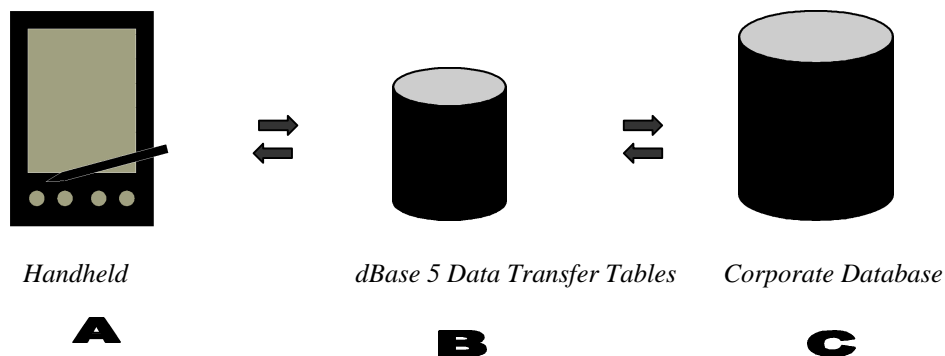
We first describe the multiple locations where data is stored in a Satellite Forms / Access application, and give an overview of how data is moved from one location to another. We then describe in detail the steps to integrate the sample “Work Order” application with Access. You should read Chapter 6 of the Satellite Forms manual for background on the integration process, and Chapter 7 for background on the “Work Order” application.

### Overview of Data Storage and Transfer

In any Satellite Forms application that interchanges data with a Corporate Database, data resides in three locations:

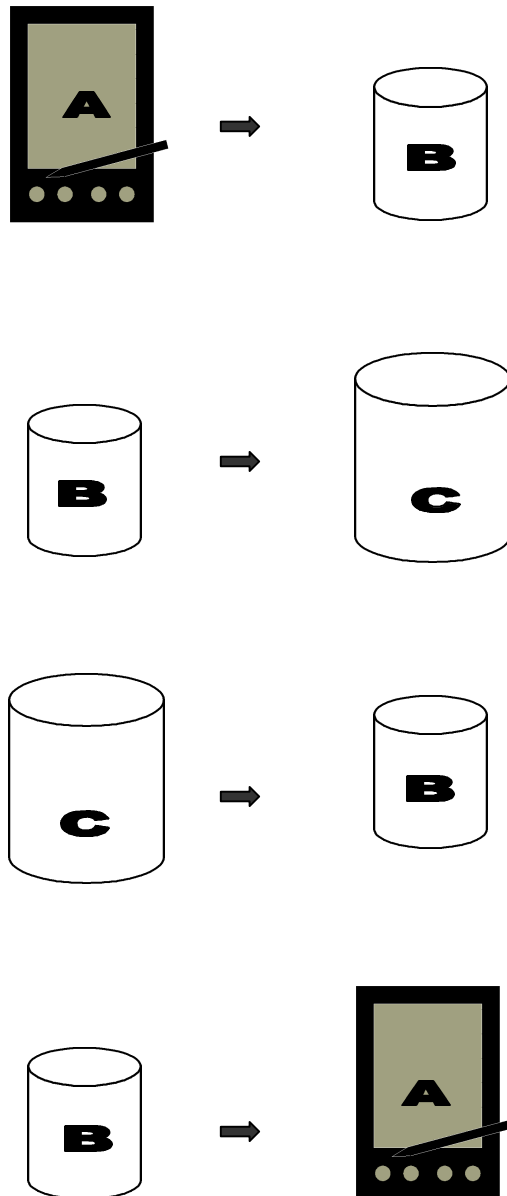
- A. The Handheld**
- B. The Satellite Forms dBase 5 data transfer tables**
- C. The Corporate Database**

This data architecture can be illustrated conceptually as follows:



As the arrows indicate, data must pass through the intermediate dBase 5 data transfer tables (B) to be moved between the Handheld and the Corporate Database. Some or all of the data exchanges indicated by the arrows will occur during each HotSync operation. The developer can control which data exchanges occur and the order in which they occur. This gives complete flexibility in managing HotSync. Note that the dBase 5 data transfer tables are used as temporary storage areas to facilitate the data transfer. Data in these tables are completely overwritten by successive HotSync operations.

To successfully integrate Corporate Data with your Satellite Forms application, you must understand how to accomplish each of the four data transfers by the arrows in the diagram:



You must also understand how to manage the HotSync operation using the methods and events of the Satellite Forms ActiveX® Control. In the example below, we describe each of these steps in detail.

## Integration of Sample “Work Order” Application

In this example, we use the “Work Order” application that is discussed in Chapter 7 of the Satellite Forms manual. If you wish to create this application yourself, you can follow the steps outlined in Chapter 10.

Alternatively, you can use the completed application in the Satellite Forms samples directory. In order to use this tutorial, you must load the work order application to your Palm. The steps we will follow to integrate the “Work Order” application with Access are:

**Step 1** Decide how data should move between Handhelds and Access.

**Step 2** Prepare Access to integrate with Satellite Forms.

**Step 3** Create Access tables to serve as the Corporate Database.

**Step 4** Build a Access Form which includes the Satellite Forms ActiveX Control.

**Step 5** Write General Declarations and a Form\_Load subroutine in the Access form.

**Step 6** Write the HotSync Event Handler in Access to manage HotSync.

### **Step 1** Decide how data should move between Handhelds and Access

The first step in any integration effort is determining which data should move between systems. This is highly dependent on the objectives of the application. Recall the objectives of the “Work Order” application by reviewing its description in Chapter 10.

The “Work Order” application is made up of three tables: *wrkSites*, *wrkWorkItems*, and *wrkLookup*. We will look at the data associated with each of these tables, in turn, to determine the following:

- Where records can be created (Handhelds, Corporate Database, or both)
- Where, if anywhere, records can be updated, and what fields can be updated
- Where, if anywhere, records can be deleted
- Where, if anywhere, the records should be transferred, and what fields should be transferred

#### “wrkSites” Table

The *wrkSites* table contains information about client sites. New *wrkSites* records are created in the Corporate Database when the order is taken. This new information should then be passed from the Corporate Database to Handhelds. Updates to *wrkSites* information can occur on the Handheld. Any changes that occur should be passed back to the Corporate Database. For simplicity, we will assume that *wrkSites* records are only updated in the field and cannot be updated by workers at headquarters (in the Corporate Database). Furthermore, we will assume that records cannot be deleted, either in the field, or at headquarters. Finally, we will send all *wrkSites* records to all Handhelds, rather than sending specific records to specific work crews. You can relax these constraints after you understand the basics of application integration.

#### “wrkWorkItems” Table

The *wrkWorkItems* table contains a list of work items to be completed. Again, records are created in the Corporate Database when the order is taken. This new information should then be passed from the Corporate Database to Handhelds. Updates to *wrkWorkItems* information can occur on the Handheld. Any changes that occur should be passed back to the Corporate Database. For simplicity, we will assume that *wrkWorkItems* records are only updated in the field and cannot be updated by workers at headquarters (in the Corporate Database). Furthermore, we will assume that records cannot be deleted, either in the field, or at headquarters.

#### “wrkLookup” Table

The wrkLookup table contains only static data that does not change. This data can be downloaded with the application and remain constant after that. It will not be updated by workers in the field or by workers at headquarters.

The following matrix summarizes the data manipulation that we will achieve in the integrated application. Checkmarks indicate that the operation is permitted for records in the corresponding table. It is a good idea to draw a similar matrix for your custom applications.

Table	Handheld			Corporate Database		
	Insert	Update	Delete	Insert	Update	Delete
<i>wrkSites</i>		✓		✓		
<i>wrkWorkItems</i>		✓		✓		
<i>wrkLookup</i>						

In summary, during each HotSync of the “Work Order” application, two operations must occur:

- Updated *wrkSites* and *wrkWorkItems* records must move from the Handhelds to the Corporate Database. In our previous notation, these records move **A** to **B** and then **B** to **C**.
- New *wrkSites* and *wrkWorkItems* records that are created in the Corporate Database must move to Handhelds. These records move **C** to **B** and then **B** to **A**.

We will perform these operations in a specific order, based on the requirements of the “Work Order” application. First, we will move updated information from Handhelds to the Corporate Database. Next, we will find new records in the Corporate Database and move them to Handhelds. Understanding how to perform these operations will give you the building blocks to manage more complex data exchanges. The next sections of this document describe how to complete these processes.

## Step 2 Prepare Access to Integrate with Satellite Forms

Preparing Access to integrate with Satellite Forms requires three steps:

1. Creating an Access database
2. Linking the Satellite Forms dBase 5 data transfer tables to the Access database
3. Importing the Satellite Forms API into the Access database

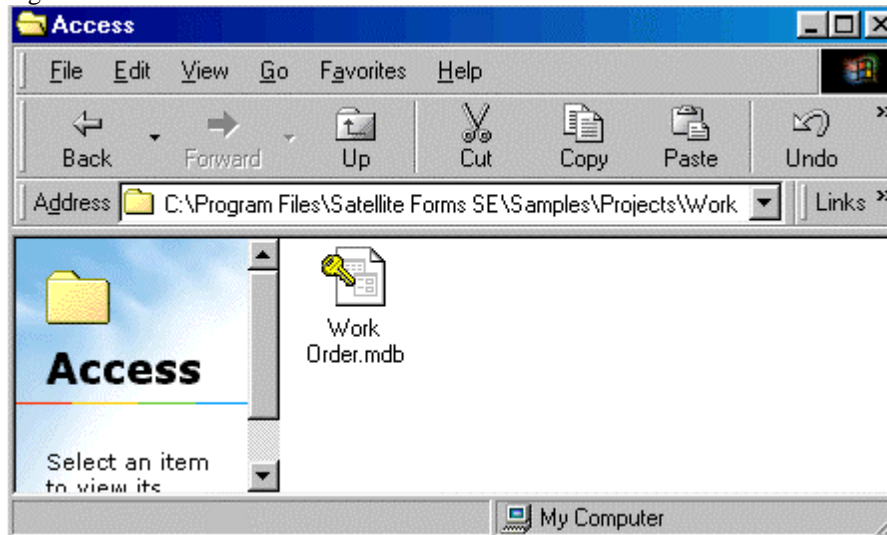
### Creating an Access database

First create a new subdirectory Access in the "Work Order" sample directory, which is typically:  
C:\Program Files\Satellite Forms\Samples\Projects\WorkOrder

Next, create a new, blank Access database and save it in this new subdirectory as "Work Order".  
The path to the application will then be:

C:\Program Files\Satellite Forms\Samples\Projects\Work Order\Access\Work Order.mdb See  
Figure 1 for the correct location of the application.

Figure 1. Work Order Database



### Linking the Satellite Forms dBase 5 data transfer tables to the Access database

Access can read linked dBase 5 tables as if they were native Access tables. Linking the Satellite Forms dBase 5 data transfer tables into Access will make it easier to test your program because you will be able to view the data in these tables directly from within Access.

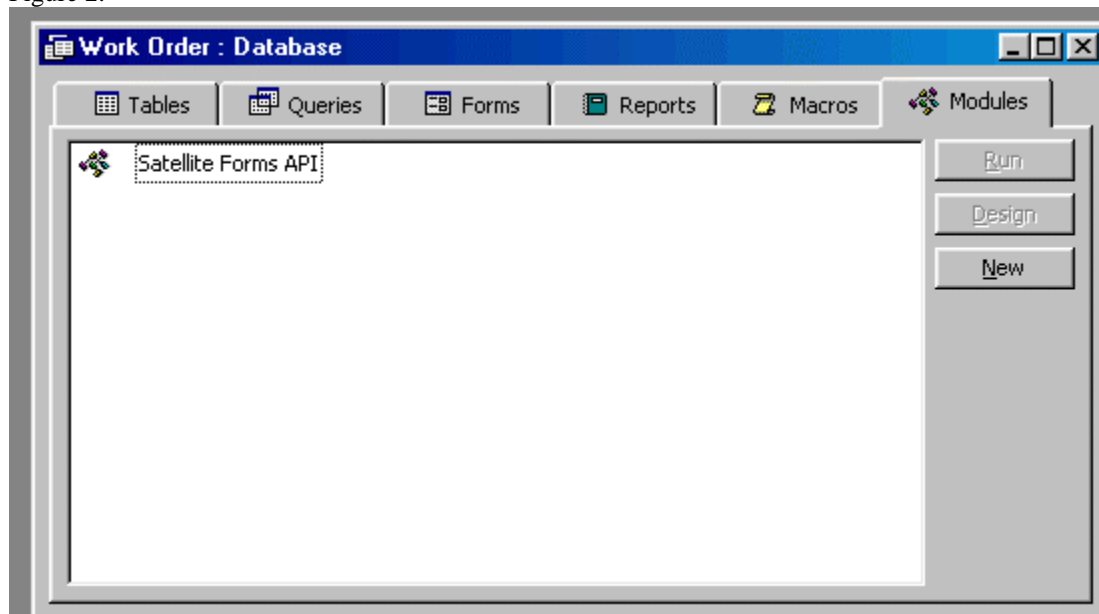
Follow the instructions for linking tables in chapter 7 of the Satellite Forms documentation. Link the "Work Order" dBase 5 tables: WRKLOOKU.dbf, WRKSITES.dbf, and WRKWORKI.dbf into your "Work Order" Access database.

### Import the Satellite Forms API into the Access database

Importing the Satellite Forms API ensures that Access will successfully execute the code that you write to manage the HotSync operation. The API sets up several constants and parameters that will be used by your code. For Access users, an Access Add-in named SATFORMS.MDA is present in the INCLUDE directory of the Satellite Forms Installation. This add-in contains useful constants and information that you can use when writing your HotSync handling code.

To enable this add-in, click on the Modules tab of your application and select the File/Get External Data/Import menu selection. If you installed to the default location, the C:\PROGRAM FILES\Satellite Forms SE(EE)\INCLUDE directory will contain the SATFORMS.MDA add-in file. See Figure 2 for an example.

Figure 2.



### **Step 3. Create Access tables to serve as the Corporate Database**

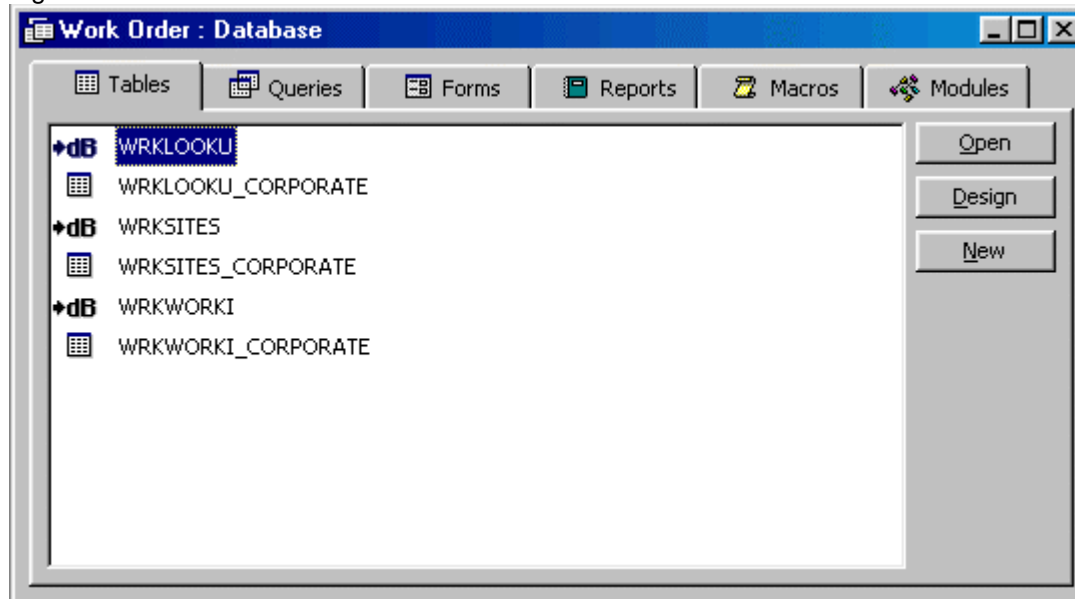
In the Tables tab of Access, you should see the three linked Satellite Forms dBase 5 data transfer tables (marked with dB). You will now need to create three additional tables in Access to represent the Corporate Database. These tables should have the same structure as the dBase 5 tables created by Satellite Forms. In general, the structure of the Corporate Database and the dBase 5 data transfer tables of your custom application may differ. This will occur if you do not need to pass all your data between Handhelds and the Corporate Database. Some data, for example, may be specific to workers at corporate headquarters and passing this data to workers in the field is unnecessary.

You can quickly create the Corporate Database tables by importing copies of the dBase 5 data transfer tables into Access. You have already linked these tables, but now you will make copies of them. From the File menu, select Get External Data / Import... Look for files of type dBase 5 in the Work Order directory. Import each of the three tables. Access will create tables called *WRKLOOKU1*, *WRKSITES1*, and *WRKWORK11*. Rename these tables to *WRKLOOKU\_CORPORATE*, *WRKSITES\_CORPORATE*, and *WRKWORK1\_CORPORATE*.

When you are finished, you should see a total of six tables in Access. Double click each table to view the data in the tables.

See Figure 3

Figure 3.



- |                      |                                    |
|----------------------|------------------------------------|
| • WRKLOOKU           | Linked dBase 5 data transfer table |
| • WRKLOOKU_CORPORATE | Access Corporate Database table    |
| • WRKSITES           | Linked dBase 5 data transfer table |
| • WRKSITES_CORPORATE | Access Corporate Database table    |
| • WRKWORKI           | Linked dBase 5 data transfer table |
| • WRKWORKI_CORPORATE | Access Corporate Database table    |

#### **Step 4. Build an Access Form including the ActiveX Control**

For the HotSync operation to correctly transfer data between Access and Satellite Forms, your Access server application must be running when the HotSync occurs. The easiest way to do this is to build a form as part of your application and have this form open (running) on the Server PC (the PC to which users HotSync). This form could be the same form used at headquarters to enter data directly in the Corporate Database, or it could be a different form.

To create a new form, select the Forms tab in Access and choose New. Use the Form Wizard to create a form of type: *AutoForm: Columnar* on the table *WRKSITES\_CORPORATE*. Save this form as **frmWrkSites**.

Next, you will need to add the Satellite Forms ActiveX Control to this form. Once you are in the design mode, select ActiveX Control from the Insert menu. Refer to Figure 4 for an example of the Insert menu. Rename the control so it is called **SatForms**. To rename the control, right click the ActiveX control icon and select the ActiveX control properties view. Please see Figure 5 to view the ActiveX Control Icon. In the name field, rename the control to **SatForms**. We will refer to it with this name in our code to manage the HotSync operation



Figure 4. Select the Insert Menu

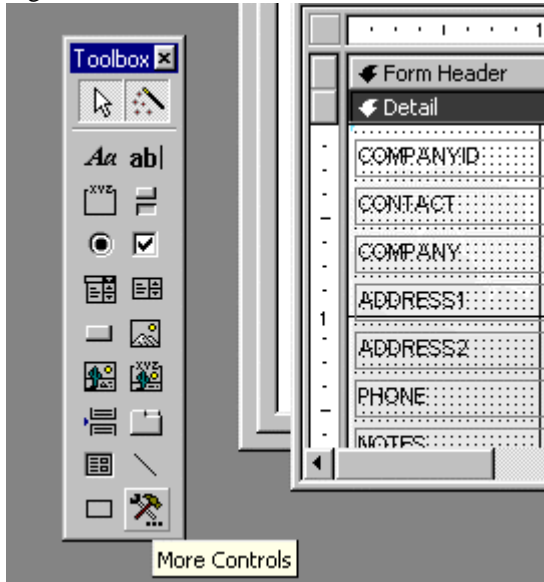
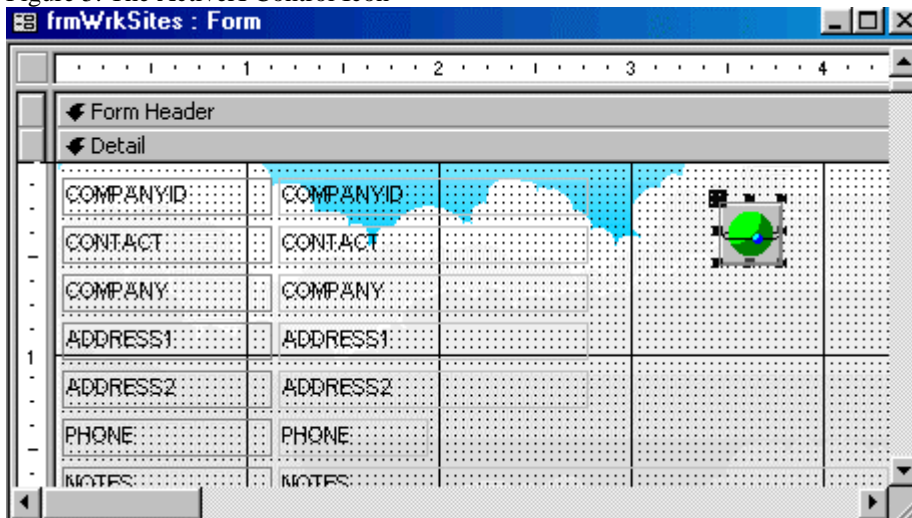


Figure 5. The ActiveX Control Icon



### Step 5. Write General Declarations and a Form\_Load subroutine in Access

The following steps of this document will discuss the code that you will write to manage the HotSync operation. Before we start moving the data, we must do some set up by declaring and setting a number of variables and creating a Form\_Load subroutine.

## Performing the General Declarations

On the Forms tab in Access, select View then Code from the Access menu and add the following Dim statements under General Declarations (Option Compare Database and Option Explicit will

already be there). These new variables will be used to store the file locations of the dBase 5 data transfer tables and track the progress of the HotSync operation:

```
Option Compare Database
Option Explicit

Const Status_HotSyncStart = 1
Const Status_HotSyncCommanComplete = 3
Const Status_HotSyncEnd = 2

Dim TableFilename_WRKLOOKU As String
Dim TableFilename_WRKSITES As String
Dim TableFilename_WRKWORKI As String
Dim HotSync_Progress As String
Dim CmdCount As Integer
```

### Creating the Form\_Load Subroutine

Add a new procedure called Form\_Load. To add this procedure, select the *insert* pull down menu from the Access toolbar and select *procedure*. Add the code below to your procedure. This procedure will disable notifications from Access about us changing data. It will set the values of the variables and enable the Satellite forms ActiveX control.

```
Public Sub Form_Load()

'Disable Warnings
DoCmd.SetWarnings 0

TableFilename_WRKLOOKU = "c:\Program Files\Satellite Forms
SE\Samples\Projects\Work Order\WRKLOOKU.dbf"
TableFilename_WRKSITES = "c:\Program Files\Satellite Forms
SE\Samples\Projects\Work Order\WRKSITES.dbf"
TableFilename_WRKWORKI = "c:\Program Files\Satellite Forms
SE\Samples\Projects\Work Order\WRKWORKI.dbf"
HotSync_Progress = "Begin"
SatForms.Enabled = True

End Sub
```

## Step 6 Write the HotSync Event Handler in Access to manage HotSync

You will now write the code that will move the data from Handhelds to the Corporate Database and from the Corporate Database back to Handhelds. As described earlier, there are four data transfers that we need to perform:

- A to B      **updated records from Handhelds to dBase 5 data transfer tables**
- B to C      updated records from dBase 5 data transfer tables to Access database
- C to B      new records from Access database to dBase 5 data transfer tables
- B to A      new records from dBase 5 data transfer tables to Handhelds

We will look at each of these steps in detail, working with the *wrkSites* and *wrkWorkItems* tables.

### **Step 6a. Moving Data A to B**

You will create the subroutine **SatForms\_HotSyncStatus**, which will serve as your HotSync event handler. To move data from the Handheld to the dBase 5 data transfer tables, you will use the **GetTableFromPalmPilot** method of the Satellite Forms ActiveX control. To transfer multiple tables, you may call this method more than once using different filenames. The variable CmdCount is used to keep track of how many tables are being transferred (or commands have been given). The subroutine should look like:

```
Private Sub SatForms_HotSyncStatus(ByVal StatusCode As Long,
ByVal Param As Long)

    If StatusCode = Status_HotSyncEnd Then
        'We're done
        HotSync_Progress = "End"
        Exit Sub
    End If

    If StatusCode = Status_HotSyncStart Then
        'Move data from A --> B
        SatForms.GetTableFromPalmPilot (TableFilename_WRKWORKI)
        SatForms.GetTableFromPalmPilot (TableFilename_WRKSITES)
        CmdCount = 2 'Count 2 commands to transfer files
        HotSync_Progress = "A>B"
    End If

End Sub
```

You should test this procedure to ensure that changes you make to the handheld's *wrkSites* and *wrkItems* data are reflected in the dBase 5 data transfer tables.

Note: In order to execute your new code, you must open the **frmWrksites** form

Verify the data is transferring by making changes to the data on your handheld then sync'ing it back to Access. The **GetTableFromPalmPilot** methods will overwrite all existing data in the tables with data from the handheld.

### **Step 6b. Moving Data B to C**

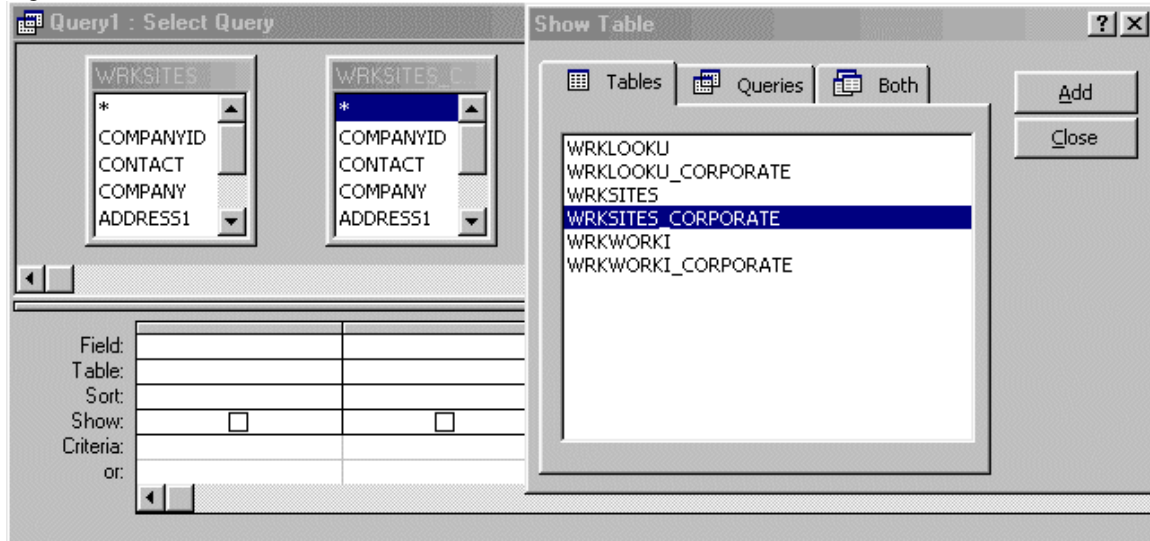
It is important that your application completes the movement of data A to B before it tries to move this data B to C. After you have successfully moved data from the handheld to the dBase 5 data transfer tables, you will need to move it further along, to the Corporate Database. Use the notification *HotSyncCommandComplete* and *CmdCount* to determine when this occurs. This sample uses the *wrkSites* table, you will need to write similar code for the *wrkWorkItems* table.

You will need to use a Access **Update Query** to move data from the dBase 5 data transfer tables to the Corporate Database. This query will update rows in the Access table based on rows in the dBase 5 table by matching on the key field. Because no records are being created on Handhelds, we don't have to worry about adding additional records to the Corporate Database. If

this were the case, however, we would have to write an additional Append Query to insert these new records.

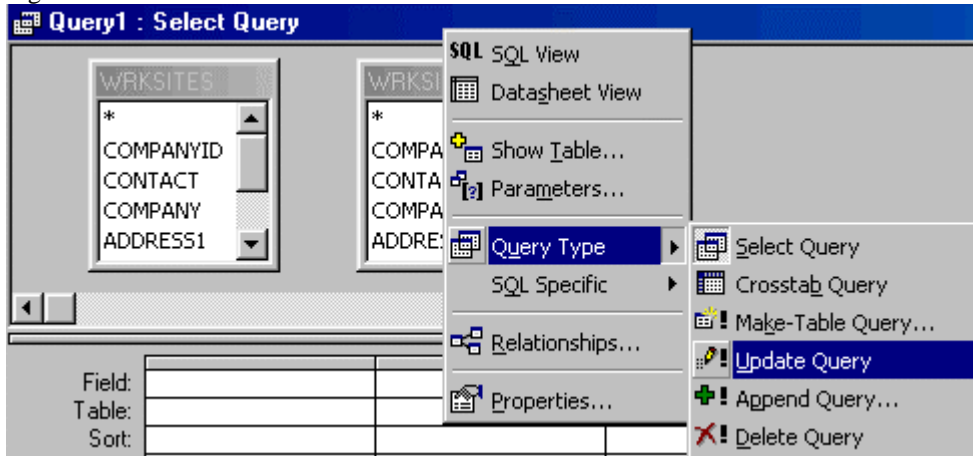
To create a query in Access, select the **Queries** tab and click New. This will bring up the New Query window. Select Design View. At this point you will have to select the tables that will be part of the query. Select WRKSITES and press Add. Next select WRKSITES\_CORPORATE and press Add. This query will only involve those two tables, so we are done. Press Close.

Figure 6.



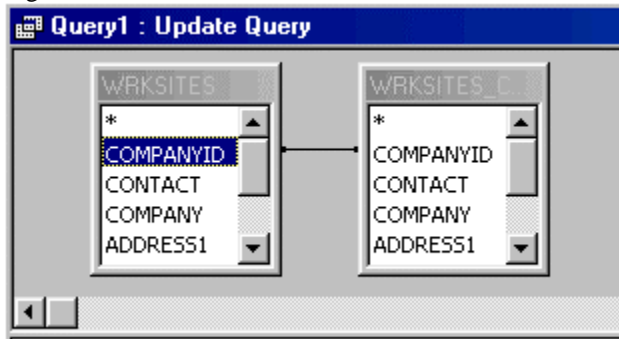
To make this query an update query, right click in the title bar of **Query1** and click on the down arrow next to the **Query Type** button on the toolbar. Change the type from **Select Query** to **Update Query**. See Figure 7 for an example.

Figure 7.



Join the two tables together by clicking on *COMPANYID* in *WRKSITES\_CORPORATE* and dragging this field on top of *COMPANYID* in *WRKSITES*. Access will draw a line between the two tables on the *COMPANYID* field showing how they are joined. See figure 8.

Figure 8.



Our query will update all the fields in *WRKSITES\_CORPORATE* (except *COMPANYID*) based on the corresponding fields in *WRKSITES*. Drag each of the fields from *WRKSITES\_CORPORATE* except *COMPANYID* into the space labeled *Field* in the bottom part of the screen. These fields will be updated by this query.

In the *Update To* field under each of the columns, type in the corresponding field from the *WRKSITES* table. For example, in the *CONTACT* field, type: *wrksites.contact*. For the *COMPANY* field type: *wrksites.company*. Access will add brackets to your syntax. When you are finished, close the window and save the Query as *UPDATE\_WRKSITES\_CORPORATE*.

You can test your query by making some changes to columns in the *WRKSITES* table (other than the *COMPANYID*) and opening your query by selecting *Open* when the *Queries* tab is selected. Access will prompt you that you are about to change data. This is OK.<sup>1</sup>

Now, you will need to add additional code to your *HotSync* handler to run this query. This code should execute after the methods **GetTableFromPilot** have completed. Use the status code *HotSyncCommandComplete* and *CmdCount* to determine when this occurs.

The *HotSyncCommandComplete* event will occur once for every command given, or file transferred. In this case there were two commands to upload files, so this event will occur twice before both tables are uploaded. Decrement *CmdCount* each time the *Status\_HotSyncCommandComplete* even fires. When *CmdCount* has decremented to zero, all the commands have completed (or files uploaded) and it is time to process the data. At this point, your event handler should look like:

---

<sup>1</sup> Access will still prompt you one time when you open (run) the query. However, when the query is run from the *HotSync* event handler, Access will not prompt at all because the *Form\_Load* subroutine disables notifications.

```
Private Sub SatForms_HotSyncStatus(ByVal StatusCode As Long, ByVal
Param As Long)
```

```
    If StatusCode = Status_HotSyncEnd Then
```

```
        'We're done
```

```
        HotSync_Progress = "End"
```

```
        Exit Sub
```

```
    End If
```

```
    If StatusCode = Status_HotSyncStart Then
```

```
        'Move data from A --> B
```

```
        SatForms.GetTableFromPalmPilot (TableFilename_WRKWORKI)
```

```
        SatForms.GetTableFromPalmPilot (TableFilename_WRKSITES)
```

```
        CmdCount = 2 'Count 2 commands to transfer files
```

```
        HotSync_Progress = "A>B"
```

```
    End If
```

```
    If StatusCode = Status_HotSyncCommandComplete Then
```

```
        CmdCount = CmdCount - 1
```

```
        If CmdCount <> 0 Then GoTo CmdCompleteExit 'File set not
finished
```

```
        If HotSync_Progress = "A>B" Then
```

```
            'Move data from B --> C
```

```
            DoCmd.OpenQuery ("Update_WRKWORKI_CORPORATE")
```

```
            DoCmd.OpenQuery ("Update_WRKSITES_CORPORATE")
```

```
            HotSync_Progress = "B>C"
```

```
        End If
```

```
CmdCompleteExit:
```

```
    End If
```

```
End Sub
```

Test your program to see if it works. Add or delete information to the mobile client Work Order application and Sync it back to your desktop. Changes to data on Handhelds should be reflected in the table WRKSITES\_CORPORATE after HotSync has completed. Also remember to add and test the code for the WRKWORKI\_CORPORATE table. See step 6a for detailed instructions.

### **Step 6c. Moving Data C to B**

We will need to move any new records that have been added to the Corporate Database in the WRKSITES and WRKWORKI tables to the dBase 5 data transfer tables. You will do this with an **Append Query**.

Choose the Access Queries tab and select New. Choose the Design View. This time, choose only the WRKSITES\_CORPORATE table. This will be where the data is coming from. Change the Query type to an Append Query. When prompted by Access, append to the table WRKSITES in the current database.

Drag each column, one at a time, from the WRKSITES\_CORPORATE table into the *Field* area in the bottom part of the screen. Each column of the WRKSITES\_CORPORATE table will be in a separate column on the screen.

Next, modify the query so it only appends rows which don't already exist. Do this by filling out the *Criteria* field for COMPANYID. Set that field to:

**[wrksites\_corporate].[companyid] Not In (select companyid from wrksites)**

Your table should now look like Figure 9.

Figure 9.

Field	COMPANYID	CONTACT	COMPANY	ADDRESS1	ADDRESS2	PHONE
Table:	WRKSITES_CORPORATE	WRKSITES_CORP	WRKSITES_CORP	WRKSITES_CORP	WRKSITES_CORP	WRKSITES_CORP
Append To:	COMPANYID	CONTACT	COMPANY	ADDRESS1	ADDRESS2	PHONE
Criteria:	[wrksites_corporate].[companyid]					

Save the Query as INSERT\_WRKSITES. You can test it by adding a few records to WRKSITES\_CORPORATE and verifying that they move to WRKSITES when you open (run) the query in Access.

Repeat the steps above for the WRKWORKI\_CORPORATE tables.

You will now have to modify your HotSync event handler to run this code automatically. Your handler should now look like:

```
Private Sub SatForms_HotSyncStatus(ByVal StatusCode As Long, ByVal
Param As Long)

    If StatusCode = Status_HotSyncEnd Then
        'We're done
        HotSync_Progress = "End"
        Exit Sub
    End If

    If StatusCode = Status_HotSyncStart Then
        'Move data from A --> B
        SatForms.GetTableFromPalmPilot (TableFilename_WRKWORKI)
        SatForms.GetTableFromPalmPilot (TableFilename_WRKSITES)
        CmdCount = 2 'Count 2 commands to transfer files
        HotSync_Progress = "A>B"
    End If

    If StatusCode = Status_HotSyncCommandComplete Then
        CmdCount = CmdCount - 1
        If CmdCount <> 0 Then GoTo CmdCompleteExit 'File set not
finished
```

```

If HotSync_Progress = "A>B" Then
    'Move data from B --> C
    DoCmd.OpenQuery ("Update_WRKWORKI_CORPORATE")
    DoCmd.OpenQuery ("Update_WRKSITES_CORPORATE")
    HotSync_Progress = "B>C"
End If

If HotSync_Progress = "B>C" Then
    'Move data from C --> B
    DoCmd.OpenQuery ("Insert_WRKWORKI")
    DoCmd.OpenQuery ("Insert_WRKSITES")
    HotSync_Progress = "C>B"
End If

```

CmdCompleteExit:

```
End If
```

```
End Sub
```

### **Step 6d. Moving Data B to A**

The last step is moving the dBase 5 data back to Handhelds. The dBase 5 data will reflect previous updates passed to the Corporate Database from Handhelds as well as any new records that have been added to the Corporate Database.

Add code to your HotSync event handler to run the **CopyTableToPalmPilot** methods. This will take the data from the dBase 5 data transfer tables and move it to handheld. Your event handler should now look like:

```
Private Sub SatForms_HotSyncStatus(ByVal StatusCode As Long, ByVal
Param As Long)
```

```

If StatusCode = Status_HotSyncEnd Then
    'We're done
    HotSync_Progress = "End"
    Exit Sub
End If

```

```

If StatusCode = Status_HotSyncStart Then
    'Move data from A --> B
    SatForms.GetTableFromPalmPilot (TableFilename_WRKWORKI)
    SatForms.GetTableFromPalmPilot (TableFilename_WRKSITES)
    CmdCount = 2 'Count 2 commands to transfer files
    HotSync_Progress = "A>B"
End If

```

```

If StatusCode = Status_HotSyncCommandComplete Then
    CmdCount = CmdCount - 1
    If CmdCount <> 0 Then GoTo CmdCompleteExit 'File set not
finished

```

```

If HotSync_Progress = "A>B" Then
    'Move data from B --> C

```



```

        DoCmd.OpenQuery ("Update_WRKWORKI_CORPORATE")
        DoCmd.OpenQuery ("Update_WRKSITES_CORPORATE")
        HotSync_Progress = "B>C"
    End If

    If HotSync_Progress = "B>C" Then
        'Move data from C --> B
        DoCmd.OpenQuery ("Insert_WRKWORKI")
        DoCmd.OpenQuery ("Insert_WRKSITES")
        HotSync_Progress = "C>B"
    End If

    If HotSync_Progress = "C>B" Then
        'Move data from B--> A
        SatForms.CopyTableToPalmPilot (TableFilename_WRKLOOKU)
        SatForms.CopyTableToPalmPilot (TableFilename_WRKSITES)
        SatForms.CopyTableToPalmPilot (TableFilename_WRKWORKI)
        CmdCount = 3 'Count 3 commands to transfer files
        HotSync_Progress = "B>A"
        Exit Sub
    End If

CmdCompleteExit:

    End If

End Sub

```

## Final Testing

You should test this procedure to ensure that new records are moved successfully to the Handheld. As a test, you can add a record from within Access and see that this record moves to the Handheld. You can also check the functionality of your new procedure by adding data to your handheld application and checking to see if the data is moved successfully to Access. You can add new records to the Corporate Tables through your form and ensure that these records move successfully to the Handheld.

## Distributing your software using the RDK

When distributing your software, you will be using the Re - Distribution Kit's (RDK) Redistributable software. If you have the Satellite Forms Standard Edition or Enterprise Edition, the re - distribution kit is included with the CD set. The RDK allows you to create an icon to launch your application. This icon file contains a bitmap and information about the program to execute when tapped. All of the icons on your handheld's Applications picker screen are icon files.

Each icon file has a unique four character CreatorID string. This CreatorID is used to group program and data files with a particular application. When a HotSync is performed, the Satellite Forms ActiveX control will fire once for each Satellite Forms icon file on the handheld. Each application's icon file will fire the HotSyncStatus event while passing it's CreatorID to the ActiveX

control. It is necessary to exit the HotSyncStatus event if the event was not fired by your application.

The Satellite forms Enterprise Edition SDK uses "SMSF" as its CreatorID string. The Satellite Forms Standard Edition uses "SMSD" as its CreatorID. Our RTK and some of the samples use the CreatorID strings "SMS0" through "SMS9". You can use CreatorID's "SMS0" through "SMS9" for testing. Your program should have it's own unique CreatorID string. A list of unique CreatorID's is maintained by 3Com®. To obtain your own unique CreatorID, go to [www.palm.com/devzone/crid/cridsub](http://www.palm.com/devzone/crid/cridsub) . A CreatorID will be assigned to you at no charge.

## Conclusions

At this point, you have successfully moved data from a single table between the Handheld and a Corporate Database, and you have moved this data in two directions. You can then try extending the capabilities of the Work Order application to:

- Allow new records to be created on Handhelds and passed to the Corporate Database
- Allow records to be updated on the Corporate Database and passed to Handhelds
- Allow records to be deleted on Handhelds and passed to the Corporate database
- Allow records to be deleted on the Corporate Database and passed to Handhelds

You will want to experiment with the result codes returned by the HotSync Extension ActiveX control. You can enhance your application to provide appropriate warnings and error messages if the HotSync event doesn't proceed as planned. Refer to Chapter 6 of the Satellite Forms manual, as well as "Customers" sample application for more information.

© 1997 - 1999 Puma Technology, Inc. All rights reserved.

Puma Technology, the Puma Technology logo, Satellite Forms and the Satellite Forms logo are trademarks of Puma Technology, Inc. that may be registered in some jurisdictions. All other company and product names are trademarks of their respective owners.