13 Informed 4D Externals

In this chapter:

13 Informed 4D Externals

Informed 4D Externals for 4th DIMENSION is a highly technical product designed for use by trained 4th DIMENSION programmers. This chapter assumes that you're an experienced 4th DIMENSION programmer and that you're familiar with Informed Designer and Informed Filler. Experience with Informed Number Server (now included with Informed Designer) is also helpful if you intend to use this product with your 4th DIMENSION application.

Overview

Informed 4D Externals contains a set of powerful 4th DIMENSION externals that provide seamless integration of Informed's forms processing capabilities with any 4th DIMENSION database. Based on the IAC (inter-applications communications) of the Mac OS, one of the included externals makes it easy to look up information from a 4th DIMENSION database while filling out forms with Informed Filler. Once a form has been completed, the information can be easily inserted—or submitted—directly into the 4th DIMENSION database, therefore eliminating the need to manually export and import data.

Also included is support for Informed Number Server. By making simple modifications to your 4th DIMENSION application, it too can obtain new form numbers from Informed Number Server. That way, users can fill out forms using either your 4th DIMENSION application or Informed Filler on the Mac OS and still maintain unique form numbering among all users.

With Informed Filler acting as the tool for filling out forms, users can avoid running 4th DIMENSION to do their work. For users that currently use your 4th DIMENSION application only for filling out forms, Informed Filler can be used in its place. Users benefit from the higher performance and reduced memory requirements of Informed Filler when compared to a 4th DIMENSION runtime.

Since Informed forms remain separate documents, changes to a form's design can be made easily without affecting your 4th DIMENSION application. Even major form revisions can be made without the need to recompile your application.

If your 4th DIMENSION application is currently used in a single-user environment, Informed can offer a practical way to expand your system to support multi-user data entry without the need to install additional 4th DIMENSION runtime applications. With one single-user 4th DIMENSION runtime available on a networked Mac OS compatible computer, multiple Informed Filler users can fill out and submit forms.

How it Works

Chapter 1, “Adding Intelligence to Your Forms”, explains how you can configure lookups, auto-incrementing cells, and form submission. Each of these features allows you to select different methods of accessing data sources. One such method is Apple events, an IAC (inter-application communications) capability available on any computer running version 7 or later of the Mac OS. The Informed 4D Externals rely on this method of communication.

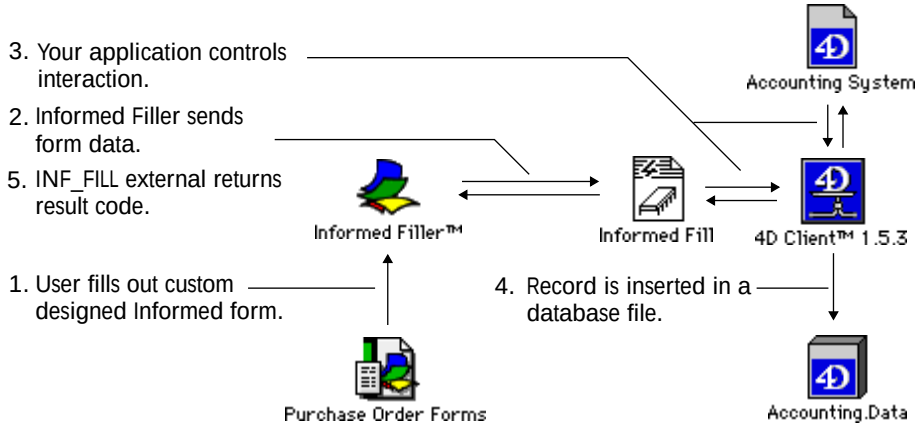
An Apple event is a message that one application sends to another. The message can contain information, or it can request that a certain command be performed. For example, when Informed Filler sends a completed form to a different application, it can send an Apple event containing the information on the form. To perform a lookup, an Apple event that requests a particular value can be sent instead. The other application performs a search and sends back a reply containing the information found.

Informed Designer, Informed Filler, and Informed Number Server use a particular set of Apple events to interact with other applications. Without the Informed 4D Externals, your 4th DIMENSION application cannot understand these Apple events. This product includes two 4th DIMENSION externals. The external named “INF_FILL” acts as an Apple event handler that can interpret and process any Apple events that are received from Informed Designer or Informed Filler. The external named “INF_NS_CLIENT” allows your 4th DIMENSION application to send Apple events to Informed Number Server.

Form Submission

Form submission normally marks the end of the forms process. Once a form is complete, its contents are accepted by an information system for further processing. The submission of a form often triggers other business procedures that process the information. For example, submission of an approved expense form may trigger the accounting procedure that issues a payment.

The INF_FILL external enables form submission into any 4th DIMENSION database. Custom form templates are designed with Informed Designer and linked to a file in the 4th DIMENSION database using Informed Designer’s Configure Submit command. Forms filled out with Informed Filler can then be submitted directly into the 4th DIMENSION database by choosing a single menu command. The following figure illustrates this process.



Linking a form template involves mapping each cell on the template to a corresponding field in a table of the 4th DIMENSION database. When a completed form is submitted using Informed Filler, the linking information ensures that each of the cell values on the template are entered in the correct fields in the 4th DIMENSION database file. Templates can be linked to both flat and relational files. For relational files, the external will automatically create the appropriate number of records in each of the related files.

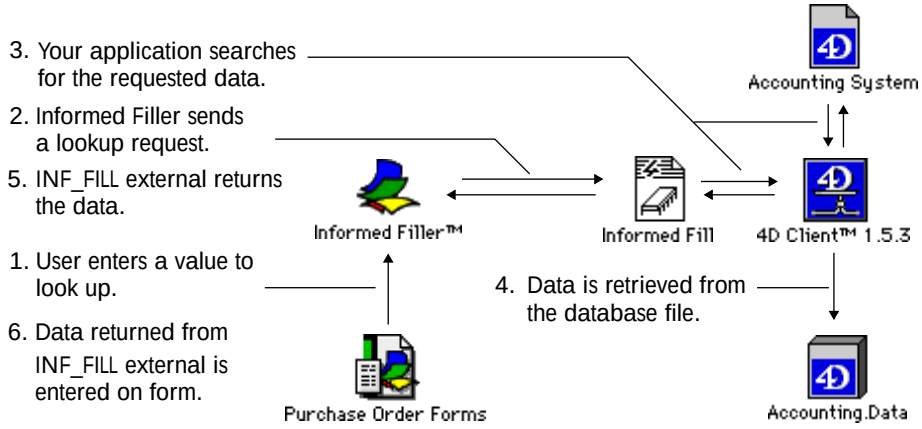
Since the link between Informed Filler and the INF_FILL external is real time, forms submission occurs interactively. When a form is submitted, your 4th DIMENSION application can examine and validate the data before it's accepted, and messages can be sent back to Informed Filler if errors are detected. That way, users are notified of mistakes immediately rather than later when it is often inconvenient and more costly to correct them.

For detailed information on configuring form submission, see "Form Submission" in Chapter 1, "Adding Intelligence to Your Forms."

Lookups

Forms often contain information that already exists electronically. Rather than retyping this information to fill out a form, Informed allows you to configure cells to lookup information in other forms or databases.

With the INF_FILL external, lookups can extract information from a 4th DIMENSION database. Using Informed Designer, you link the cells on a form that are to be looked up with fields in a file of your 4th DIMENSION database. When the user filling out a form enters a value in a particular cell, a lookup request is sent to the 4th DIMENSION application. Your application can search for and return the requested information to Informed Filler where it's automatically entered on the form.



Note The INF_FILL external is capable of returning multiple records to Informed Filler. If a lookup is performed and multiple matching records are found, Informed Filler will prompt the user with a scrolling list displaying all matches. A single record can then be selected by the user and entered on the form. This feature is helpful in situations where the user doesn't know in advance exactly which information is being looked up. The user may, for example, enter only the first few letters of a name to lookup a customer record.

For detailed information on configuring lookups, please see "Using Lookups" in Chapter 1, "Adding Intelligence to Your Forms."

Supporting Informed Number Server

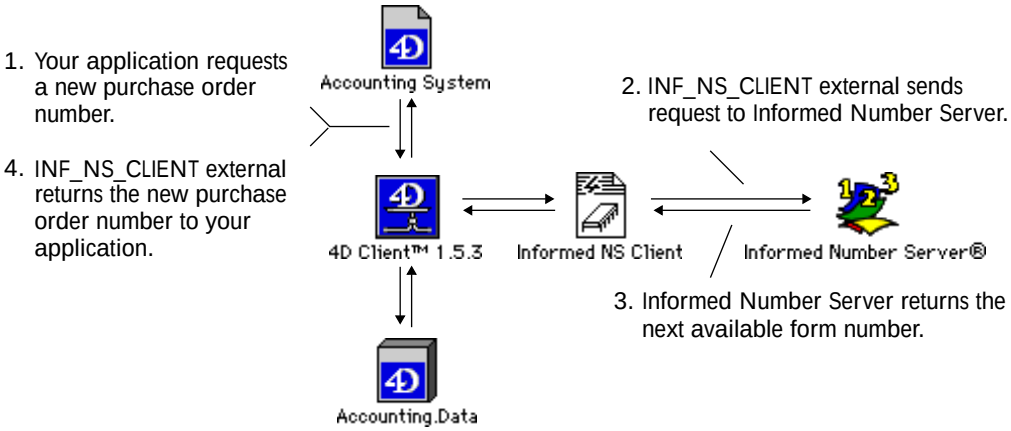
Unique form numbering is important for many types of forms. Form numbers provide a way to uniquely identify any form. Invoice numbers and purchase order numbers are examples of form numbers.

"Auto-incrementing Cells" in Chapter 1, explains how you can link an auto-incrementing cell to a variety of data sources. One such method of linking uses Apple events as a means of communication. This is the method used to link with Informed Number.

Informed Number Server was designed to automate the assignment of form numbers as different people fill out different forms. Each time a new form is filled out, Informed Filler sends an Apple event to Informed Number Server requesting a form number for the type of form being filled out. Informed Number Server replies with the next available form number.

In order to maintain unique consecutive numbering, form numbers must be obtained from a single source. Support for Informed Number Server in your 4th DIMENSION application is important if users are to fill out forms using both 4th DIMENSION and Informed Filler. Although Informed Filler may be your users' primary tool for filling out forms, you may have duplicated certain form designs in your 4th DIMENSION application as a convenience for those who use 4th DIMENSION

more often. The following figure illustrates how your 4th DIMENSION application interacts with Informed Number Server.



If your 4th DIMENSION application is already designed to assign unique form numbers to new forms, making the necessary modifications to take advantage of Informed Number Server is simple and straightforward. Rather than obtaining new form numbers the way you normally do, you'll change your application to request new numbers from Informed Number Server instead. This is done by making calls to the Informed NS Client external to specify where the Informed Number Server is running and the type of form for which numbers are requested.

System Requirements

The Informed 4D Externals for 4th DIMENSION depend on specific versions of system software and applications. They are:

- System software version 7.0 or greater
- Informed Designer version 1.3 or greater
- Informed Filler version 2.0 or greater (or Informed Manager version 1.3 or greater)
- 4th DIMENSION version 2.2.1 or greater
- CallProcs.Ext external (required for use with 4th DIMENSION 2.2)

The externals work with both versions 2.2 and 3.x of 4th DIMENSION. They also work with the 4D SERVER. Due to certain changes and enhancements in version 3.x and the 4D SERVER, simple changes must be made to your application and how it interacts with the INF_FILL external. These changes are explained in the following section, "Using the INF_FILL external."

Using the INF_FILL External

In order to protect data integrity, the INF_FILL external is built on an open architecture that gives you, the 4th DIMENSION developer, full control over how Informed interacts with your application. Once an Apple event has been received, you make calls to the external to process the event in a controlled manner. That way, you can reject invalid data, deny access to certain files, and ensure that the integrity of your 4th DIMENSION application and database is protected.

Certain minimal changes must be made to your 4th DIMENSION application in order to use it with Informed. This requirement is imposed intentionally to prevent unauthorized access to a 4th DIMENSION application and database. It prevents Informed users from obtaining access to a 4th DIMENSION database without the application developer enabling this capability. This chapter describes how you can modify your application to take advantage of form submission and lookups using Informed. An explanation of the INF_NS_CLIENT external can be found later in this chapter.

Using 4th DIMENSION Version 3.x and the 4D SERVER

The INF_FILL external works with both versions 2.2 and 3.x of 4th DIMENSION, as well as the 4D SERVER. With the exception of the mechanism used to poll the INF_FILL external's event queue, a 4th DIMENSION application interacts with the external in much the same way, regardless of which version of 4th DIMENSION you're using.

As explained in "Accepting Apple Events" later in this chapter, Informed Filler sends Apple events to your application whenever the user submits a form or triggers a lookup. Your application is expected to poll a queue and process any Apple events that are received. For 4th DIMENSION version 2.2, ACIUS's CallProcs.Ext external is required. This external allows you to configure 4th DIMENSION to repeatedly call a specified polling procedure during idle time.

With 4th DIMENSION version 3.x and the 4D SERVER, the new *processes* capability offers a more appropriate polling mechanism. A process is like a procedure that executes concurrently in its own 4th DIMENSION environment independent of other processes. Since each process has its own current selection and current record information, its operation doesn't interfere with the 4th DIMENSION application or any other processes that might be running.

For more information about processing Apple events, please see "Accepting Apple Events" and "Processing Apple Events" later in this chapter.

Installing the External

You install 4th DIMENSION externals using the 4D External Mover application provided with 4th DIMENSION. The INF_FILL external can be found in the file ‘Informed Fill.’

The INF_FILL external can be installed in any copy of your interpreted or compiled database application, or in your CallProc.Ext file.

Since the interaction between Informed Filler and your 4th DIMENSION database occurs through a preselected 4th DIMENSION development or runtime application, the INF_FILL external must be available to that application. When you select which 4th DIMENSION development or runtime application to link forms to, you should choose the one that can be running whenever Informed Filler users may be filling out forms.

Informed 4D Externals comes with two versions of a sample 4th DIMENSION database application (one for 4th DIMENSION version 2.2 and the other for version 3.x and the 4D SERVER), and an Informed template that’s already configured for lookups and form submission. You can use these items to experiment with the capabilities of the INF_FILL external.

Accepting Apple Events

When an Informed Filler user triggers a lookup or sends a completed form, an Apple event is sent to your 4th DIMENSION application. The external accepts the event and places it in a queue where it waits to be processed. Your 4th DIMENSION application is expected to periodically check for events in the queue by calling the procedure InfAECount. This procedure returns a single integer parameter indicating the number of events in the queue. Its declaration is shown below.

```
InfAECount ($theCount)
```

If an event is found in the queue, you then call the external procedure InfAERead to read the event. This procedure returns a parameter indicating the type of the event.

```
InfAERead ($theType)
```

For insert or submit events, the value of $theType will be 1 whereas lookup events are identified by the value 2. Other event types are processed internally and should be ignored. These are events that Informed Designer generates while you link a form or configure a lookup.

Once you’ve read an event and you know its type, you should then process the event in an appropriate manner. Event processing is explained in the section “Processing Apple Events” later in this chapter.

Apple events can be sent to your 4th DIMENSION application from multiple users connected to the same network. Each Apple event, however, is read and queued sequentially. As a result, your 4th DIMENSION application processes incoming Apple events one at a time, thereby avoiding potential problems caused by concurrent access by multiple users.

Polling the Apple Event Queue

In order to process Apple events, your application must continually poll the Apple event queue. If you're using 4th DIMENSION version 2.2, we recommend that you use ACIUS's CallProcs.Ext external. With the CallProcs.Ext external installed, 4th DIMENSION can be configured to call one or more global procedures during idle time. The call shown below configures 4th DIMENSION to automatically call the global procedure ProcessAllAEs once every 60 ticks (60 ticks = 1 second).

```
AddProc ("ProcessAllAEs"; 60)
```

If you're using version 3.x of 4th DIMENSION or the 4D SERVER, polling is accomplished by invoking a new process rather than using the CallProcs.Ext external. Since a process executes independently, processing Apple events using this mechanism ensures that your 4th DIMENSION application and other processes are not affected. The call shown below creates a new process named AEPollProcess. The value of 96000 should be adequate for most applications.

```
pid := New Process ("AEPollProcess";96000;"CallProAllAEs")
```

The process' procedure should be written to repeatedly check for and process any Apple events in the event queue, or call another procedure that does so. This is essentially what the CallProcs external accomplishes for applications based on version 2.2 of 4th DIMENSION. The process procedure shown below uses an endless repeat loop to call the ProcessAllAEs procedure. The 'Delay Process' call is necessary so that 4th DIMENSION allows other processes time to execute. 60 Ticks is adequate for most applications.

```
PROCEDURE AEPollProcess

While (True)
    ProcessAllAEs
    Delay Process (current process;60)
End while
```

The ProcessAllAEs procedure should be written to check for Apple events waiting in the event queue and process those that are found. Below is an example procedure that's provided with the Informed 4D Externals. This procedure can be used unmodified in your 4th DIMENSION application.

```
PROCEDURE ProcessAllAEs

C_INTEGER ($theCount)
C_INTEGER ($theType)

InfAECount ($theCount)
While (Not ($theCount = 0))
    InfAERead ($theType)
    Case of
     :($theType = 1)
      ProcessInsert
    :($theType = 2)
      ProcessLookup
    End case
    InfAECount ($theCount)
End while
```

In addition to the above procedure, you must also write the procedures ProcessInsert and ProcessLookup. These procedures should be written to process incoming Apple events in a manner that's appropriate for your application. Information about processing Apple events can be found later in this chapter.

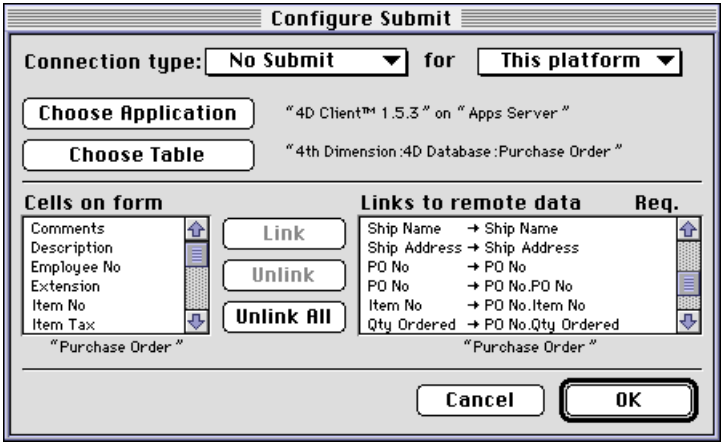
Linking Forms

Forms are linked to your 4th DIMENSION application using Informed Designer. Linking is required both to configure lookups and to link forms for form submission.

The linking process involves mapping cells on the Informed form to fields in a file of a 4th DIMENSION database. For detailed information on Informed Designer's Lookup and Configure Submit commands, see Chapter 1, "Adding Intelligence to Your Forms." The issues related specifically to Informed and 4th DIMENSION are discussed here.

Required and Unique Fields

When you link a form to a file in your 4th DIMENSION database, the Configure Submit dialog box displays a list of cells on the form and a list of the fields in the database file.



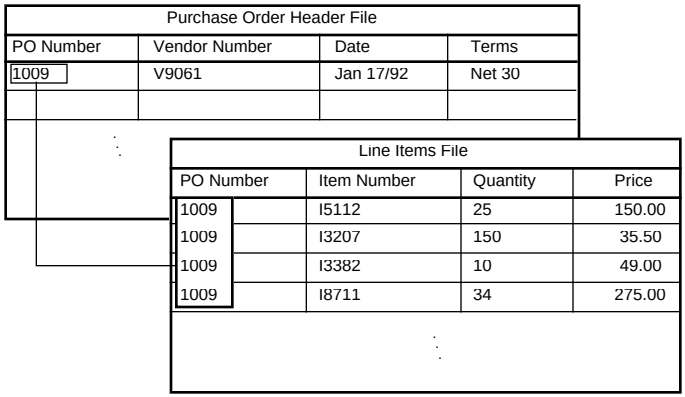
The 'Req.' column next to the list of database fields is intended to indicate which fields require values when a completed form is submitted. This column does not reflect 4th DIMENSION's required field attribute of each field.

You should not rely on 4th DIMENSION's required and unique field attributes to prevent the entry of blank or duplicate values. Instead, you should check for blank or duplicate values when you examine the contents of a form for errors. See "Validating Data" later in this chapter for more information.

Relational Files

Lookups can retrieve information from a single file in your 4th DIMENSION database. A form, however, can be configured for submission to multiple related files. The submission of a single form can result in the creation of records in each of the files.

Related files are often used to reduce the amount of redundant data stored in a database. For example, a purchase order may be stored in two files, one containing header information and the other containing line item information. Each purchase order would consist of one header record and multiple line item records, one for each item ordered. By dividing the purchase order into two files, the header information (that is, the purchase order number, date, terms, an so on) is stored only once rather than redundantly with each line item record. Using 4th DIMENSION terminology, the header file is called the ‘one file.’ The line item file is called the ‘many file.’



The 4th DIMENSION programming interface does not allow an external to determine the relations that exist in the structure of your application. In order to make this information available to the INF_FILL external, your application should call the external procedure InfAddRelation once for each pair of related files on startup.

```
InfAddRelation ($oneFile; $relatedField; $manyFile)
```

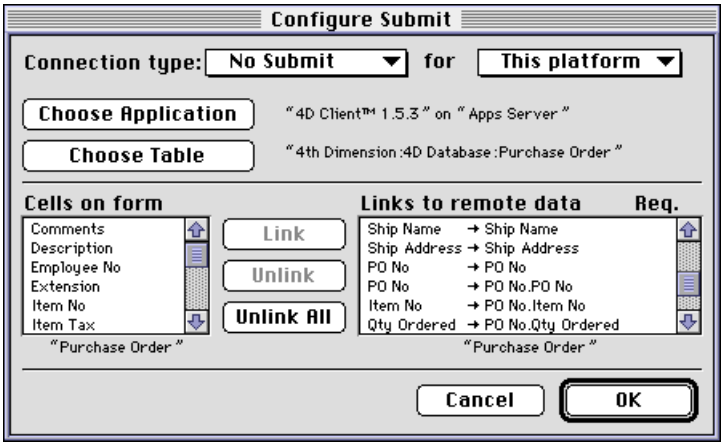
$oneFile is the file number that identifies the one file (the header file in the purchase order example). $manyFile is the file number of the related file. $relatedField is the field number of the field in the one file that binds the related records in the two related files. In the purchase order example, the related field would be the purchase order number field. This field is stored in both the header file and with each line item record. The call shown below would correctly describe the related files.

```
InfAddRelation (File (»[PO Header]); Field (»[PO Header]PO Number); File (»[Line
Items]))
```

By calling InfAddRelation to describe related files, Informed Designer will allow you to link a single form to multiple related files.

Note Your application must call InfAddRelation before any Apple events are accepted. This initialization should occur in your application's STARTUP procedure. After your application processes the first Apple event, calls to InfAddRelation will be ignored.

When you select a file in your 4th DIMENSION database to submit a form to, Informed Designer will list all fields of both the selected file as well as any related files on the Configure Submit dialog box. The field names of the fields contained in a related many file will be prefixed by the name of the related field.



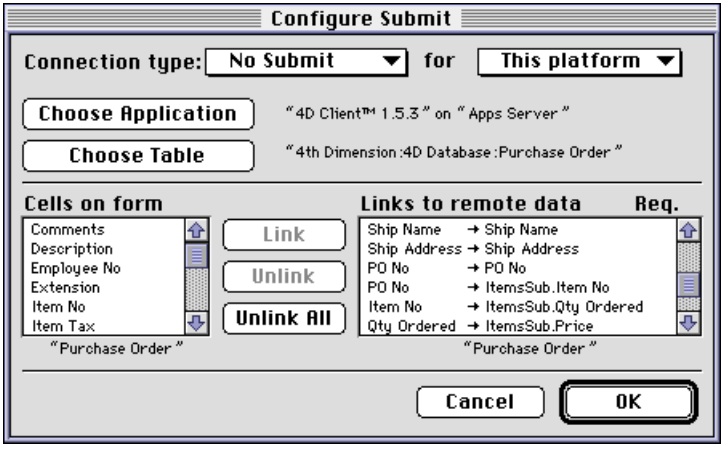
Fields in a many file should be linked to multi-value column cells on your form. For example, fields such as the item number, quantity, and price in the line items file of a purchase order should be linked to the corresponding column cells on your purchase order form. When the Informed Filler user submits a completed purchase order form, the INF_FILL external will automatically insert one line item record for each value in the linked column cells. If you link a single-value cell to a field in a many file, the cell's value will be stored in each of the multiple records created.

Subfields

4th DIMENSION offers an alternate method for storing relational data. In addition to standard data types such as text, integer, and date, a single field can store multiple *subfile* records. Each subfile contains one or more fields called *subfields*. With subfiles, relational data can be stored in a single file. A purchase order, for example, could be stored in a single file containing fields for header information and a subfile for the line items.

Note The INF_FILL external allows you to insert records into files containing subfiles. You cannot, however, lookup information that's stored in subfiles.

When you configure submission to a file containing one or more subfiles, each subfield can be linked to an individual cell on your form. On the Configure Submit dialog box, Informed Designer will prefix each subfield with the name of its containing subfile.



Subfields should be linked to multi-value column cells on your form. When Informed Filler sends a completed form to your 4th DIMENSION application, the INF_FILL external will automatically create a new subfile record for each set—or row—of values in the column cells that are linked to the subfields. If you link a single-value cell to a subfield, the cell's value will be stored in each of the multiple subfile records created.

Data Types

Informed supports a rich set of data types and formatting options. The data types include text, character, number, name, date, time, boolean, picture, and signature. Although 4th DIMENSION supports most of these data types, its formatting capabilities are not as robust. For example, 4th DIMENSION does not allow you to enter a date such as 'Monday, June 24, 1996,' whereas Informed will accept this format.

Cells that are linked to fields in your 4th DIMENSION database must be formatted according to the constraints of 4th DIMENSION. The following table lists each Informed data type along with the compatible 4th DIMENSION data type and any formatting constraints.

Informed and 4th DIMENSION data types

Informed Data Type	4th DIMENSION Data Type	Required Format
Text	Text	No formatting constraints
Character	Alpha	Format characters must be stripped
Name	Text	No formatting constraints
Number	Number	No formatting constraints
Date	Date	MM/DD/YY
Time	Time	HH:MM:SS
Boolean	Boolean	No formatting constraints
Picture	Picture	No formatting constraints
Signature	Picture	No formatting constraints

If you want to use a cell format that's not supported by 4th DIMENSION, you can work around this limitation by creating an additional cell with an acceptable format. This additional cell would be calculated to be equal to the original cell and linked to the 4th DIMENSION field in its place. You might place this cell on the work page.

Like Informed, 4th DIMENSION supports character formatting for telephone numbers and other fixed format text values. Format characters such as dashes or parentheses are automatically entered for you. With 4th DIMENSION, format characters must be omitted when you enter a value. When Informed Filler sends a value to the INF_FILL external, format characters are included. Therefore, a telephone number such as '(403) 463-3330' might be interpreted incorrectly by 4th DIMENSION as '((40) 3) -463-' if the value is displayed using a telephone output formatter. As explained above, you can work around this incompatibility by creating an additional cell and calculating its value as the formatted value with the format characters removed.

Processing Apple Events

Once an Apple event has been accepted, your application should then call a procedure to process the event. Processing an event involves either inserting data in the case of an insert event, or searching for data in the case of a lookup event. Processing is completed by unloading the current records of all affected files and sending a reply back to Informed Filler indicating either that the event was processed successfully, or that an error occurred.

Processing Insert Events

When the Informed Filler user completes and submits a form, an Apple event containing the form information is sent to your 4th DIMENSION application according to the linking configuration of the form. Your application is expected to read the Apple event, insert or reject the data, then send a reply back to Informed Filler.

Depending on the nature of the form, you may or may not want to validate the data before accepting it. For example, you may want to check sensitive accounting data on financial forms to ensure that accounting errors are not made. You may also want to control more tightly the integrity of your 4th

DIMENSION application by saving and restoring the current selection of any affected files, or by completing a transaction through additional entries in the database. For example, processing an invoice form may involve adjusting the customer's balance and the inventory quantities.

Determining the Form Type

Once your application has read an insert event by calling InfAERead (see "Accepting Apple Events" earlier in this chapter), the data to be inserted is held in a special buffer maintained internally by the external. At this point you can determine which file or files the data is being inserted into by calling the two external procedures shown below.

```
InfNewRecCount ($theCount)
InfNewRecInfo ($theIndex; $theFileNum; $theRecNum)
```

InfNewRecCount returns an integer value in $theCount indicating the number of new records that will be created. For forms that map to a single file in your 4th DIMENSION database, $theCount will return the value 1. For forms that are linked to related files, $theCount may return a value greater than 1. For example, suppose that your 4th DIMENSION application stores information for a single invoice in two related files, one containing the invoice header fields (that is, the invoice number, date, terms, and so on), and the other containing line item information, one record for each item. If an invoice with three line items were sent to your application, the value of $theCount would be 4, indicating one header record and three line item records. (For more information about linking forms to related files, see "Relational Files" earlier in this chapter.)

To find out the file or files in which new records will be created, call the external procedure InfNewRecInfo. The value of $theIndex should be between 1 and the number of records to be created. In $theFileNum, InfNewRecInfo will return the file number of the file in which the record will be inserted. The value of $theRecNum will be -1 and should be ignored at this point.

Controlling Access to Files

By knowing which file a record is being inserted into before the record is inserted, you can programmatically prevent the creation of data in particular files of your database. For example, suppose that of the ten files in your 4th DIMENSION database, you're allowing remote submission into only two files. After determining which file or files records will be inserted into, you can proceed with processing the event only if insertion into that file is allowed.

Below is an example procedure that processes insert Apple events. In this example, remote forms submission is allowed only for the files identified by file numbers 4 and 5.

```
PROCEDURE ProcessInsert

C_BOOLEAN ($allowInsert)
C_INTEGER ($theCount)
C_INTEGER ($theFileNum)
C_LONGINT ($theRecNum)

InfNewRecCount ($theCount)
$allowInsert := True
While ($theCount # 0)
```

```
    InfNewRecInfo ($theCount, $theFileNum, $theRecNum)
    If (Not (($theFileNum = 4) | ($theFileNum = 5)))
        $allowInsert := False
        InfErrorMsg ("You cannot submit that type of form.")
        $theCount := 1
    End if
    $theCount := $theCount - 1
End while

If $allowInsert
    InfDoInsert
    UnloadRecords
End if
InfSendReply
```

This procedure calls InfNewRecInfo for each record about to be inserted to determine which files will be affected. If the file number is determined to be either 4 or 5 for at least one of the records, the insert event is not processed.

If you're not concerned about preventing access to certain files, much of the code in the previous example is unnecessary. Your ProcessInsert procedure could, therefore, be as simple as the following procedure.

```
PROCEDURE ProcessInsert

InfDoInsert
UnloadRecords
InfSendReply
```

The above procedure inserts the data from the sent form and sends a confirmation reply back to Informed Filler. The procedure doesn't check which file data is being inserted into, nor does it check the inserted data for errors.

After calling InfDoInsert (see "Inserting the Data"), the procedure UnloadRecords is called to unload the current records of any affected files. Unloading records is the responsibility of your application. See "Unloading Current Records" later in this section for more information.

Inserting the Data

The information from the submitted form remains in a special buffer until your application calls the external procedure InfDoInsert. This procedure creates the necessary record or records in the database files and fills their contents with the form information. The following example demonstrates the use of the InfDoInsert external procedure.

```
PROCEDURE ProcessInsert

If (FileAllowed)
    InfDoInsert
    ValidateData
    UnloadRecords
Else
    InfErrorMsg ("You cannot submit that type of form.")
End if
InfSendReply
```

The function FileAllowed is assumed to return false if insertion into the requested file is not allowed. After calling InfDoInsert, you can then call InfNewRecInfo to obtain the record numbers of the records inserted. This allows you to examine the contents of the inserted records in order to check for errors (see “Validating Data” for more information).

Validating Data

Data validation is often critical in preventing errors from being accepted. After calling InfDoInsert, your application can examine the contents of the new record (or records) by first calling the external procedures InfNewRecCount and InfNewRecInfo. InfNewRecCount returns the number of new records created and InfNewRecInfo returns the record numbers of the new records. By calling 4th DIMENSION’s GOTO RECORD procedure, you can examine the data for each new record, then either proceed normally or return an appropriate error message. The following example shows a function that checks new purchase orders for blank or duplicate purchase order numbers.

```
FUNCTION Validate_PO

C_BOOLEAN ($theResult)
C_STRING (20;$thePONum)
C_INTEGER ($theFileNum)
C_LONGINT ($theRecNum)
C_INTEGER ($theCount)

$theResult := True
InfNewRecInfo (1; $theFileNum; $theRecNum)

GOTO RECORD ([PO Header]; $theRecNum)
$thePONum := [PO Header]PO No

` Check for blank purchase order number
If ($thePONum = "")
    $theResult := False
    InfErrorMsg ("You must enter the purchase order number.")
Else
    ` Check for duplicate purchase order number
    ALL RECORDS ([PO Header])
    SEARCH ([PO Header];[PO Header]PO No; =;$thePONum)
    If (Records in selection ([PO Header]) > 1)
        $theResult := False
        InfErrorMsg ("A purchase order with that number already exists")
    End if
End if
```

```
` If an error was detected, delete the inserted records
If (Not ($theResult))
    InfNewRecCount ($theCount)
    While (Not ($theCount = 0))
        InfNewRecInfo ($theCount, $theFileNum, $theRecNum)
        GOTO RECORD (File ($theFileNum)»; $theRecNum)
        DELETE RECORD (File ($theFileNum)»)
        $theCount := $theCount - 1
    End while
End if

$0 := $theResult
```

This example assumes that purchase orders are stored in two files, one containing the header information, and the other containing one record for each item ordered. The name of the purchase order header file is ‘PO Header’ and the field in this file containing the purchase order number is named ‘PO No.’ For information about error reporting, please see “Sending Errors Back to Informed Filler” later in this chapter.

For forms that are mapped to multiple related files, you can assume that calling InfNewRecInfo with a value of 1 for $theIndex will return the file and record numbers of the single record inserted into the ‘one file.’ In the example shown above, there’s no need to call InfNewRecCount before the first call to InfNewRecInfo since we know that passing 1 in $theIndex will return the record number for the record inserted into the purchase order header file. For more information about how forms are mapped to related files, see “Relational Files” earlier in this chapter.

Unloading Current Records

After calling InfDoInsert, your application is expected to unload the current records for any files that were affected by the insert. Unloading these records ensures that they’re unlocked for use in other transactions. The procedure shown below is written generically and can be used in your application without modification.

```
PROCEDURE UnloadRecords

C_INTEGER ($theCount;$theFileNum;$theRecNum)

InfNewRecCount ($theCount)
While ($theCount # 0)
    InfNewRecInfo ($theCount, $theFileNum, $theRecNum)
    UNLOAD RECORD (File ($theFileNum)»)
    $theCount := $theCount - 1
End while
```

If your application is aware of which files records were inserted into, you can avoid calling InfNewRecCount and InfNewRecInfo and simply call UNLOAD RECORD with the appropriate file number.

Using Transactions

If processing the insertion of a single form involves making several entries in the database, you may want to start a 4th DIMENSION transaction before calling InfDoInsert. By starting a transaction, you can more easily roll back submission of the form if an error is detected.

Suppose that processing an invoice involves adjusting inventory quantities and customer balances in addition to inserting the invoice header and line item records. After inserting the new records, your application may check for errors such as a duplicate invoice number or insufficient customer credit. If an error is detected, the new records must be deleted and any other changes to the database reverted. If your application starts a transaction before processing the event, rolling back or cancelling can be done simply by calling the 4th DIMENSION procedure CANCEL TRANSACTION. Below is an example.

```
PROCEDURE ProcessInsert

C_BOOLEAN ($dataOK)

If (FileAllowed)
    START TRANSACTION (*)
    InfDoInsert
    ValidateData ($dataOK)
    If $dataOK = True
        VALIDATE TRANSACTION
    Else
        CANCEL TRANSACTION
    End if
Else
    InfErrorMsg ("You cannot submit that type of form.")
End if
UnloadRecords
InfSendReply
```

Even for simple forms, the use of transactions is highly recommended if your 4th DIMENSION application is used in a multi-user environment. Using transactions will ensure that the changes made to a database as a result of processing a submitted form will not be available to other users until processing has completed and the transaction has been validated. However, if you're using version 2.2 of 4th DIMENSION, care must be taken to ensure that the processing of Apple events doesn't interfere with other transactions that may be invoked directly by your application. For more information, see "Coordinating Interaction" later in this chapter.

Processing Lookup Events

A lookup is triggered when the Informed Filler user types a value in a lookup cell. The value typed is looked up either in another form or by another application, and the information retrieved is entered on the form.

If a lookup is linked to your 4th DIMENSION application, Informed Filler will send it an Apple event when the lookup is triggered. After accepting the event (see "Accepting Apple Events" earlier in this chapter), your application is expected to search for the requested record and make it the cur-

rent selection of the appropriate file. Later when you send a reply back to Informed Filler, the external will transfer the contents of the current selection. Informed Filler will then enter the field values from your 4th DIMENSION database in the appropriate cells on the form.

Performing the Lookup

Once you've read an Apple event (via InfAERead) and determined it to be a lookup event, your application should call the external procedure InfDoLookup to perform the lookup. This procedure takes a single string parameter.

```
InfDoLookup ($theSearchProcedure)
```

The value of $theSearchProcedure should be the name of a global procedure in your application which performs the search and leaves the data found in the current selection of the appropriate file. InfDoLookup passes four parameters to the search procedure. They are:

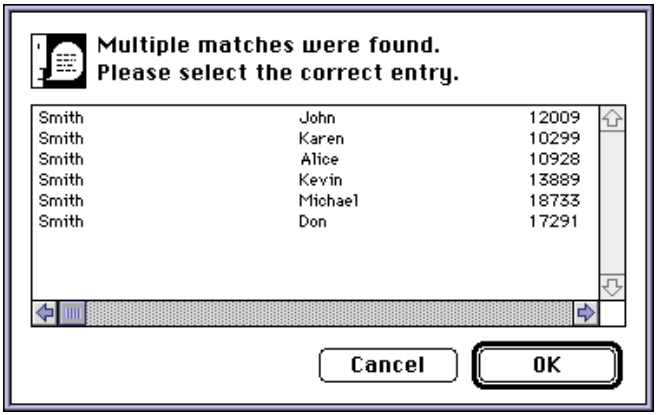
`$1:` the file number of the file for which the lookup has been requested
`$2:` the field number of the field on which the lookup is based
`$3:` the match option of the lookup as configured by the form designer
1 = do not lookup; 2 = use next value
`$4:` the value to be searched for as entered by the Informed Filler user

Suppose, for example, that an invoice form were configured to look up the part number and price of an item when the Informed Filler user enters the item's description. InfDoLookup would call your search procedure passing the file number of the inventory file in $1, the field number of the description field in $2, the lookup match option in $3, and in $4, the value of the description entered by the Informed Filler user. The following figure shows a simple search procedure called 'MySearch' as well as a procedure which your application might call to process lookup events

```
PROCEDURE MySearch

C_INTEGER ($1;$2;$3)
C_STRING (80;$4)

ALL RECORDS (File ($1)»)
SEARCH (File ($1)»; Field ($1; $2)» = $4)


PROCEDURE ProcessLookup

InfDoLookup ("MySearch")
InfSendReply
```

The search procedure searches for an exact match of the search value. If the search fails, the current selection of the file will be left empty. If one or more records are found, they will be available in the file's current selection following the call to SEARCH. Their field values will be included with the reply that's sent back to Informed Filler when your application calls InfSendReply.

If your search procedure returns a single record selection, Informed Filler will enter the field values of that record in the appropriate cells on the form. If the selection contains more than one record, a dialog box will display allowing the Informed Filler user to browse through the list of records and select the correct one. The browsing dialog box is shown below.



Lookup browsing is useful when there are multiple records containing the requested search value. For example, suppose that an invoice form is configured to lookup customer information when the customer's surname is entered on the form. If there are two or more customers with the same surname, Informed Filler will display those customers in the lookup browsing dialog box. The user can select the correct customer based on related customer information then click 'OK' to enter the information on the form.

The number of records that the external can return to Informed Filler is limited by the maximum size of an Apple event. The size of the combined records cannot exceed 20K. You can further limit the number of records that can be returned to Informed Filler by calling the external procedure InfMaxRecords. By default, the limit is set to 50 records.

```
InfMaxRecords ($numRecords)
```

This setting should be based on network performance and cost. The slower the network, the longer it takes to transfer the data from your 4th DIMENSION application to Informed Filler. Users should be encouraged to enter complete values before triggering lookups. That way, the number of matching records is kept to a minimum.

Match Options and Wildcards

When a lookup is configured, the form designer chooses one of two match options. The match option determines what happens if an exact match is not found when the lookup is performed.

When InfDoLookup calls your 4th DIMENSION search procedure, it passes the lookup's match option as the third parameter according to the values shown below.

1 = Do not lookup
2 = Use next value

You can control the interpretation of the match option by changing the lookup search procedure, or you can ignore the match option entirely if it's not relevant to your application. The sample search procedure below returns the record containing the next higher search value if an exact match is not found.

```
PROCEDURE MySearch

C_LONGINT ($recCount)
C_INTEGER ($1;$2;$3)
C_STRING (80;$4)

ALL RECORDS (File ($1)»)
SEARCH (File ($1)»; Field ($1; $2)»; "="; $4)
SORT SELECTION (File ($1)»; Field ($1; $2)»; ">")
$recCount := Records in selection (File ($1)»)
If ($recCount = 0) And ($3 = 2))
    ALL RECORDS (File ($1)»)
    SEARCH (File ($1)»; Field ($1; $2)» >= $3)
    $recCount := Records in selection (File ($1)»)
    If ($recCount > 0)
        SORT SELECTION (File ($1)»; Field ($1; $2)»; ">")
        ` Make the first record the current record.
        FIRST RECORD (File ($1)»)
        ` Remove other records from the current selection.
        ONE RECORD SELECT (File ($1))
    End if
End if
```

4th DIMENSION allows you to use wildcard symbols for partial matches. The Informed Filler user could, for example, find all customers whose surname starts with 'Smi' by entering the search value 'Smi@.' This feature is useful if the user doesn't know the exact spelling of the value being looked up.

Unloading Current Records

If your 4th DIMENSION application is used in a multi-user environment, you should be sure to unload the current records of any files that were affected by a lookup. Records are unloaded by calling the 4th DIMENSION procedure UNLOAD RECORD. If you do not unload the current records after performing a lookup, users of your application may encounter 'File in use' errors.

Sending Errors Back to Informed Filler

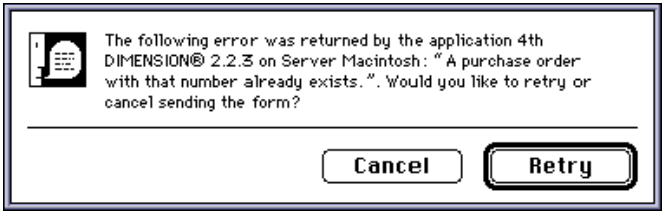
Once an Apple event has been processed, your application is expected to send a reply back to Informed Filler by calling the external procedure InfSendReply. Sending a reply confirms that processing has completed.

Your application can call one of two external procedures to report errors back to the Informed Filler user. Before calling InfSendReply, simply call either InfErrorCode or InfErrorMsg to set the error code or message, respectively.

```
InfErrorCode ($theCode)
InfErrorMsg ($theMessage)
```

The value of $theCode is an integer and is defined by your application. The value of $theMessage is a string that allows for a maximum length of 255 characters.

Each time an Apple event is accepted, the INF_FILL external automatically clears the error code and message. If an error is detected while processing an insert event, your application should call either InfErrorCode or InfErrorMsg before calling InfSendReply. The dialog box that Informed Filler displays when your application calls InfSendReply will show the error code or message.



In practice, error codes generally do not provide sufficient information for the Informed Filler user to understand a problem and, therefore, are best suited for debugging purposes.

Coordinating Interaction

With version 2.2 of 4th DIMENSION, the interaction between Informed Filler and the 4th DIMENSION database occurs through your 4th DIMENSION client application. Care must be taken to prevent the processing of Apple events from interfering with the application's normal use. With version 3.x of 4th DIMENSION and the 4D SERVER, the INF_FILL external interacts directly with an independent process, therefore eliminating any possible interference with your client application.

If your v2.2 4th DIMENSION application is used in a multi-user environment, you can avoid the concern of coordinating Apple event processing altogether. Instead of relying on a user's copy of 4th DIMENSION for processing insert and lookup events, you could link forms to an additional 4th DIMENSION runtime that's dedicated for this purpose.

The precautions described in the following sections apply only if you're using version 2.2 of 4th DIMENSION and only if forms are linked to a 4th DIMENSION development application or runtime which is also being shared with user.

Current Selection and Current Record

Since processing a lookup event changes the file's current record and current selection, you may want to save these attributes before searching and restore them afterwards. If your application maintains global variables for attributes such as the number of records in a file, you may have to update these variables as well.
Certain 4th DIMENSION actions such as printing a report or editing a layout rely on a consistent selection. Calling commands that change the current selection during these operations can generate errors. Your application, therefore, must ensure that the processing of Apple events doesn't interfere with commands that may be invoked due to user actions.

Transactions

If you're using 4th DIMENSION version 2.2.x, and the procedures that you've written to process Apple events use transactions, additional precautions must be taken to avoid potential conflicts. By definition, 4th DIMENSION allows no more than one active transaction per user. Your application, therefore, must delay the processing of an Apple event that modifies the database if a transaction is already active. If processing the Apple event relies on transactions, an error will occur when the transaction is initiated if a another transaction is already active. If processing occurs without the use of transactions, the database entries made as a result of processing the Apple event could be lost if an active transaction is later cancelled.

Quick Start

The open architecture of the INF_FILL external provides control and flexibility to your 4th DIMENSION application. As explained earlier, you can write global procedures to validate data, control access to the different files in your database, and customize the processing that occurs when completed forms are submitted or lookups requested. If these issues are not important to your application, the modifications necessary to enable forms submission and lookups are simple and straightforward.

To enable forms submission and lookups with Informed, simply add the global procedures listed below to your application. These procedures can be found in the text file named "Fill Ext. Procs.TEXT".

Procedure: STARTUP

This code should be added to your application's STARTUP procedure. For 4th DIMENSION v2.2, it calls AddProc to ensure that the global procedure ProcessAllAEs is called once every 60 ticks during idle time. For 4th DIMENSION v3.x or the 4D SERVER, the procedure invokes a new process which continually calls ProcessAllAEs. The global variable ErrorFound is also initialized to False.

For 4th DIMENSION version 2.2:

```
AddProc ("ProcessAllAEs";60)
```

For 4th DIMENSION version 3.x or the 4D SERVER:

```
C_LONGINT (pid)
pid := New Process ("AEPollProcess";96000;"CallProAllAEs")
```

Procedure: AEPollProcess

This procedure is required only if you're using version 3.x of 4th DIMENSION or the 4D SERVER.

```
While (True)
    ProcessAllAEs
    Delay Process (current process;60)
End while
```

Procedure: ProcessAllAEs

This procedure is called continually to process any Apple events in the queue by calling either ProcessInsert or ProcessLookup repeatedly until the queue is empty.

```
C_INTEGER ($I;$theType)
InfAECount ($I)
While (Not($I=0))
    InfAERead ($theType)
    Case of
    : ($theType=1)
        ProcessInsert
    : ($theType=2)
        ProcessLookup
    End case
    InfAECount ($I)
End while
```

Procedure: ProcessInsert

This procedure is called to process an insert Apple event. The call to PreInsert is intended to start a transaction and ensure that access to the file or files in which records are being inserted is allowed.

The call to ValidateIns is intended to examine that inserted data for any errors. If no errors are detected, ErrorFound is cleared to False.

```
C_BOOLEAN (ErrorFound)
ErrorFound:=True
If (PreInsert)
    InfDoInsert
    If (ValidateIns)
        ErrorFound:=False
    End if
End if
PostInsert
UnloadRecords
InfSendReply
```

Function: **PreInsert**

This function is called by ProcessInsert before any records are created. A transaction is initiated and the function returns True. If you want to deny access to certain files, you should insert code here that calls InfNewRecCount and InfNewRecInfo to determine which file or files records will be created in. To prevent the insert from occurring, return False instead of True.

```
START TRANSACTION (*)
$0:=True
```

Function: **ValidateIns**

This function is called by ProcessInsert. It is intended to validate the inserted data. If an error is detected, you should return a False result.

```
$0:=True
```

Procedure: **PostInsert**

This procedure is called by ProcessInsert after the data has been inserted. If the insert event was processed successfully, the transaction is validated. Otherwise, the transaction is cancelled.

```
If (ErrorFound)
    CANCEL TRANSACTION
Else
    VALIDATE TRANSACTION
End if
```

Procedure: **ProcessLookup**

This procedure is called by ProcessAllAEs to process a lookup event.

```
InfDoLookup ("DoSearch");
InfSendReply
```

Procedure: **DoSearch**

This procedure is called by InfDoLookup. Its parameters specify the file for which the lookup has been requested ($1), the field in this file to search ($2), the lookup's match option ($3), and the search value ($4). It performs the search and leaves the results in the file's current selection. If you want to deny access to certain files, insert code that examines the file number parameter ($1) and calls InfErrorMsg or InfErrorCode accordingly.

```
C_INTEGER ($1;$2;$3)
C_STRING (80;$4)

ALL RECORDS(File($1)»)
SEARCH(File($1)»;Field($1;$2)»=$4)
```

Procedure: **UnloadRecords**

```
C_INTEGER ($theCount;$theFileNum;$theRecNum)

InfNewRecCount ($theCount)
While ($theCount # 0)
    InfNewRecInfo ($theCount, $theFileNum, $theRecNum)
    UNLOAD RECORD (File ($theFileNum)»)
    $theCount := $theCount - 1
End while
```

Using the INF_NS_Client External

Informed Number Server is an application that generates unique form numbers at the request of other applications. A single Informed Number Server application can be configured to generate form numbers for several different types of forms.

Using Informed Designer, you can link a 'form number' cell on a form to the Number Server application. When the Informed Filler user adds a new form or manually requests a new number, Informed Filler sends a message to the Number Server application requesting the next available form number. The Number Server application replies with the next available number.

If users fill out forms using both Informed Filler and your 4th DIMENSION application, you'll need to modify your application so that it too obtains new form numbers from the Number Server application. The INF_NS_CLIENT external contains four external functions that your application can call to select a Number Server application, determine which form numbers are available, and request a new form number.

Installing the External

Like any 4th DIMENSION external, you install the INF_NS_CLIENT external using the 4D External Mover application provided with 4th DIMENSION. The INF_NS_CLIENT external can be found in the file 'Informed NS Client.'

The INF_NS_CLIENT external can be installed in any copy of your interpreted or compiled 4th DIMENSION application, in any copy of a 4th DIMENSION development or runtime application, or in your Proc.Ext file.

Modifying Your 4th DIMENSION Application

Informed Number Server uses Apple events, an IAC (inter-application communications) capability of the Mac OS (version 7 or later) to communicate with other applications. In addition to the computer on which the Number Server application is running, each computer running your 4th DIMENSION application must also be using system software version 7.0 or later.
Integrating your 4th DIMENSION application with Informed Number Server is a simple and straightforward exercise. Rather than generating new form numbers from within your application, a new number is obtained by calling a function in the INF_NS_CLIENT external. The external sends an Apple event message over the network to the Number Server application. The Number Server application replies with the next available form number. Other external functions allow your application to select the Number Server application and find out which form numbers are available.

Identifying the Number Server Application

When your application requests a new form number, information that identifies the computer on which the Number Server application is running must be provided. This information includes the name of the Number Server application, the name of the computer on which it's running, and the name of the network zone to which the computer is connected.

Although you may already know the names of these items, your application can obtain them by calling the external function ChooseNS.

```
ChooseNS ($theAppName, $theMacName, $theZoneName)
```

ChooseNS displays the Program Linking dialog box. This dialog box contains three scrolling lists from which you can select the correct network zone, computer, and Number Server application.

When you click 'OK,' ChooseNS will return the names of the selected items in the three string parameters $theAppName, $theMacName, and $theZoneName. These names are then passed to the external function GetNSNextValue to obtain a new form number (see "Requesting a New Form Number" later). If your network contains only a single zone, the Program Linking dialog box will contain two lists instead of three. The zone name of a single-zone network is always the asterisk character ('*').

ChooseNS is an external function that returns a long integer result. A result of zero indicates that a Number Server application was successfully selected. Non-zero results correspond to the various errors that can occur. For information about error handling, please see “Error Codes” later in this chapter.

Determining the Available Form Numbers

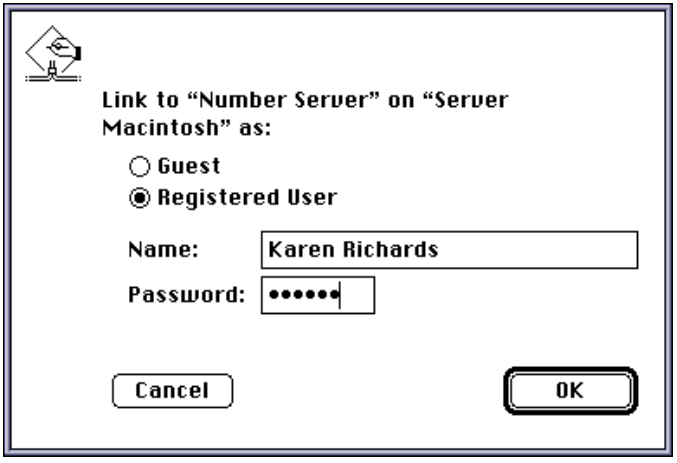
A single Number Server application can generate form numbers for several different types of forms. Form numbers are configured using the Number Server application. Each form number has a name (‘invoice number,’ or ‘purchase order number,’ for example). When your application requests a new form number, the name of the form number must be provided.

To obtain the names of the form numbers available from a particular Number Server application, your 4th DIMENSION application can call the external functions GetNSNameList and GetNSName.

```
GetNSNameList ($theAppName, $theMacName, $theZoneName, $theCount)

GetNSName ($theIndex, $theName)
```

GetNSNameList connects to the Number Server application identified by the string parameters $theAppName, $theMacName, and $theZoneName, then requests the names of all available form numbers. If your network consists of a single zone only, pass the asterisk character (‘*’) in $theZoneName. Otherwise, pass the zone name. GetNSNameList will return the number of names in the list in the integer parameter $theCount. The list of names itself is held in the external’s memory area.

If the Number Server application is not running on the same computer as your 4th DIMENSION application (which is commonly the case), the user of your application might be requested to connect to the Number Server application.



As part of the Informed Number Server installation process, the administrator must be sure to enable program linking privileges for the appropriate users. These settings determine if each user is required to connect as a registered user or if guest access is permitted. To connect as a registered user, entry of a name and password is required.

After calling GetNSNameList, you can then call GetNSName to obtain the name of a specific form number. $theIndex specifies which form number; its value should be between 1 and the number returned by GetNSNameList in $theCount. The name of the form number is returned in the string parameter $theName.

GetNSNameList and GetNSName are external functions that return long integer results. A result of zero indicates that the function completed successfully. Non-zero results correspond to the various errors that can occur. For information about error handling, please see “Error Codes” later in this chapter.

Requesting a New Form Number

Whenever a new form number is needed, your 4th DIMENSION application should call the external function GetNSNextValue.

```
GetNSNextValue ($theAppName, $theMacName, $theZoneName,
                        $theNumberName, $increment, $theValue)
```

$theAppName, $theMacName, and $theZoneName identify the Number Server application and the computer on which it’s running. If your network consists of a single zone only, pass the asterisk character (‘*’) in $theZoneName. Otherwise, pass the zone name.

GetNSNextValue will connect to the specified Number Server application and request the next available number for the form number identified by the string parameter $theNumberName. Only if you pass 1 in $increment, the Number Server application will actually increment the corresponding form number. By passing 0, you can obtain the next available form number without advancing it. This feature is useful for testing purposes. The form number is returned in the string parameter $theValue.

Like GetNSNameList, calling GetNSNextValue may require that the user of your application connect to the Number Server application. Please see the explanation of GetNSNameList for more information.

GetNSNextValue is an external function that returns a long integer result. A result of zero indicates that the function completed successfully. Non-zero results correspond to the various errors that can occur. For information about error handling, please see “Error Codes” later in this chapter.

Error Codes

Each of the four Informed Number Server external functions return a long integer result code. A non-zero result can occur due to invalid input parameters or problems with network or operating system components or with the Number Server application itself. The following table lists some of the more common errors, each with a brief explanation.

Common Error Codes

Error Code		Description
0	No error	The function or procedure completed successfully
-128	userCanceledErr	The user clicked Cancel on the Program Linking dialog box.
-1708	errAEEventNotHandled	An Apple event was not handled by the target application. This error normally indicates that the application specified by the parameters to either GetNSNameList or GetNSNextValue is not a Number Server application.
-1712	errAETimeout	The Number Server application is not responding and a time out occurred
-906 -908 -915	destPortErr noSessionErr noResponseErr	The errors -906, -908, and -915 will occur if the Number Server application is not running on the computer specified by the parameters $theAppName, $theMacName, and $theZoneName.
-1		The form number specified in $theNumberName when calling GetNSNextValue does not exist.
-1000	kIndexRangeErr	The value of $theIndex passed to the GetNSName function is out of range.