

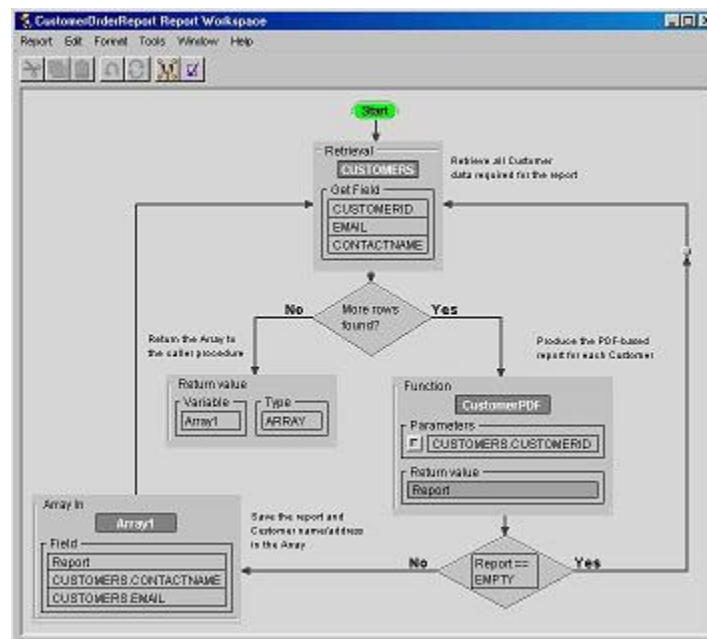
Tutorial 18 - Passing Arrays between procedures

In Tutorial 11 we made use of the Array component to accumulate data retrieved from a group of tables, and then to return it to the procedure for drawing a bar chart. Both instances of the Array component ("Array In" and "Array Out") resided within the same procedure, and they were not visible/useable outside of it. In other words, their lifespan was limited by that of the procedure, and when the procedure exited, the Array components were destroyed.

However, Arrays can also be passed from one procedure to another. The ability to share Array components between different procedures facilitates structured design, with some procedures doing data extraction and formatting (and placing it into the Array), and other procedures making use of the extracted data by retrieving it from the Array and thus focusing on the specific task at hand.

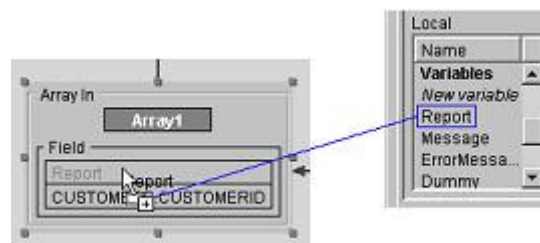
In this Tutorial, we will revisit the MailCustomerOrderInfo process built in Tutorial 13, and split it along the data production (retrieval) and data consumption (sending e-mail) lines. This time, we will package the procedures as Reports, rather than Processes.

Taking the MailCustomerOrderInfo process as the prototype, we build the following Report procedure and save it as CustomerOrderReport:

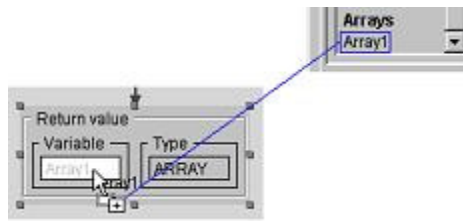


Notes:

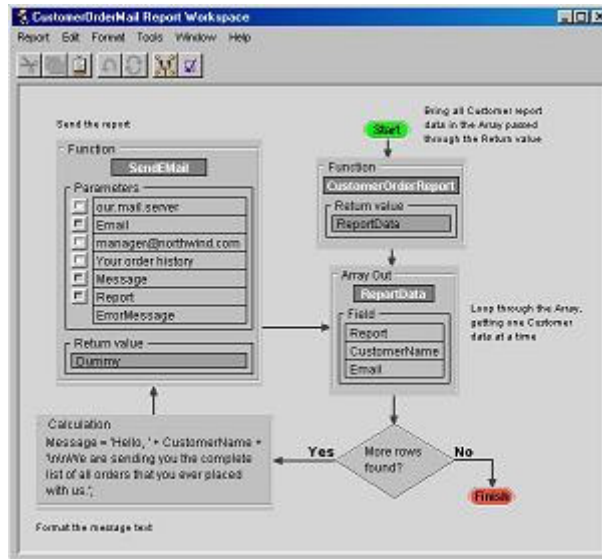
1. The "Report" variable created by the CustomerPDF Function component is passed to the Array, along with the Customer ID, as shown in the following figure:



2. The Array component is passed in its entirety (identified by its name) to the Return value component, as shown below:



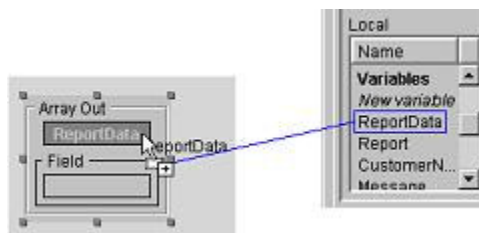
The outer (calling) Report procedure now becomes very small and simple:



We will save it as CustomerOrderMail report.

Notes:

1. The CustomerOrderReport function returns a variable that is named "ReportData" in the procedure. This variable is then dragged into the Array component's Header area as shown below:



and the component is sized to accommodate three fields, as required by the Array created inside CustomerOrderReport.

2. Scribe performs rigorous syntactical check on the Array that receives its content from another procedure. It makes sure that both the number of fields, and their data type (as established from the procedure's context) match the Array component in the procedure that returned it. If a mismatch occurs, an error message is displayed, similar to the one below (a field was intentionally removed from the ReportData Array in order to demonstrate this):

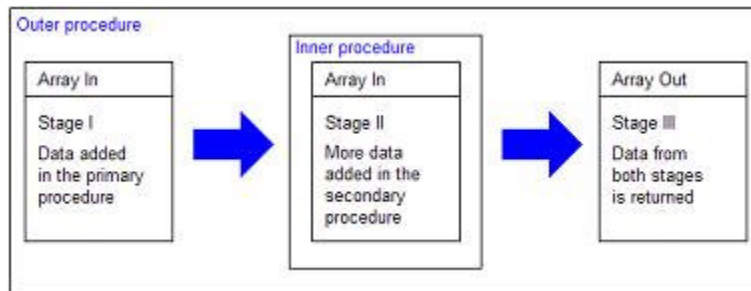


Other ways of passing Arrays

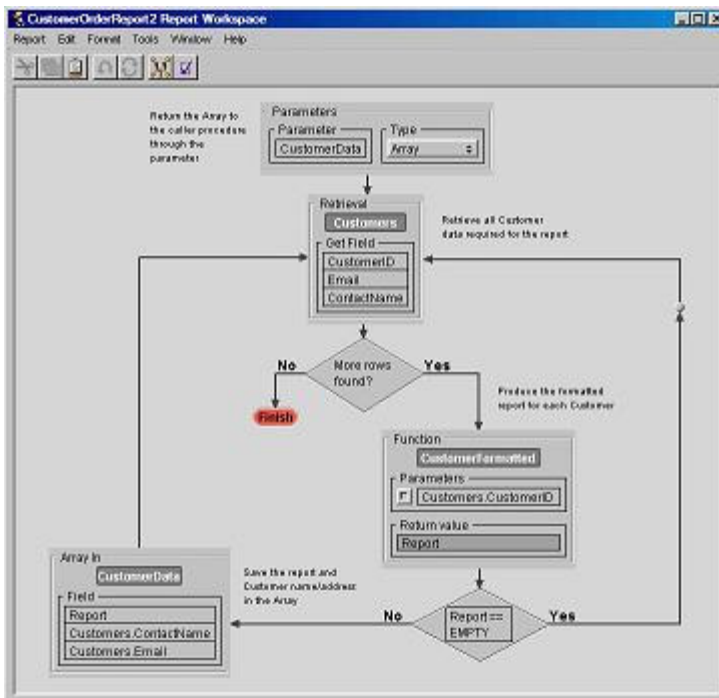
Arrays can be passed into and out of procedures through the Parameters component; using the "Array" data type:



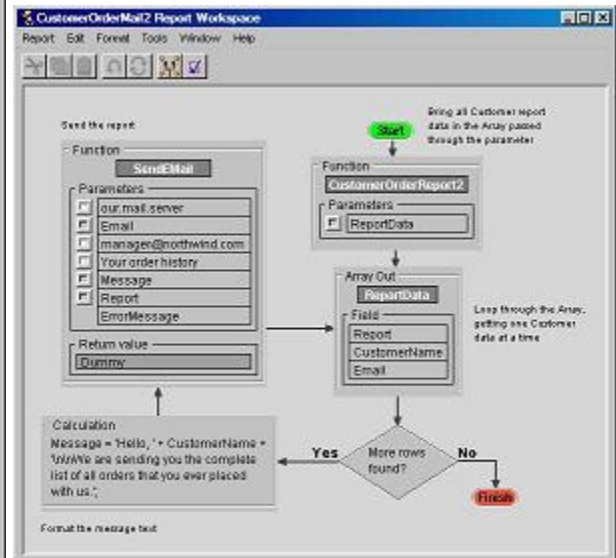
The work of loading data into the Array can be shared between different procedures. The figure below schematically shows an Array In that is partially filled in the calling (outer) procedure, and then is passed as a parameter into an inner (called) procedure which continues adding data to it. Finally, the Array Out component retrieves all data:



The following pair of Report procedures is based on CustomerOrderReport and CustomerOrderMail, modified to pass the Array through the Parameters component:



CustomerOrderReport2



CustomerOrderMail2

Array field data types

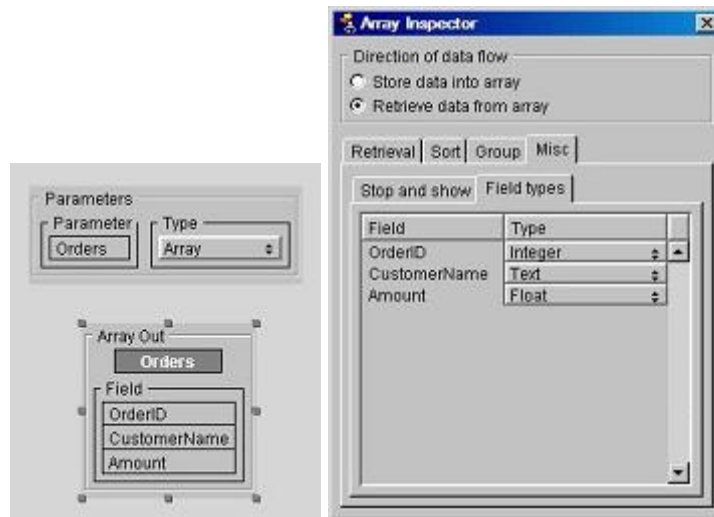
Most of the programming environments rely on the end user to provide information on the data types of variables used in the context of the program's logic. First, the user declares the variables to be of a certain type (like text string, integer, floating-point value etc.), the compiler then runs syntactical checks to ensure the compliance of the program code to the declared data types.

Scribe is designed to *derive* all data type information from the context of the procedure where the variables are used. This information is inferred from the way the variables are used in the procedure (e.g. the Calculation component with "var1 = 3.14" creates a variable "var1" of FLOAT data type), or from the data type of the database table fields taking part in arithmetic/assignment expressions. In short, Scribe tries to figure out how to handle the variables and set their data types without asking too many questions of the user. Only when it is impossible to implicitly derive the type of the variable, Scribe requires the user to provide the necessary information; an example of this is the Parameters component, where every parameter is accompanied by a data type pop-up list.

This type-setting strategy applies to the Array fields as well. Arrays In inherit their data types from the variables that are passed into the Array component. Arrays Out, when used together with similarly-named Arrays In, adopt their field data types from those in their Array In counterparts. Arrays (either In or Out) that are passed as parameters into lower-level array-handling procedures are assigned their data types by the Syntax Checker that extracts the data types from these procedures; the same happens with the Arrays that are returned from the procedures.

There is only one case where deriving of the data types from the procedure context is not possible: when the procedure takes an Array as a parameter, expecting the Array to be formatted outside, and attempts to extract the data from this Array using the Array Out form of the component. In this case, the user must explicitly assign the data type to each field listed in the Array component; this is done in the "Field types" tab view of the Array Inspector that becomes selectable (normally it is disabled).

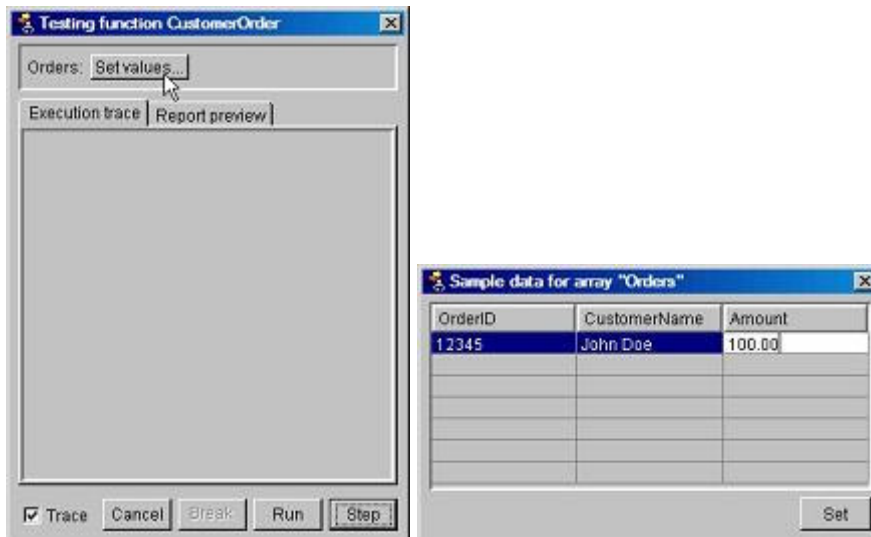
Here is the example:



Array Out passed in as a parameter "Field Types" tab activated

Testing with Array parameters

When procedures are tested, their input parameters are displayed as editable fields, or pop-up lists, at the top of the Tester window. Arrays, passed as parameters, do not fit into the "one parameter - one field" paradigm. Scribe provides an editable table to enter the test data that becomes part of the Array and is passed into the procedure. The following example is based on the Parameter component shown in the previous section:



Array parameter is shown as a button Table for setting the Array test data

Pressing the "Set values..." button brings up the modal panel with the editable table; pressing the "Set" button in it passes the table's content into the Array.