Chapter 6
Interfacing to Application DLLs

Introduction

R&R includes a special function, CDLL( ), that allows you to call a function in a Windows Dynamic-Link Library (DLL) from an R&R report. You might use CDLL( ) when you want to write a DLL-based function to perform an operation that R&R's UDFs don't support, such as a trigonometric function. CDLL( ) also gives R&R access to functions that are used by other elements of your application, since DLLs are available to all parts of a Windows application.

Syntax

CDLL( ) takes three string arguments and returns a string value. The syntax is:

```
CDLL(string1,string2,string3)
```

where **string1** is the name of the DLL that contains the function, **string2** is the name of the function, and **string3** is an argument being passed to the DLL function. You can use R&R functions to convert the argument from other data types and to return other data types. For example, the calculated field expression

```
CDLL("CONVERTS.DLL","MILES_KILO",STR(DISTANCE))
```

uses the STR function to convert the decimal value of DISTANCE into a character string and passes the string value to the MILES_KILO function in CONVERTS.DLL, which converts the distance in miles to kilometers.

CDLL( ) expects a boolean return value from the called DLL function: true to indicate the function executed successfully, or false to indicate an error. If the DLL returns a false value, CDLL( ) returns an error string. If the DLL function executes successfully, it should overwrite its input string with the output string to be returned by the R&R CDLL( ) function. R&R passes the input and output strings using an 8000-byte buffer.

Example

This example uses CDLL( ) to call the functions RR_SIN, RR_COS, and RR_TAN from TRIGS.DLL. The functions are used to return the sine, cosine, and tangent values of the field DEGREES. Since CDLL( ) takes an input string and produces an output string, we first created three UDFs that take the value of DEGREES as a numeric and return its value as a numeric. These values are converted to character strings before being passed to TRIG.DLL. The three UDFs and their declarations and formulas are:

```
SIN(N_DEGREES) =
   VAL(CDLL("TRIG.DLL","RR_SIN",STR(DEGREES,6,0)))
COS(N_DEGREES) =
   VAL(CDLL("TRIG.DLL","RR_COS",STR(DEGREES,6,0)))
TAN(N_DEGREES) =
   VAL(CDLL("TRIG.DLL","RR_TAN",STR(DEGREES,6,0)))
```

In each UDF formula, the STR function converts the numeric value of DEGREES into a character string, as required for the third argument to CDLL( ). The second argument of STR specifies the character length of the string.; the third argument specifies the number of decimal places. The VAL function converts the string result of CDLL( ) into a numeric representation, which is more useful for such functions.

Creating these UDFs allows you to supply the DEGREES argument as a numeric value and return it as a numeric value; the conversion to string representation and back is "hidden." To use these UDFs to access the DLL functions, you create calculated fields whose expressions supply the DEGREES as numeric arguments to SIN( ), COS( ), and TAN( ).

Note the following:

- Although you can pass only a single argument to a DLL, that string can contain multiple arguments that can be parsed by the DLL function. The single string value returned by the function can also contain multiple values that can be parsed in R&R within a calculated field expression.
- If you use CDLL( ) in reports you plan to distribute for use with the Runtime, make sure that the referenced DLLs are available when the Runtime is executed.