Title:	D2KCOMN Library
Version:	2.0.3
Date:	26-JUN-1997
Author:	DRMILLS
Purpose:	This document defines the prototypes and usage of the D2KCOMN PL/SQL library. Information on each program unit or package within the library is provided, along with dependencies

1. D2KCOMN {Function}

1.1. Prototype

D2kComn Return VARCHAR2

Returns a version string for the D2Kcomn library

1.2. Dependencies

None

2. D2K_BuildParamList {Function}

BuildParamlist provides a quick and simple way to create a parameter list to pass to another Form, or to Graphics & Reports. The actual creation of the list is internal to the function as is cleaning up old copies of the list.

2.1. Prototype

```
BuildParamlist (  ParamString       in varchar2,
                  ParamListName     in varchar2
        ) return PARAMLIST
```

The **ParamString** (varchar2) is just a sequence of ParameterName=ParameterValue statements separated by a pipe symbol. Thus, the argument '*ENAME=SCOTT|EMPNO=1234*' would create a parameter list with two entries, ENAME and EMPNO, with their respective values.

To indicate a DATA parameter rather than a TEXT parameter prefix the name of the parameter with the keyword 'DATA:' e.g. '*ENAME=SCOTT|**DATA:**EMPQUERY=RECORDGROUPNAME*' this will then create a parameter list with a text parameter ENAME and a data parameter EMPQUERY.

You are responsible for escaping any quotes or spaces in TEXT data strings into a format acceptable by the target product..

ParamlistName can be left null for a default name, or if you need to create several co-existing parameter lists you can supply your own name. If you re-use a list name, the old list will be replaced by the new one.

2.2. Dependencies

D2K_Delimited_String

2.3. Error Handling

If a error is raised from the function the exception NO_DATA_FOUND will be raised this will only occur if the function is unable to create the new parameter list for some reason.

3. D2K_Delimited_String{Package}

D2K_Delimited_String is a package for handling delimited strings. The strings can be delimited by any character or set of characters (default is comma), and can be terminated or un-terminated by the delimiter. The various functions within the package provide random access to the delimited string, much as if it were an array.

The package can handle values of up to 2000 chars in length, although this could be increased to 32k if required by changing the buffer sizes. The package is not capable of handling multi-byte strings.

All the values saved to the string are of course saved as string values, but can be extracted to their native datatypes using the correct Get functions. GetString will extract any data type into a string variable.

Numeric data types are stored in the supplied precision, and Dates with full precision.

All of the Functions and Procedures within the Package take three common arguments:

- **SourceString** in varchar2 → The delimited string being accessed
- **Unterminated** in boolean → TRUE if there is no trailing delimiter or FALSE (default) if there is. e.g. in a string that had a comma for the delimiter " a,b,c" would be classed as Unterminated=TRUE, whereas "a,b,c," would be FALSE.
- **Delimiter** in varchar2 → The actual delimiting char (or chars). This defaults to a comma.

3.1. Prototype

3.1.1. Constants

None

3.1.2. Variables

none

3.1.3. Procedures

3.1.3.1. **PutString**(
```
Source_string      in out varchar2,
String_to_add      in varchar2,
Field_position     in number,
UnTerminated in Boolean  default FALSE,
Delimiter      in varchar2 default ',');
```

PutString adds the string specified in String_to_Add to the specified Field_Position. The procedure will handle adding any required delimiters if the position is bejond the current scope of the Source_string. Source_String itself is implicitly updated with the new value.

3.1.3.2. **PutNumber**(
```
Source_string in out varchar2,
Number_to_add      in varchar2,
Field_position in number,
UnTerminated in Boolean  default FALSE,
Delimiter      in varchar2 default ',');
```

As PutString for adding a number.

3.1.3.3.**PutDate**(Source_string in out varchar2,
Date_to_add in varchar2,
Field_position in number,
UnTerminated in Boolean default FALSE,
Delimiter in varchar2 default ',');

As PutString for adding a date.

3.1.3.4.**Put** → An overloaded version that can put any of the above datatypes.

3.1.4. Functions

3.1.4.1.**Counter** (Source_string in varchar2,
UnTerminated in boolean default FALSE,
Delimiter in varchar2 default ','
) return number;

Returns the number of members in the string e.g. delimstr.counter('a,b,c,d,') would return "4".

3.1.4.2.**GetString**(Source_string in varchar2,
Field_position in number,
UnTerminated in boolean default FALSE,
Delimiter in varchar2 default ','
) return varchar2;

Returns the value at Field_Position as a string

3.1.4.3.**GetNumber**(Source_string in varchar2,
Field_position in number,
UnTerminated in boolean default FALSE,
Delimiter in varchar2 default ','
) return number;

Returns the value at Field_Position as a number. If the field in question does not contain a valid number a value error will be raised.

3.1.4.4.**GetDate**(Source_string in varchar2,
Field_position in number,
UnTerminated in boolean default FALSE,
Delimiter in varchar2 default ',
') return date;

Returns the value at Field_Position as a date. If the field in question does not contain a valid date a value error will be raised.

3.1.4.5.**Locate**(Source_string in varchar2,
Search_<> in <overloaded>,
UnTerminated in boolean default FALSE,
Delimiter in varchar2 default ',') return number;

An overloaded function which returns the field number of the first field that exactly matches the supplied string, date or number. Returns Zero if no match is found.

3.2. Dependencies

None

3.3. Error Handling

None

4. D2K_Filler {Package}

D2K_Filler raises a modal dialog window containing a progress bar. Each time the increment procedure within the package is called, the progress bar is increased in size.

The programmer has to call the Setup procedure to define the scope of the Fill (e.g. the number of steps required must be calculated in advance. After this, calls to the increment procedure will increase the size of the filler bar, and update text in the display with progress information if required.

4.1. Prototype

4.1.1. Constants

None

4.1.2. Variables

4.1.2.1. Interrupted [Exception]

You can raise this exception to stop the control in a clean manner.

4.1.3. Procedures

4.1.3.1. **Setup** (

```
WindowTitle    in varchar2,
MessageText    in varchar2,
ReturnTo       in varchar2,
MinValue       in number default 0,
MaxValue       in number default 100,
FillerBlock in varchar2 default 'FILLER$')
```

WindowTitle sets the title for the dialog and MessageText defines the message displayed in the window. ReturnTo defines where the cursor should go when the OK button on the dialog is pressed. MinValue and MaxValue define the range for the fill in conjunction with the IncrementBy argument of the Increment procedure. Generally the range will start at Zero (when the bar will be at 0%) and increase in steps up to the MaxValue at 100%. The number of Increments to fill the control is (MaxValue-MinValue) / IncrementSize.

You supply a different value for the FillerBlock when you've subclassed and renamed the filler block from the D2COMN object library. If you have not renamed the block, then you can leave the value at the default block name of "FILLER$".

4.1.3.2. **Increment** (

```
Activity       in varchar2,
IncrementBy    in number default 1)
```

Each call to Increment will advance the filler bar and the text in the Activity variable is put into a progress field.

4.1.3.3. **OK_Button**;

Returns focus back to the item specified in the Setup call.

4.1.3.4. **Aborted**

Resets the internal state of the control when an error occurs. If you implement your own interrupt mechanism, e.g. by using functions within D2KWUTIL.PLL, then be sure to call the **Aborted** procedure after the interrupt, in order to ensure that subsequent calls to the component work.

4.2. Dependencies

D2KCOMN.OLB → Object Group D2K_FILLER_COMPONENT
D2K_SHOW_WINDOW_CENTERED

4.3. Error Handling

If a VALUE_ERROR is raised by the calculations that resize the bar, or if the programmer causes the d2k_filler.interrupted exception to be raised, then the Aborted procedure will be called to reset the control. Any other exceptions will be passed through.

5. D2K_Finder {Package}

Finder is a package concerned with locating files, it provides a way of listing each of the available directory locations that are contained in ORACLE_PATH, FORMS45_PATH, FORMS50_PATH, TK23_ICON, TK25_ICON and the source directory containing the current form. Additionally it has a procedure for checking that a named file actually exists.

You can add additional path variables to search by altering the package body initialisation section. Any path variables that you add must be created at the same scope as ORACLE_PATH, e.g. in the HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE branch of the Registry, or in the current UNIX shell.

5.1. Prototype

5.1.1. Constants

None.

5.1.2. Variables

5.1.2.1. PossiblePaths [pls_integer]

PossiblePaths is initialised when this package is first accessed. It contains the number of paths that have been extracted from the various information sources.

5.1.3. Procedures

None.

5.1.4. Functions

5.1.4.1. **GetDirString** (PathIndex in pls_integer) return varchar2

GetDirString pulls out the requested path from the list held internally by the package. The path is returned with a trailing separator.

The normal usage would be to Loop from 1 to PossiblePaths, using GetDirString to pull out each path in turn, then test each path for a particular file. The directory string is returned with a terminating directory separator already included. E.g. C:\TEMP would be returned as C:\TEMP\.

5.1.4.2. **GetLocation** (FileName in varchar2) return varchar2

Unlike GetDirString which gives you a list of possible discreet directories for you to examine, GetLocation does the hard work for you and returns the full path and name of the requested file if it's anywhere in the directory list that D2K_Finder holds.

5.1.4.3. **FileExists** (FileName in varchar2) return boolean

If the named file exists in the specified location TRUE will be returned, else FALSE.

5.2. Dependencies

D2K_OSFILES

5.3. Error Handling

None

6. D2K_OSFiles {Package}

OSFiles is a utility for parsing out the various components of a file name (e.g. path to file, the name itself, file extension etc.) in a platform independent way.

6.1. Prototype

6.1.1. Constants

None

6.1.2. Variables

None

6.1.3. Procedures

None

6.1.4. Functions

6.1.4.1. **SepChar** return varchar2

Just returns the separator character for the Operating System the form is running on.

6.1.4.2. **Name** (FileString in varchar2) return varchar2

Extracts the File name only, without Path or extension from the FileString.

6.1.4.3. **FullName** (FileString in varchar2) return varchar2

Extracts the full path to, and the filename itself, but no file extension.

6.1.4.4. **QualifiedName** (FileString in varchar2) return varchar2

Extracts the File name and extension, but no path.

6.1.4.5. **Path** (FileString in varchar2) return varchar2

Extracts just the file component of FileString. The vcalue returned is not terminated with the separator for that platform.

6.1.4.6. **Extension** (FileString in varchar2) return varchar2

Extracts just the file extension from FileString.

6.1.4.7. **MakeName** (Path in varchar2, FileName in varchar2, Extension in varchar2) return varchar2

Reverses the whole process and allows you to build a correctly formatted file string for your platform.

6.2. Dependencies

None

6.3. Error Handling

None

7. D2K_Res {Package}

The D2K_Res package is concerned with reading text strings from resource files. These can be retrieved as normal strings, or as messages, or into alerts. D2K_Res can handle multiple open resource files at once, and maintains it's state across multiple forms, providing that the library is shared.

7.1. Prototype

7.1.1. Constants

None

7.1.2. Variables

None

7.1.3. Procedures

7.1.3.1. **InitialiseRes** (ResourceList in varchar2,
UseNLS in boolean)

InitialiseRes should just need to be called once in a Forms session. It is concerned with checking the existence of and opening the requested resource files. The ResourceList argument needs to be a comma separated list of all the required resource files (if more than one). You may supply the full path to the resource file. If you do not D2k_Res will use the D2k_Finder package to search for the resource file. In each case it will look in each of the possible directories, then in a sub-directory called "resource" off each directory.

Do not include the .RES extension in the filenames, the procedure will add these for you.

If the UseNLS boolean is set to TRUE, then the package will check the NLS settings for the current system and append the correct NLS territory abbreviation onto the filename, so allowing auto-selection of language specific versions of resource files. E.g. with an NLS setting of america_american.we8iso59p, UseNLS set to TRUE, and a supplied resource file name of "DEMO" then the package will locate a resource file called "DEMOUS.RES"

You can attempt to load the same resource file many times, the package will only actually load it the first time it encounters it in the runform session.

7.1.3.2. **CloseRes**

Closes any resource files opened by InitialiseRes in this instance of the library. All the open resource files should be closed when Forms Exits and so you'll probably never use this procedure in any case.

7.1.3.3. **Debug** (String in boolean,
Msg in boolean,
Response in boolean,
Files in boolean);

The debug procedure acts as an aid to system builders and administrators. The boolean arguments will cause the strings returned in each case to have the message identifier prefixed to them. So if you need to alter the text of, say a message, you can set the Msg argument to TRUE, then all messages, (or the alert title in the case of alerts) will be prefixed with the resource identifier in angle brackets e.g. `<Resource_Id>Message Text`

You must set each of the boolean arguments at the same time, they are all initialised to FALSE. If you set one of the arguments to NULL, it's state will remain unchanged.

Setting Files to TRUE will report the name and location that is tested for each resource file.

7.1.3.4. **GetMsg** (Target in varchar2,
Arg1..Arg10 in varchar2)

GetMsg returns the specified resource in the form of a message. This can either be directed to an alert, or to the message line. In both cases the length of the string resource will be limited to 200 chars.

The resource string should be formatted into the following "fields" as follows with each value delimited by a Pipe symbol "|".

1. Text (max length implicitly 200 chars)
2. Alert Style (as varchar2) choose from 'NONE' (default), 'STOP', 'CAUTION' or 'NOTE'.
3. Audible Signal: choose from 'BEEP' or 'NOBEEP' (note that messages displayed in alerts may beep as a result of O/S setup independently of this setting)
4. Alert Title (default 'Error')

If and Alert style, the alert for a message will always be defined with a single button with the text 'OK'

So to define an message with a Stop Icon, the title of 'Fatal Error' the text of 'An Error Has Occurred', and a beep to alert the user, you would enter the following string into the resource file.

```
An Error Has Occurred|STOP|BEEP|Fatal Error
```

No terminating pipe symbol is required.

If an alert style of NONE (or blank) is selected, the message will just be echoed to the console line rather than being popped up in an alert.

A style of STOP, CAUTION or NOTE will cause a conventional windows alert to pop up, from which the user must press OK to exit.

If the resource string contains %s placeholders, these will be replaced with the contents of the arguments Arg1 to Arg10.

Any tab placeholders "%t" will be replaced with a tab character

7.1.3.5. ClearMsg

Cleans off the message line.

7.1.4. Functions

7.1.4.1. **GetString** (Target in varchar2,
MaxLength in pls_integer default 2000,
Arg1..Arg10 in varchar2
) return varchar2

GetString just returns the specified string. It will search through each open resource file until it is found. You can ensure that the returned string does not exceed a specified length by including the MaxLength parameter.

Resource files have a maximum length of 1984 chars per resource.

If the resource string contains %s placeholders, these will be replaced with the contents of the arguments Arg1 to Arg10.

Any tab placeholders "%t" will be replaced with a tab character

7.1.4.2. **GetResponse** (Target in varchar2,
Arg1..Arg10 in varchar2
) return pls_integer

GetResponse takes the specified resource string from any open resource file, and displays it in an alert. The format of the alert is described by information within the string itself. The string in the resource file can be delimited into extra fields which will hint as to which type of alert to display.

The resource string should be formatted into the following “fields” as follows with each value delimited by a Pipe symbol “|”.

1. Text (max length implicitly 200 chars)
2. Alert Style (as varchar2) choose from ‘STOP’, ‘CAUTION’ (default) or ‘NOTE’.
3. Alert Title (default ‘Caution’)
4. Button 1 Text (default ‘OK’)
5. Button 2 Text (default ‘Cancel’)
6. Button 3 Text (default NULL - No Third button)

So to define an alert with a Stop Icon, the title of ‘Fatal Error’ the text of ‘An Error Has Occurred’ and three buttons containing ‘Yes’, ‘No’ and ‘Cancel’ you would enter the following string into the resource file.

```
An Error Has Occurred|STOP|Fatal Error|Yes|No|Cancel
```

No terminating pipe symbol is required.

The function returns the number of the button that the user pressed in response to the alert. Use the defined forms alert button constants to compare against (ALERT_BUTTON1 etc.).

If the resource string contains %s placeholders, these will be replaced with the contents of the arguments Arg1 to Arg10.

Any tab placeholders “%t” will be replaced with a tab character

7.2. Dependencies

D2K_Delimited_String, D2K_Finder, D2k_OSFiles, ToBoolean

D2KCOMN.OLB → Object Group D2K_RES _COMPONENT

7.3. Error Handling

If a string resource is requested and cannot be found the value “<Target> No resource found” for will be returned.

8. D2K_Show_Window_Centered {Procedure}

A handy function for positioning a named window in the centre of the screen (relative either to the MDI frame or the physical screen).

8.1. Prototype

```
PROCEDURE SHOW_WINDOW_CENTERED ( WindowName     in VARCHAR2,
                                 WindowStyle    in VARCHAR2 := 'DOCUMENT',
                                 Console        in BOOLEAN := TRUE,
                                 Xoffset        in number,
                                 Yoffset        in number)
```

WindowStyle indicates if this is a Document or Dialog window. Documents are positioned relative to the MDI and Dialogs to the physical Screen.

The code checks for the USESDI parameter and positions relative to the display is this is set.

Console indicates that the console line should be taken into account.

The additional arguments Xoffset and Yoffset allow you to "tweek" the position of the centered window.

8.2. Dependencies

ToBoolean

8.3. Error Handling

None

9. ToBoolean {Function}

A simple function primarily for converting the value of checkboxes to a boolean value, e.g. Y become TRUE, N becomes FALSE.

9.1. Prototype

```
ToBoolean(      Sample          in varchar2,
                TrueValue       in varchar2 default 'Y'
        ) return boolean
```

Simply takes the sample value and compares it with the TrueValue. If they match it will return TRUE, else FALSE. If one of the values is NULL then NULL will be returned.

The function is case insensitive.

9.2. Dependencies

None

9.3. Error Handling

None

10. The D2KCOMN.OLB Object Library

Note: Objects in the D2KCOMN object library are all based on Points as a co-ordinate system.

10.1. D2K_FILLER Tab

This section of the library contains the object group *d2k_filler_component*. This object group contains the block, canvas and window that d2k_filler uses. Just drag the component in to your application intact

10.2. D2K_RES Tab

This section of the library contains the object group *d2k_res_component*. This component contains the Alerts used by the d2k_res package.