Using the Remote Access Library

The Remote Access Library (RACLib) was created to give non-Oracle Insurance applications the ability to start, stop, and control (to some degree) the Documaker Workstation, also known as the Processing System and PPS. The library provides API functions that can be called from any computer language that can interface to C functions in a DLL. In addition, an ActiveX component (RacCo) is provided for Windows.

This document discusses using Remote Access in one of two ways: through the RacCo ActiveX component or through direct API function calls. The various API functions provided by RACLib are described in detail in this manual as a Function Reference. The RacCo library is detailed in the ActiveX Object Reference and the Class Reference. Topics covered include using C, C++, Visual Basic, and Oracle forms to remotely access the Documaker Workstation.

Verifying Access to the System

Run Documaker Workstation

Before beginning to use the Remote Access Library to customize Documaker Workstation, make sure Documaker Workstation works in your environment. You should be able to run the AFEMNW32.EXE program to access your archives and do forms entry (if needed in your environment). For more information on setting up the system, refer to the Documaker Workstation Administration Guide and the Documaker Workstation User Guide.

Path statement

When you write your own application to access Documaker Workstation your program will need to locate and load DLLs and INI files that Documaker Workstation uses. Your application will make calls into RACLib, which will then call into other DLL files of Documaker Workstation. Therefore, the PATH environment variable should include Documaker Workstation program files location. Verify that the PC's PATH= variable includes the program files location; such as C:\FAP\DLL.

The normal Documaker Workstation installation process updates the PATH= variable to include the program directory.
Trouble Shooting

- Check your path statement. You should have C:\FAP\DLL (if that is the installed location) in your path statement. Be sure that it is in your path after a reboot.
- Check your working directory. If you are running from an icon make sure the working directory is correct.
- Check your INI settings. The INI files may be missing and you may have an incorrect directory path statements.
- Check to see if the shipping sample (SAMPCO) works. If not, perhaps you need to go through installation again.
Using the ActiveX Component (RacCo)

Library Platforms

<table>
<thead>
<tr>
<th>Win 32</th>
<th>OS/2</th>
<th>Win 16</th>
<th>MVS</th>
<th>Unix</th>
<th>OS/400</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACCO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Library Header Files

RACAPI.H, CRACLIEB.H

Overview of Component

The component code is responsible for...

- Determining current window handle that will be passed to RACLib functions. The RACLib functions create a window from the handle passed to them.
- Calling the RACTerminate function when the component destructs and before additional calls to RACLib functions.
- Loading the RACLib DLL (RACW32.DLL) dynamically.

Registering the ActiveX Component

You must register the RACCO.DLL file in the Windows registry to make the Remote Access Component (RacCo) available. To do so:

- Within the MS-DOS command prompt change directory to Documaker Workstation program directory.
  cd c:\fap\dll
- Type:
  regsvr32 racco.dll
Windows will display a message that shows the registration was successful.

## Component Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RacVw</td>
<td>View an archive</td>
</tr>
<tr>
<td>RacEdit</td>
<td>Edit a form set</td>
</tr>
<tr>
<td>RacImport</td>
<td>Import transactions</td>
</tr>
<tr>
<td>RacProxy</td>
<td>Edit form sets</td>
</tr>
</tbody>
</table>

## Using Visual Basic

### How to use the component from Visual BASIC

1. **Register component.**
   - From the command line, you can run the following program.
     - `Regsvr32 racco.dll`
   - If you need to unregister the component, type the following.
     - `Regsvr32 /u racco.dll`

2. **Introduce the type library to the Visual BASIC project.**
   - Select the `Project` menu.
   - Select `References`.
   - Click the checkbox beside `RacCo ’V.v’ Type Library.` (V.v indicates the version number)

3. **Now you can declare an object of the type `RacVw`.**
   - See Example below.
   ```vbscript
   Public RacLib as RacVw
   ```

The `RACSetWorkingPath` API can be used to set the current working directory to the MRL data set you want your application and DAP to work with. A good place to call this function is on the Form Load event, since the MRL should be the current directory before you can work with data in the directory.

### Example

The following sample code fragments retrieves an archived form set. The sample uses the SAMPCO master resource library that ships with Documaker Workstation. It creates a `RacVw` object and uses the `SetWorkingDir` method to set the working directory then calls `ViewByKey` to display the archive list selection window in Documaker Workstation.

First, we declared a `RacVw` object variable in the Declarations area of the form1 source file.
```vbscript
Public vbRacVw As RacVw
```

Next, the RacCo object is created and methods are called in sub-routines of the form1 source file. When the form is loaded, the `RacVw` object is created and the working path is set.
```vbscript
Private Sub Form_Load()
```
Set vbRacVw = New RacVw
vbRacVw.SetWorkingDir "c:\fap\mstrres\sampco"
End Sub

When a form button is pressed, RacCo is asked to retrieve archives that have a Key1 value starting with "F".

Private Sub cmdRACRetArc_Click()
    vbRacVw.ViewByKey "F", "", "", 1, 0
End Sub

When the ViewByKey method executes, a window is created to display the form set in and the following Archive List window appears.

Once a policy is selected, the form set appears in the created window.

Using C++

Files

<table>
<thead>
<tr>
<th>File Type</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveX DLL</td>
<td>RACCO.DLL</td>
</tr>
<tr>
<td>Type Library</td>
<td>RACCO.TLB</td>
</tr>
</tbody>
</table>

Example

The following sample code fragments retrieves an archived form set. The sample uses the SAMPCO master resource library that ships with Documaker Workstation. It creates an IRacVw object and uses the SetWorkingDir method to set the working directory then calls ViewByKey to display the archive list selection window in Documaker Workstation.

First, we create a RacCo object's IRacVw class using the type RACCO.TLB library. In the Microsoft Visual Studio you can choose:

- Select the View menu.
- Select Class Wizard...
- Select Add Class...
- Select From a type library...
- Locate RACCO.TLB

Next, create an instance of the **IRacVw** object when the application is initialized. Then set the working path for RacCo.

```cpp
pRacVw = new IRacVw();
if (pRacVw != NULL)
{
    pRacVw->CreateDispatch("DocuCorpDAPRacCo.RacVw");
    pRacVw->SetWorkingDir("c:\fap\mstrres\sampco");
}
```

When a menu item is selected, RacCo is asked to retrieve archives that have a Key1 value starting with "F".

```cpp
pRacVw->ViewByKey("F", "", "", 1, 0);
```

When the **ViewByKey** method executes, a window is created to display the form set in and the following Archive List window appears.

![Archive List](image)

Once a policy is selected, the form set is displayed in the created window.
Using API Functions

Library Platforms

<table>
<thead>
<tr>
<th></th>
<th>Win 32</th>
<th>MVS</th>
<th>Unix</th>
<th>OS/400</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RACW32</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Library Header Files

RACAPI.H, RACLlib.H

Overview of API Functions

Oracle Insurance's RACLib was created to give non-Oracle Insurance applications the ability to start (and to somewhat control) the form's entry system.

RACLib supports both a modal or non-modal session. A modal session will assume control of window message handling and only return once the user completes form entry. A non-modal session relies upon the controlling application for message handling.

A non-modal session allows an external application to remain responsive to message commands. However, this greater control result in a little more complexity because the application becomes responsible for ensuring that the entry system performs all the necessary operations to collect and save data.

This reference includes those functions exported by RACLib for use by external applications. External applications can either include the RACAPI.H header into the source and link with the RAC import library (RACW32.LIB), or manually load the DLL thus querying the procedure address by name before each call. The examples in this documentation will assume the application is linked with RACLib.

For those applications that attempt to query procedures by name, it will still be important to note the prototypes of these functions. Calling a function indirectly (as this manner suggests) does not ensure the correct calling convention is used or that the correct size of parameter is pushed. When in doubt, provide a proper variable cast (especially when a "long" variable is required under Windows).

Most functions are prototyped with the calling convention defined by _RACAPI, others use EXPENTRY defines.
The definition of \_RACAPI is:
```
#ifdef WIN32
#define _RACAPI WINAPI
#define _RACAPIPTR WINAPI
#endif
```

The type \_EXPENTRY is defined for Windows as:
```
#ifdef __WIN__
#define     EXPENTRY    far pascal
#endif
```

Please note that \_VMMAPI for Windows 32-bit is defined as nothing.
```
#ifdef WIN32
#define _VMMAPI
#endif
```

This is included for platform compatibility and may change in the future.

### Archive Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACRetrieve</td>
<td>Display transaction from archive.</td>
</tr>
<tr>
<td>RACRetrieveArchive</td>
<td>View form set specified.</td>
</tr>
<tr>
<td>RACRetrieveArchiveHab</td>
<td>View form set specified.</td>
</tr>
<tr>
<td>RACViewByKey</td>
<td>View form set specified.</td>
</tr>
<tr>
<td>RACViewByKeyHab</td>
<td>View form set specified.</td>
</tr>
<tr>
<td>RACViewData</td>
<td>View form set specified.</td>
</tr>
<tr>
<td>RACViewDataHab</td>
<td>View form set specified.</td>
</tr>
</tbody>
</table>

### Entry Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACCreate</td>
<td>Create a New WIP Entry</td>
</tr>
<tr>
<td>RACCreateEntry</td>
<td>Create a New WIP Entry</td>
</tr>
<tr>
<td>RACCreateWipEntry</td>
<td>Create a New WIP Entry</td>
</tr>
<tr>
<td>RACEditData</td>
<td>View form set specified.</td>
</tr>
<tr>
<td>RACEntry</td>
<td>Start the Entry (PPS) System.</td>
</tr>
<tr>
<td>RACMain</td>
<td>Remote Start of Entry (MODAL)</td>
</tr>
<tr>
<td>RACSave</td>
<td>Save transaction.</td>
</tr>
<tr>
<td>RACThread</td>
<td>New Process operating within another window.</td>
</tr>
<tr>
<td>RACUpdate</td>
<td>Edit an existing WIP entry.</td>
</tr>
<tr>
<td>RACWipSelectFunction</td>
<td>WIP selection support function.</td>
</tr>
<tr>
<td>RACWipSelection</td>
<td>Support function for WIP selection</td>
</tr>
</tbody>
</table>
### Error Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACGetDescription</td>
<td>Return description of error message</td>
</tr>
<tr>
<td>RACGetStatus</td>
<td>Get Last Error Status Code</td>
</tr>
<tr>
<td>RACSetStatus</td>
<td>Set the Error Status Code</td>
</tr>
</tbody>
</table>

### Form Set Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACCountPages</td>
<td>Count the pages in a form set</td>
</tr>
<tr>
<td>RACFindCompanyLOB</td>
<td>Locate a specified Key1 and Key2 values</td>
</tr>
<tr>
<td>RACFindTransaction</td>
<td>Locate a specified transaction</td>
</tr>
</tbody>
</table>

### Proxy Functions

The proxy file is the file representation of one DAP transaction. Usually, it is the archive transaction. This file has enough information for the Power Office application to be able to get the key information for display. Additionally, if needed, it has form names, section names, field data, and so on, so the application can invoke some kind of text search mechanism. Therefore, proxy files can be thought of as another import/export file format.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACArchive2Proxy</td>
<td>Display transaction from archive.</td>
</tr>
<tr>
<td>RACSaveAsProxy</td>
<td>Save the current form set in the current AFEData structure into a proxy file.</td>
</tr>
<tr>
<td>RACSetProxyHandle</td>
<td>Set the document handle of a proxy file for RACViewProxy.</td>
</tr>
<tr>
<td>RACViewProxy</td>
<td>Display archive from proxy file.</td>
</tr>
</tbody>
</table>

### Session Functions

Multiple Session RACLib allows Documaker Workstation to have concurrent multiple desktops within the same process. Each window can perform different WIP and ARCHIVE functions within the same process concurrently and independent of the other session. A window's handle identifies each session. The RACInit function attaches Documaker Workstation to a window's handle. This window's handle can be obtained by calling RACGetFrameWindow.
The window's handle returned by RACGetFrameWindow will change after each successive call to RACInit.

RACInit can be called from another function within RACTION or external to RACTION. The following RACTION functions call RACInit internally. Therefore, if you make successive calls to any of these APIs we have a new session.

- RACViewData
- RACViewByKey
- RACRetrieveArchive
- RACMain
- RACInitCtrl
- RACInitAll
- RACThread

The maximum number of sessions can be defined with the following INI option:

This example sets the maximum to five sessions. Therefore, the sixth call to RACInit will fail. The current version of RACTION will support no more than 10 concurrent sessions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACGetSession</td>
<td>Get the session for this window</td>
</tr>
<tr>
<td>RACSetCurSession</td>
<td>Set this window to be the current session.</td>
</tr>
<tr>
<td>RACSetSessionMenu</td>
<td>Set the menu's handle in the session structure</td>
</tr>
<tr>
<td>RACTermSession</td>
<td>Terminate Entry Session.</td>
</tr>
</tbody>
</table>

**System Functions**

**Command Line Arguments**

This module parses command line arguments and keeps the valid arguments in a static structure.

The pointer to the structure is initially NULL but it's filled by the RACSetCmdLineArg function. The memory for the structure is static but there is a linked list's handle that is created via VMMCreateList. This handle will not survive RACTerminate. Therefore, RACSetCmdLineArg will need to be called before each RACInit.

This pointer to the command line structure is set to static memory when RACSetCommandLineArg is called successfully.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACClose</td>
<td>Close.</td>
</tr>
<tr>
<td>RACGetAFEData</td>
<td>Get the Entry Data Structure</td>
</tr>
</tbody>
</table>
**RACInit** Initialize Entry System

**RACInitAll** Initialize Entire Entry System.

**RACInitCtrl** Initialize Entry System (menu and accelerator control).

**RACLibVersion** Get library version information.

**RACLoadIni** Load the INI file.

**RACPackDatabase** Pack the database.

**RACRestorePath** Restore original working directory.

**RACSetCmdLineArg** This allows historical command line options for AFEMAIN program to be passed to **RACInit**.

**RACSetIniFile** Sets the INI file path name used by RACLib functions.

**RACSetWorkingPath** Set current working directory.

**RACTerminate** Terminate Entry Session.

**RACWorkingPath** Establish correct working directory.

---

**Window Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RACCommand</strong></td>
<td>Request command execution.</td>
</tr>
<tr>
<td><strong>RACEnableMenu</strong></td>
<td>Change to the Entry Menu.</td>
</tr>
<tr>
<td><strong>RACGetAccelHandle</strong></td>
<td>Get the Accelerator Table Handle.</td>
</tr>
<tr>
<td><strong>RACGetClientWindow</strong></td>
<td>Return client window.</td>
</tr>
<tr>
<td><strong>RACGetFrameWindow</strong></td>
<td>Return frame window.</td>
</tr>
<tr>
<td><strong>RACGetMenuHandle</strong></td>
<td>Get the Entry Menu Handle.</td>
</tr>
<tr>
<td><strong>RACGetOrigMenuHandle</strong></td>
<td>Get the Original Menu Handle.</td>
</tr>
<tr>
<td><strong>RACHookProc</strong></td>
<td>Windows accelerator hook procedure</td>
</tr>
<tr>
<td><strong>RACLoadMenu</strong></td>
<td>This function will query the INI file for the MEN.RES compatible file to load.</td>
</tr>
<tr>
<td><strong>RACMainWndProc</strong></td>
<td>Documaker Workstation message handler.</td>
</tr>
<tr>
<td><strong>RACModal</strong></td>
<td>Start Modal Entry.</td>
</tr>
<tr>
<td><strong>RACRemoveScrollbar</strong></td>
<td>Remove the scroll bar from window.</td>
</tr>
<tr>
<td><strong>RACRestoreMenu</strong></td>
<td>Restore the Original Menu.</td>
</tr>
<tr>
<td><strong>RACSetCaptionOff</strong></td>
<td>Set caption off for this window.</td>
</tr>
<tr>
<td><strong>RACSetCaptionOn</strong></td>
<td>Set caption on for this window.</td>
</tr>
<tr>
<td><strong>RACSetHook</strong></td>
<td>Establish Windows message hook.</td>
</tr>
<tr>
<td><strong>RACSetParent</strong></td>
<td>Set parent window.</td>
</tr>
<tr>
<td><strong>RACSubClass</strong></td>
<td>Subclass a window's procedure with <strong>RACMainWndProc</strong>.</td>
</tr>
<tr>
<td><strong>RACUnhook</strong></td>
<td>Remove Windows message hook.</td>
</tr>
</tbody>
</table>
Using C/C++

The Remote Access Library (RACLib) was created using the C/C++ computer language. It is a natural fit to call the API functions from your C source. You can include the RACAPI.H header file and link the import library. The Function Reference in this document details the functions and their declaration syntax.

Files

<table>
<thead>
<tr>
<th>File Type</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include file</td>
<td>RACAPI.H</td>
</tr>
<tr>
<td>Import library</td>
<td>RACW32.LIB</td>
</tr>
<tr>
<td>DLL</td>
<td>RACW32.DLL</td>
</tr>
</tbody>
</table>

Example

The following sample code fragments retrieves an archived form set. The sample uses the SAMPCO master resource library that ships with Documaker Workstation. It sets the working directory by using the RACSetWorkingPath API function then calls RACRetrieveArchive to display the archive list selection window in Documaker Workstation.

First, the RACLib header file RACAPI.H is included.

```c
#include "racapi.h"
```

Next, the RACLib API functions are called in window procedure code when a menu item is selected. The working path is set then RACLib is asked to retrieve archives that have a Key1 value starting with "F".

```c
RACSetWorkingPath("c:\fap\mstrres\sampco");
RACRetrieveArchive(hWnd,"F","","",1,0);
```

When the RACRetrieveArchive function executes, a window is created to display the form set in and the following Archive List window appears.
Once a policy is selected, the form set is displayed in the created window.

Using Visual Basic

To make direct function calls into RACLib, you must declare the API's to be used in the General Declarations section of the Visual Basic project. The RACLIB.BAS file has been provided that declares all the Remote Access API's. You can include or copy the contents of the RACLIB.BAS file into your project. The Function Reference in this document details the functions and their declaration syntax.

Files

<table>
<thead>
<tr>
<th>File Type</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration file</td>
<td>RACLIB.BAS</td>
</tr>
<tr>
<td>DLL</td>
<td>RACW32.DLL</td>
</tr>
</tbody>
</table>

Example

The following sample code fragments retrieves an archived form set. The sample uses the SAMPCO master resource library that ships with Documaker Workstation. It sets the working directory by using the RACSetWorkingPath API function then calls RACRetrieveArchive to display the archive list selection window in Documaker Workstation.

First, the RACLib API functions are declared in the Declarations area of the form1 source file. These declarations can be found in RACLIB.BAS.

```vbnet
Private Declare Sub RACSetWorkingPath Lib "racw32.dll" _
    (ByVal filename As String)

Private Declare Function RACRetrieveArchive Lib "racw32.dll" _
    (ByVal hwndParent As Long, _
    ByVal Key1 As String, _
    ByVal Key2 As String, _
    ByVal keyID As String, _
    ByVal startpage As Long, _
    ByVal location As Long) As Long
```

Next, the RACLib API functions are called in sub-routines of the form1 source file. When the form is loaded, the working path is set.

```vbnet
Private Sub Form_Load()
    RACSetWorkingPath "c:\fap\mstrres\sampco"
End Sub
```

When a form button is pressed, RACLib is asked to retrieve archives that have a Key1 value starting with "F".

```vbnet
Private Sub cmdRACRetArc_Click()
    RACRetrieveArchive Form1 hwnd, "F", "", "", 1, 0
End Sub
```

When the RACRetrieveArchive function executes, a window is created to display the form set in and the following Archive List window appears.
Once a policy is selected, the form set is displayed in the created window.

Using ORACLE Forms

You can make direct function calls into RACLib from ORACLE Forms. You can specify the API's to be used in a globalvar package specification and package body. The Function Reference in this document details the functions and their declaration syntax.

Example Code

This is how the RACThread function has been accessed from Oracle forms.

The globalvar package specification and package body was defined under "PROGRAM UNITS" in a specific form that accesses the RACLib functions. A pushbutton was defined in the Oracle form to start the RACThread function. An Oracle package is a set of functions and data that is grouped together. The package idea is similar to a C++ class.

This is the specification section similar to a class specification defined in the include file.

```sql
PACKAGE globalvar IS
  threadactive NUMBER;
  fh_mylib ora ffi.libHandleType;
  fh_windll ora ffi.libHandleType;
  fh_thread ora ffi.funcHandleType;
  fh_terminate ora ffi.funcHandleType;
  fh_getparent ora ffi.funcHandleType;
END;
```

The body section contains function names and code that will be executed when the package is instantiated. In the globalvar package, there are no functions but the DLLs are loaded and the functions registered.

```sql
PACKAGE BODY globalvar IS
BEGIN
  threadactive := 0;
  fh_mylib := ora ffi.load_library('E:REL10\FAP400\SHIPW32’, ’racw32.dll’);
  fh_windll := ora ffi.load_library(’D:WINDOWS\SYSTEM’, ’user32.dll’);
  fh_thread := ora ffi.register_function(fh_mylib, ’RACThread’, ora ffi.PASCAL_STD);
END;
```
fh_terminate := ora_ffi.register_function(fh_mylib, 'RACTerminate', ora_ffi.PASCAL_STD);
fh_getparent := ora_ffi.register_function(fh_windll, 'GetParent', ora_ffi.PASCAL_STD);
ora_ffi.register_parameter(fh_thread, ORA_FFI.C_INT);
ora_ffi.register_return(fh_thread, ORA_FFI.C_INT);
ora_ffi.register_parameter(fh_terminate, ORA_FFI.C_INT);
ora_ffi.register_return(fh_terminate, ORA_FFI.C_INT);
ora_ffi.register_parameter(fh_getparent, ORA_FFI.C_INT);
ora_ffi.register_return(fh_getparent, ORA_FFI.C_INT);
END globalvar;

The following code was placed in the trigger section under a WHEN_BUTTON_PRESSED event. We obtain the parent of the current window so the RACThread window will not be placed inside the pushbutton.

declare
FUNCTION i_TestDll(funcHandle in ora_ffi.funcHandleType, aNumber in BINARY_INTEGER) RETURN BINARY_INTEGER;
pragma interface(c, i_TestDll, 11265);
FUNCTION i_Term(funcHandle in ora_ffi.funcHandleType, aNumber in BINARY_INTEGER) RETURN BINARY_INTEGER;
pragma interface(c, i_Term, 11265);
windowH BINARY_INTEGER;
begin
if globalvar.threadactive = 1 then
windowH := i_Term(globalvar.fh_terminate, windowH);
end if;
windowH := Get_Item_Property(name_in('SYSTEM.CURSOR_ITEM'), WINDOW_HANDLE);
windowH := i_TestDll(globalvar.fh_getparent, windowH);
windowH := i_TestDll(globalvar.fh_thread, windowH);
globalvar.threadactive := 1;
end;
Tips and Techniques

Verify that you can access Documaker Workstation before you customize it with RACLib/RacCo.

Run Documaker Workstation

Before beginning to use the Remote Access Library to customize Documaker Workstation, you must verify that Documaker Workstation works in your environment. You should be able to run the AFEMNW32.EXE (if using Win32 product) to access your archives and do forms entry (if needed in your environment). For more information on setting up Documaker Workstation, refer to the Documaker Workstation Administration Guide and the Documaker Workstation User Guide.

Path statement

When you write your own application to access Documaker Workstation your program will need to locate and load DLLs and INI files that Documaker Workstation uses. Your application will make calls into RACLib, which will then call into other DLL files of Documaker Workstation. Therefore, the PATH environment variable should include Documaker Workstation program files location. Verify that the PC's PATH= variable includes the program files location; such as C:\FAP\DLL.

The normal Documaker Workstation installation process updates the PC's PATH= variable to include the program directory.

Trouble Shooting

- Check your path statement. You should have C:\FAP\DLL (if that is the installed location) in your path statement. Be sure that it is in your path after a reboot.
- Check your working directory. If you are running from an icon make sure the working directory is correct.
• Check your INI settings. The INI files may be missing and you may have incorrect directory path statements.

• Check to see if the shipping sample (SAMPCO) works. If not, perhaps you need to go through installation again.

Check that the working directory and INI file settings are always accessible from your application.

Documaker Workstation uses the working directory path to locate the FSIUSER.INI and FSYS.INI files. Within these INI files are path and file settings that the system uses to locate numerous other files and resources. When you are remotely controlling Documaker Workstation, you need to be aware of this working directory system requirement. Therefore, your application should set the working directory path for RACLib/RacCo.

If your application can change its working directory while running, this can cause issues with RACLib/RacCo finding files and resources. For instance, a simple file open window can change the current working directory of an application.

RACLib/RacCo can have trouble finding files and resources if your INI files contain settings with relative paths and your application has changed its working directory while running. If you have this problem, try one of these possible solutions:

• Change your INI files to have absolute path settings

• Save and restore the working path around the process that can change your path. For instance, you could get the current working path before the file open window then restore it once finished.

• Set the working directory for RACLib/RacCo before running each remote access process (RAC)
ActiveX Object Reference
AfeProxy (IAfeProxy)

AFE proxy class

Syntax

```cpp
#include <AFEPROXY.H>

class AfeProxy : public CRacBase
```

Remarks

The proxy file is the file representation of one DAP transaction. Usually, it is the archive transaction. This file has enough information for the Power Office application to be able to get the key information for display. Additionally, if needed, it has form names, section names, field data, and so on, so the application can invoke some kind of text search mechanism. Therefore, proxy files can be thought of as another import/export file format.

There are two proxy objects in the RacCo component.

- **RacProxy**
  Queries information from the proxy file, like getting the key information.

- **AfeProxy**
  Displays an archived transaction based on the input proxy file. Additionally, it lets you create a proxy file from an archived transaction.

Overview

Related Classes

- **CRacBase**

Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CloseFile</code></td>
<td>Close proxy file.</td>
</tr>
<tr>
<td><code>GetDocHandle</code></td>
<td>Get the Documaker Workstation document handle.</td>
</tr>
<tr>
<td><code>GetIndex</code></td>
<td>Get the index value for a field.</td>
</tr>
<tr>
<td><code>LoadFile</code></td>
<td>Load a proxy file.</td>
</tr>
<tr>
<td><code>LoadFileVar</code></td>
<td>Load proxy file.</td>
</tr>
</tbody>
</table>
Details of **AfeProxy** class

**Constructors**

```c
public AfeProxy(void)
```

**CloseFile**

Close proxy file.

**Syntax**

```c
public void CloseFile(void)
```

**GetDocHandle**

Get the Documaker Workstation document handle.

**Syntax**

```c
public void GetDocHandle(long* docH)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>docH</td>
<td>Document handle.</td>
</tr>
</tbody>
</table>

**GetIndex**

Get the index value for a field.

**Syntax**

```c
public void GetIndex(BSTR fldName, BSTR* fldText)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fldName</td>
<td>Field name.</td>
</tr>
<tr>
<td>fldText</td>
<td>Field text value.</td>
</tr>
</tbody>
</table>

**LoadFile**

Load a proxy file.

**Syntax**

```c
public void LoadFile(BSTR fileName)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>File name.</td>
</tr>
</tbody>
</table>
LoadFileVar

Load proxy file.

Syntax

```java
public void LoadFileVar(VARIANT* fileName)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>File name.</td>
</tr>
</tbody>
</table>
RacEdit (IRacEdit)

Use this function to run Documaker Workstation for form entry.

Syntax

```cpp
#include <RACEDIT.H>
class RacEdit : public CRacBase
```

Overview

Remarks

Use this component to run Documaker Workstation for form entry. The component is an "in-process" DLL implemented with ATL. RACLib is responsible for the actual functionality of the component.

Related Classes

CRacBase

Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACMain</td>
<td>Starts a modal session of the form entry system.</td>
</tr>
<tr>
<td>RACThread</td>
<td>Starts a new process operating within another window.</td>
</tr>
<tr>
<td>SetINIPath</td>
<td>Sets the path that Documaker Workstation uses to locate the INI files.</td>
</tr>
<tr>
<td>SetParent</td>
<td>Use this function to set the parent window.</td>
</tr>
<tr>
<td>SetRacLibDll</td>
<td>Set path and file name for RACLib.</td>
</tr>
<tr>
<td>SetWorkingDir</td>
<td>Set working directory for RACLib DLL.</td>
</tr>
</tbody>
</table>

Details of RacEdit class

Constructors

```cpp
public RacEdit(void)
```

RACMain

Starts a modal session of the form entry system.
Syntax

```java
public void RACMain(BSTR transaction,
                  BSTR company,
                  BSTR lob,
                  BSTR policy,
                  BSTR description,
                  BSTR userid,
                  BSTR sysid,
                  long mode)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction</td>
<td>A string that represents one of the valid [TRANSACTIONS] abbreviations defined in the INI file. For instance: &quot;NB&quot; for New Business &quot;EN&quot; for Endorsement and so on.</td>
</tr>
<tr>
<td>company</td>
<td>A string that represents the first WIP key value (which is also the first component of a line from the FORM.DAT). This parameter is usually referred to as the &quot;Company&quot; value.</td>
</tr>
<tr>
<td>lob</td>
<td>A string that represents the second WIP key value (which is also the second component of a line from the FORM.DAT). This parameter is usually referred to as the &quot;Line of Business&quot; value.</td>
</tr>
<tr>
<td>policy</td>
<td>A string that represents the WIP key ID value. This parameter is usually referred to as the &quot;ID Number&quot; value.</td>
</tr>
<tr>
<td>description</td>
<td>A string that should be assigned as the WIP description value.</td>
</tr>
<tr>
<td>userid</td>
<td>A string that corresponds to a valid user ID with access to the Entry module.</td>
</tr>
<tr>
<td>sysid</td>
<td>Not currently used.</td>
</tr>
<tr>
<td>mode</td>
<td>A valid RACLib action. Currently defined: 1 = <code>AFEACTION_CREATE</code> = Create new WIP. 2 = <code>AFEACTION_UPDATE</code> = Update existing WIP.</td>
</tr>
</tbody>
</table>

Remarks

Starts a modal session that is similar to starting Documaker Workstation (`AFEMNW32.EXE`). No other initialization or termination functions need be called by the user's application.

Since this function starts a modal session, it does not return until the session completes. It will take over the main window handling and replace the existing menu. The new menu will be created from the file specified by the [ALTMENU] setting in the INI file. If omitted, the menu will default to the file specified by the [MENU] setting in the INI file.

The original menu will be restored before returning. During initialization, the login screen can appear if the user ID is not specified by the parameter and is not specified in the INI file.

This method calls `RACMain` in RACLib.

RACThread

Starts a new process operating within another window.
Syntax
public void RACThread(long hndl)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hndl</td>
<td>Parent window to start this application</td>
</tr>
</tbody>
</table>

Remarks
It is assumed that the window handle passed represents the area that must contain the program. A frame window is created inside this area. It then starts the Entry module after the usual initialization.

This method calls the RACThread in RACLib.

SetINIPath
Sets the path that Documaker Workstation uses to locate the INI files.

Syntax
public void SetINIPath(BSTR iniPath)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iniPath</td>
<td>Path name for INI files.</td>
</tr>
</tbody>
</table>

Remarks
The system will load FSISYS.INI and FSIUSER.INI from that location. This method calls RACSetIniFile in RACLib.

SetParent
Use this function to set the parent window.

Syntax
public void SetParent(HWND hwnd)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

Remarks
This method calls RACSetParent in RACLib.

SetRacLibDll
Set path and file name for RACLib.
Syntax
public void SetRacLibDll(BSTR racLibPath)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>racLibPath</td>
<td>Library path name.</td>
</tr>
</tbody>
</table>

Remarks
If this method is not used the component will default to RACW32.DLL.

SetWorkingDir
Set working directory for RACLib DLL.

Syntax
public void SetWorkingDir(BSTR workDir)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workDir</td>
<td>Working directory path.</td>
</tr>
</tbody>
</table>

Remarks
This method also sets the FSIUSER environment variable. It calls RACSetWorkingPath in RACLib.
RacImport (IRacImport)

Use this function to run Documaker Workstation for form entry.

Syntax

```c
#include <RACIMPOR.H>
class RacImport : public CRacBase
```

Overview

Remarks

Use this component to run Documaker Workstation for form entry. The component is an "in-process" DLL implemented with ATL. RACLib is responsible for the actual functionality of the component.

Related Classes

CRacBase

Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACEditData</td>
<td>View and edit a form set.</td>
</tr>
<tr>
<td>SetINIPath</td>
<td>Sets the path that Documaker Workstation uses to locate the INI files.</td>
</tr>
<tr>
<td>SetRacLibDll</td>
<td>Set path and file name for RACLib.</td>
</tr>
<tr>
<td>SetWorkingDir</td>
<td>Set working directory for RACLib DLL.</td>
</tr>
</tbody>
</table>

Details of RacImport class

Constructors

public RacImport(void)

RACEditData

View and edit a form set.

Syntax

```c
public void RACEditData(long hndl, BSTR fileName)
```
### SetINIPath

Sets the path that Documaker Workstation uses to locate the INI files.

**Syntax**

```java
public void SetINIPath(BSTR iniPath)
```

#### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iniPath</td>
<td>Path name for INI files.</td>
</tr>
</tbody>
</table>

**Remarks**

The system will load FSISYS.INI and FSIUSER.INI from that location. This method calls `RACSetIniFile` from RACLib.

### SetRacLibDll

Set path and file name for RACLib.

**Syntax**

```java
public void SetRacLibDll(BSTR racLibPath)
```

#### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>racLibPath</td>
<td>Library patch name.</td>
</tr>
</tbody>
</table>

**Remarks**

If this method is not used the component will default to racw32.dll.

### SetWorkingDir

Set working directory for RACLib DLL.

**Syntax**

```java
public void SetWorkingDir(BSTR workDir)
```

#### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workDir</td>
<td>Working directory path.</td>
</tr>
</tbody>
</table>

**Remarks**

This method also sets the FSIUSER environment variable.
RacProxy (IRacProxy)

Use this function to run Documaker Workstation for form entry.

Syntax
   #include <RACPROXY.H>
   class RacProxy : public CRacBase

Overview

Remarks

Use this component to run Documaker Workstation for form entry. The component is an "in-process" DLL implemented with ATL. RACLlib is responsible for the actual functionality of the component.

Related Classes
   CRacBase

Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CloseView</td>
<td>Close the form set</td>
</tr>
<tr>
<td>RACArchive2Proxy</td>
<td>Display transaction from archive.</td>
</tr>
<tr>
<td>RACCommand</td>
<td>Request command execution.</td>
</tr>
<tr>
<td>RACSetCaptionOff</td>
<td>Set caption off for this window.</td>
</tr>
<tr>
<td>RACSetCaptionOn</td>
<td>Set caption on for this window.</td>
</tr>
<tr>
<td>RACViewAfeProxy</td>
<td>Display archive from proxy file.</td>
</tr>
<tr>
<td>RACViewProxy</td>
<td>Display archive from proxy file.</td>
</tr>
<tr>
<td>RemoveScrollBar</td>
<td>Remove the scroll bar from window.</td>
</tr>
<tr>
<td>SetINIPath</td>
<td>Sets the path that Documaker Workstation uses to locate the INI files.</td>
</tr>
<tr>
<td>SetRacLibDll</td>
<td>Set path and file name for RACLlib.</td>
</tr>
<tr>
<td>SetWorkingDir</td>
<td>Set working directory for RACLlib DLL.</td>
</tr>
</tbody>
</table>

Details of RacProxy class

Constructors
   public RacProxy(void)
CloseView
Close the form set

Syntax
public void CloseView(void)

Remarks
Close the form set in the RACViewProxy session but don't end the session, by keeping the session alive we can avoid doing another initialization for RACLib.

RACArchive2Proxy
Display transaction from archive.

Syntax
public void RACArchive2Proxy(long hwnd,
BSTR Key1,
BSTR Key2,
BSTR KeyID,
short startpage,
BSTR* proxyfile)

Parameter | Description
----------|----------------
hwnd       | The parent window to contain this application.
Key1       | Search data for Key1.
Key2       | Search data for Key2.
KeyID      | Search data for KeyID.
startpage  | Page to begin display of form set.
proxyfile  | Character pointer that contains name of the proxy file

Remarks
Display an archived transaction and allow user to store a proxy file pointing to the transaction in archive. Transactions are filtered the same as RACRetrieveArchive.

This method calls RACArchive2Proxy in RACLib.

RACCommand
Request command execution.

Syntax
public void RACCommand(long cmd)

Parameter | Description
----------|----------------
**Remarks**

This will send the main window a command message corresponding to the command ID passed as a parameter. Most often, this function will be used during a non-modal session to pass in the IDs associated with WM_COMMAND messages. You must initialize the system before you call this function.

This method calls [RACCommand](#) in RACLlib.

**RACSetCaptionOff**

Set caption off for this window.

**Syntax**

```java
public void RACSetCaptionOff(void)
```

**Remarks**

Set caption off for this window. This method calls [RACSetCaptionOff](#) in RACLlib.

**RACSetCaptionOn**

Set caption on for this window.

**Syntax**

```java
public void RACSetCaptionOn(void)
```

**Remarks**

Set caption on for this window. This method calls [RACSetCaptionOn](#) in RACLlib.

**RACViewAfeProxy**

Display archive from proxy file.

**Syntax**

```java
public void RACViewAfeProxy(long hwnd, AfeProxy* obj, short page, short isChildWindow)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Must be a valid window handle to contain form set display.</td>
</tr>
<tr>
<td>page</td>
<td>Page to begin display of form set.</td>
</tr>
</tbody>
</table>

**Remarks**

This method calls [RACViewProxy](#) in RACLlib.
RACViewProxy
Display archive from proxy file.

Syntax
   public void RACViewProxy(long hwnd,
                        BSTR filename,
                        short page,
                        short isChildWindow)

Parameter | Description
----------|----------------
hwnd       | Must be a valid window handle to contain form set display.
filename   | path to a valid proxy file
page       | page to begin display of form set

Remarks
This method calls RACViewProxy in RACLib.

RemoveScrollBar
Remove the scroll bar from window.

Syntax
   public void RemoveScrollBar(void)

Remarks
Remove the scroll bar from window. This method calls RACRemoveScrollBar in RACLib.

SetINIPath
Sets the path that Documaker Workstation uses to locate the INI files.

Syntax
   public void SetINIPath(BSTR iniPath)

Parameter | Description
-----------|----------------
iiniPath   | Path name for INI files.

Remarks
The system will load FSISYS.INI and FSIUSER.INI from that location. This method calls RACSetIniFile from RACLib.

SetRacLibDll
Set path and file name for RACLib.
Syntax
    public void SetRacLibDll(BSTR racLibPath)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>racLibPath</td>
<td>Library patch name.</td>
</tr>
</tbody>
</table>

Remarks
If this method is not used the component will default to racw32.dll.

SetWorkingDir
Set working directory for RACLib DLL.

Syntax
    public void SetWorkingDir(BSTR workDir)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workDir</td>
<td>Working directory path.</td>
</tr>
</tbody>
</table>

Remarks
This method also sets the FSIUSER environment variable.
RacVw (IRacVw)

Use this function to view archived form sets.

Syntax

```
#include <RACVW.H>
class RacVw : public CRacBase
```

Overview

Remarks

Use this component to view archived form sets. The component is an "in-process" DLL implemented with ATL. RACLlib is responsible for the actual functionality of the component.

Related Classes

CRacBase

Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetINIPath</td>
<td>Sets the path that Documaker Workstation uses to locate the INI files.</td>
</tr>
<tr>
<td>SetParent</td>
<td>Use this function to set the parent window.</td>
</tr>
<tr>
<td>SetRacLibDll</td>
<td>Set path and file name for RACLlib.</td>
</tr>
<tr>
<td>SetWorkingDir</td>
<td>Set working directory for RACLlib DLL.</td>
</tr>
<tr>
<td>ViewArcKey</td>
<td>Find transactions based on CAR Key.</td>
</tr>
<tr>
<td>ViewFilename</td>
<td>View transaction stored in the export file.</td>
</tr>
<tr>
<td>ViewByKey</td>
<td>View the form set that matches the key values supplied.</td>
</tr>
</tbody>
</table>

Details of RacVw class

Constructors

```
public RacVw(void)
```

SetINIPath

Sets the path that Documaker Workstation uses to locate the INI files.
**Syntax**

```java
public void SetINIPath(BSTR iniPath)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iniPath</td>
<td>Path name for INI files.</td>
</tr>
</tbody>
</table>

**Remarks**

The system will load FSISYS.INI and FSIUSER.INI from that location. This method calls `RACSetIniFile` from RACLlib.

**SetParent**

Use this function to set the parent window.

**Syntax**

```java
public void SetParent(HWND hwnd)
```

**Remarks**

This method calls `RACSetParent` from RACLlib.

**SetRacLibDll**

Set path and file name for RACLlib.

**Syntax**

```java
public void SetRacLibDll(BSTR racLibPath)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>racLibPath</td>
<td>Library patch name.</td>
</tr>
</tbody>
</table>

**Remarks**

If this method is not used the component will default to racw32.dll.

**SetWorkingDir**

Set working directory for RACLlib DLL.

**Syntax**

```java
public void SetWorkingDir(BSTR workDir)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workDir</td>
<td>Working directory path.</td>
</tr>
</tbody>
</table>
Remarks
This method also sets the FSIUSER environment variable.

ViewByArcKey
Find transactions based on CAR Key.

Syntax
   public void ViewByArcKey(BSTR carkey,
                             short page,
                             short location)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>carkey</td>
<td>Key value of CAR key.</td>
</tr>
<tr>
<td>page</td>
<td>Page number.</td>
</tr>
<tr>
<td>location</td>
<td>Screen location.</td>
</tr>
<tr>
<td></td>
<td>0 = Left half display (default)</td>
</tr>
<tr>
<td></td>
<td>1 = Right half display</td>
</tr>
<tr>
<td></td>
<td>2 = Top half display</td>
</tr>
<tr>
<td></td>
<td>3 = Bottom half display</td>
</tr>
<tr>
<td></td>
<td>4 = normal window</td>
</tr>
<tr>
<td></td>
<td>5 = maximized window</td>
</tr>
</tbody>
</table>

Remarks
This method calls RACViewByKey from RACLib.

ViewByFilename
View the transaction stored in the export file.

Syntax
   public void ViewByFilename(BSTR filename,
                               short page,
                               short location)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of export file.</td>
</tr>
<tr>
<td>page</td>
<td>Page number.</td>
</tr>
<tr>
<td>location</td>
<td>Screen location.</td>
</tr>
<tr>
<td></td>
<td>0 = Left half display (default)</td>
</tr>
<tr>
<td></td>
<td>1 = Right half display</td>
</tr>
<tr>
<td></td>
<td>2 = Top half display</td>
</tr>
<tr>
<td></td>
<td>3 = Bottom half display</td>
</tr>
<tr>
<td></td>
<td>4 = normal window</td>
</tr>
<tr>
<td></td>
<td>5 = maximized window</td>
</tr>
</tbody>
</table>
Remarks
This method calls RACViewData from RACLlib.

ViewByKey
View the form set that matches the key values supplied.

Syntax

\[
\text{public void ViewByKey(BSTR Key1, BSTR Key2, BSTR KeyID, short page, short location);}
\]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key1</td>
<td>Search data for Key1. Usually known as company.</td>
</tr>
<tr>
<td>Key2</td>
<td>Search data for Key2. Usually known as line of business.</td>
</tr>
<tr>
<td>KeyID</td>
<td>Search data for KeyID. Usually known as policy number.</td>
</tr>
<tr>
<td>page</td>
<td>set beginning page</td>
</tr>
</tbody>
</table>
| location  | Set screen location.  
            | 0 = Left half display (default)  
            | 1 = Right half display  
            | 2 = Top half display  
            | 3 = Bottom half display  
            | 4 = normal window  
            | 5 = maximized window |

Remarks
Find transactions that match Key1, Key2, and KeyID. The fields that are used for these keys are defined in the INI file. See the ArcRet control group options Key1, Key2, and KeyID. Method calls RACRetrieveArchive from RACLlib.
Class Reference
AfeLib

Interface class to *AFELib*.

**Syntax**

```c
#include <CAFELIB.H>
class AfeLib
```

**Overview**

**Member Functions**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFECloseProxyFile</td>
<td>Close proxy file.</td>
</tr>
<tr>
<td>AFEGetProxyFieldData</td>
<td>Get proxy field data.</td>
</tr>
<tr>
<td>AFEIndex2Fld</td>
<td>Get Field given an index</td>
</tr>
<tr>
<td>AFEOpenProxyFile</td>
<td>Open proxy file.</td>
</tr>
</tbody>
</table>

**Details of AfeLib class**

**Constructors**

```c
public AfeLib(void)
```

**AFECloseProxyFile**

Close proxy file.

**Syntax**

```c
public void AFECloseProxyFile(VMMHANDLE docH)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>docH</td>
<td>Document handle.</td>
</tr>
</tbody>
</table>

**AFEGetProxyFieldData**

Get proxy field data.
Syntax

```
public int AFEGetProxyFieldData(VMMHANDLE docH,
                              char* fldName,
                              char* flddata,
                              char** pAttr,
                              int fldsize)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>docH</td>
<td>Document handle.</td>
</tr>
<tr>
<td>fldName</td>
<td>Field name.</td>
</tr>
<tr>
<td>flddata</td>
<td>Field data.</td>
</tr>
<tr>
<td>pAttr</td>
<td>Attributes.</td>
</tr>
<tr>
<td>fldsize</td>
<td>Field size.</td>
</tr>
</tbody>
</table>

AFEIndex2Fld

Get Field given an index

Syntax

```
public void AFEIndex2Fld(enum ProxyIndexFields fldno,
                         char* fldName,
                         short fldSize)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fldno</td>
<td>Field number.</td>
</tr>
<tr>
<td>fldName</td>
<td>Field name.</td>
</tr>
<tr>
<td>fldSize</td>
<td>Field size.</td>
</tr>
</tbody>
</table>

AFEOpenProxyFile

Open proxy file.

Syntax

```
public VMMHANDLE AFEOpenProxyFile(public VMMHANDLE AFEOpenProxyFile@char* fileName)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>File name.</td>
</tr>
</tbody>
</table>
CRacBase

Base class for common classes used by RacCo.

Syntax

```cpp
#include <RACBASE.H>
class CRacBase
```

Overview

Remarks

The ActiveX components are derived from this class. It keeps track of the parent window handle and the CRacLib interface class to RACLib.

Related Classes

AfeProxy | RacEdit | RaclImport | RacProxy | RacVw

Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ConvertToUnicode</code></td>
<td>Convert a string to Unicode.</td>
</tr>
<tr>
<td><code>DestroyProblemWindow</code></td>
<td>Destroy any problem window that may have been left.</td>
</tr>
<tr>
<td><code>Error</code></td>
<td>Error function.</td>
</tr>
<tr>
<td><code>GetMyHab</code></td>
<td>Get the current application instance handle.</td>
</tr>
<tr>
<td><code>GetMyWindow</code></td>
<td>Get the current active window.</td>
</tr>
<tr>
<td><code>makeWindow</code></td>
<td>Create a window for Documaker Workstation.</td>
</tr>
<tr>
<td><code>RACCommand</code></td>
<td>Request command execution.</td>
</tr>
<tr>
<td><code>RACSetCaptionOff</code></td>
<td>Set caption off for this window.</td>
</tr>
<tr>
<td><code>RACSetCaptionOn</code></td>
<td>Set caption on for this window.</td>
</tr>
<tr>
<td><code>SetINIPath</code></td>
<td>Sets the path that Documaker Workstation uses to locate the INI files.</td>
</tr>
<tr>
<td><code>SetParent</code></td>
<td>Use this function to set the parent window.</td>
</tr>
<tr>
<td><code>SetRacLibDll</code></td>
<td>Set path and file name for RACLib.</td>
</tr>
<tr>
<td><code>SetWorkingDir</code></td>
<td>Set working directory for RACLib DLL.</td>
</tr>
</tbody>
</table>
Details of CRacBase class

Constructors
protected CRacBase(void)

Protected Data Members

hwndParent
    HWND hwndParent

Remarks
Parent window handle.

RacLib
    CRacLib* RacLib

Remarks
Interface to RACLib API functions.

ConvertToUnicode
Convert a string to Unicode.

Syntax
    protected OLECHAR* ConvertToUnicode(char* szA)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>szA</td>
<td>String to convert.</td>
</tr>
</tbody>
</table>

DestroyProblemWindow
Destroy any problem window that may have been left.

Syntax
    protected void DestroyProblemWindow(CRacLib* pRacLib,
                                           CRacErr& err)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pRacLib</td>
<td>Pointer to a CRacLib class.</td>
</tr>
<tr>
<td>err</td>
<td>Reference to error class.</td>
</tr>
</tbody>
</table>
Remarks
Destroy any problem window that may have been left by RACTerminate. This has happened before and while RACTerminate was fixed.

Error
Error function.

Syntax
protected virtual HRESULT Error(LPCSTR err) pure

Remarks
This method is a virtual method.

GetMyHab
Get the current application instance handle.

Syntax
protected HINSTANCE GetMyHab(void)

GetMyWindow
Get the current active window.

Syntax
protected HWND GetMyWindow(void)

Remarks
This function tries to handle the situation if the current window is not available.

makeWindow
Create a window for Documaker Workstation.

Syntax
protected HWND makeWindow(HWND parentWindowH)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parentWindowH</td>
<td>Handle of window to use as parent of new window.</td>
</tr>
</tbody>
</table>

RACCommand
Request command execution.
Syntax
protected HRESULT RACCommand(long cmd)

Parameter | Description
---------|------------
cmd       | Command ID

Remarks
This sends the main window a command message corresponding to the command ID passed as a parameter. This method calls RACCommand in RACLib.

RACSetCaptionOff
Set the caption off for this window.

Syntax
protected HRESULT RACSetCaptionOff(long hwnd)

Parameter | Description
---------|------------
hwnd      | Handle of window.

Remarks
Set the caption off for this window. This method calls RACSetCaptionOff in RACLib.

RACSetCaptionOn
Set the caption on for this window.

Syntax
protected HRESULT RACSetCaptionOn(long hwnd)

Parameter | Description
---------|------------
hwnd      | Handle of window.

Remarks
Set the caption on for this window. This method calls RACSetCaptionOn in RACLib.

SetINIPath
Sets the path that Documaker Workstation uses to locate the INI files.

Syntax
protected void SetINIPath(BSTR iniPath)
### Parameter iniPath

**Description**
Path name for INI files.

### Remarks

The system loads the FSISYS.INI and FSIUSER.INI files from that location. This method calls `RACSetIniFile` in RACLib.

### SetParent

Use this function to set the parent window.

#### Syntax

```java
protected void SetParent(long hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

### Remarks

This method calls `RACSetParent` in RACLib.

### SetRacLibDll

Set path and file name for RACLib.

#### Syntax

```java
protected void SetRacLibDll(BSTR racLibPath)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>racLibPath</td>
<td>Library path name.</td>
</tr>
</tbody>
</table>

### Remarks

If this method is not used the component will default to `RACW32.DLL`.

### SetWorkingDir

Set working directory for RACLib DLL.

#### Syntax

```java
protected void SetWorkingDir(BSTR workDir)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workDir</td>
<td>Working directory path.</td>
</tr>
</tbody>
</table>
Remarks

This method also sets the FSIUSER environment variable. It calls RACSetWorkingPath in RACLlib.
CRacErr

Error message class

Syntax
#include <CRACERR.H>
class CRacErr

Overview

Remarks
Translates RACLib error messages to text as well any other error number we need to define.

The CRacLib constructor throws an object of this type if it is unable to load a portion of the DLL.

Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDescription</td>
<td>Get the error description.</td>
</tr>
<tr>
<td>GetErrNum</td>
<td>Get the error code number.</td>
</tr>
</tbody>
</table>

Details of CRacErr class

Constructors

Construct Error object with number.
public CRacErr(int RacErrNo, CRacLib* pRacLib)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RacErrNo</td>
<td>Error code number</td>
</tr>
<tr>
<td>pRacLib</td>
<td>Interface class to RACLib. This constructor calls RACGetDescription in RACLib.</td>
</tr>
</tbody>
</table>

Remarks
This constructor translates RACLib error messages to text as well any other error number we need to define.
Construct Error object with number.
   public CRacErr(int RacErrNo)

Construct Error object with a string.
   public CRacErr(char* desc)
   public CRacErr(void)

GetDescription

Get the error description.

Syntax
   public LPCSTR GetDescription(void)

GetErrNum

Get the error code number.

Syntax
   public int GetErrNum(void)
CRacLib

Interface class to RACLib.

Syntax

```c
#include <CRACLIB.H>
class CRacLib
```

Overview

Remarks

This class is responsible for making calls into RACLib API functions. All of the ActiveX components have an instance of this class.

Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACArchive2Proxy</td>
<td>Display transaction from archive.</td>
</tr>
<tr>
<td>RACClose</td>
<td>Close.</td>
</tr>
<tr>
<td>RACCommand</td>
<td>Request command execution.</td>
</tr>
<tr>
<td>RACEditData</td>
<td>View form set specified.</td>
</tr>
<tr>
<td>RACGetDescription</td>
<td>Return description of error message</td>
</tr>
<tr>
<td>RACGetMainWindow</td>
<td>Return frame window.</td>
</tr>
<tr>
<td>RACGetSession</td>
<td>Get the session for this window.</td>
</tr>
<tr>
<td>RACInitCtrl</td>
<td>Initialize the Entry system (menu and accelerator control).</td>
</tr>
<tr>
<td>RACLoadIni</td>
<td>Load the INI file.</td>
</tr>
<tr>
<td>RACLoadMenu</td>
<td>This function will query the INI file for the MEN.RES compatible file to load.</td>
</tr>
<tr>
<td>RACMain</td>
<td>Starts a modal session of the Entry system.</td>
</tr>
<tr>
<td>RACNeed2Terminate</td>
<td>Gets terminate state.</td>
</tr>
<tr>
<td>RACRemoveScrollbar</td>
<td>Remove the scroll bar from window.</td>
</tr>
<tr>
<td>RACRestoreMenu</td>
<td>Restore the Original Menu.</td>
</tr>
<tr>
<td>RACRetrieveArchive</td>
<td>View form set specified.</td>
</tr>
<tr>
<td>RACSetCaptionOff</td>
<td>Set the caption off for this window.</td>
</tr>
<tr>
<td>RACSetCaptionOn</td>
<td>Set the caption on for this window.</td>
</tr>
<tr>
<td>RACSetCurSession</td>
<td>Set this window to be the current session.</td>
</tr>
<tr>
<td>RACSetIniFile</td>
<td>Sets the path that Documaker Workstation uses to locate the INI files.</td>
</tr>
</tbody>
</table>
**RACSetProxyHandle**  Set the document handle of a proxy file for **RACViewProxy**.

**RACSetWorkingPath**  Set working directory for RACLlib DLL.

**RACTerminate**  Terminate an Entry session.

**RACTermSession**  Terminate an Entry session.

**RACThread**  Starts a new process operating within another window.

**RACViewByKey**  View form set specified.

**RACViewData**  View form set specified.

**RACViewProxy**  Display archive from proxy file.

**SetParent**  Use this function to set the parent window.

### Details of CRacLib class

#### Constructors

public CRacLib(void)

public CRacLib(char* dllname)

#### Public Data Members

**isDLLLoaded**

BOOL isDLLLoaded

**Remarks**

Indicates if RACLlib DLL was loaded.

#### RACArchive2Proxy

Display transaction from archive.

**Syntax**

```c
public void RACArchive2Proxy(HWND hwnd, 
    char* Key1, 
    char* Key2, 
    char* KeyID, 
    int startpage, 
    char* proxyfilename)
```

**Parameter** | **Description**
--- | ---
hwnd | The parent window to contain this application.
Key1 | Search data for Key1.
Key2 | Search data for Key2.
KeyID | Search data for KeyID.
Remarks

This method calls RACArchive2Proxy in RACLib. Display an archived transaction and allow user to store a proxy file pointing to the transaction in archive. Transactions are filtered the same as RACRetrieveArchive.

RACClose

Close.

Syntax

public void RACClose(HWND hwnd,
                     BOOL querysave,
                     BOOL open)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
<tr>
<td>querysave</td>
<td>Boolean to indicate if a message should ask to save current form set.</td>
</tr>
<tr>
<td>open</td>
<td>Boolean to indicate if it should be opened.</td>
</tr>
</tbody>
</table>

Remarks

This method calls RACClose in RACLib.

Performs AFEClose, thus closing the form set and any windows that were associated with that form set. This lets you use the same session again without doing RACInit. You should call this function before you perform a consecutive view function.

RACCommand

Request command execution.

Syntax

public void RACCommand(int cmd)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd</td>
<td>Command ID</td>
</tr>
</tbody>
</table>

Remarks

This will send the main window a command message corresponding to the command ID passed as a parameter. Most often, this function will be used during a non-modal session to pass in the IDs associated with WM_COMMAND messages. You must initialize the system before you call this function.

This method calls RACCommand in RACLib.
RACEditData

View form set specified.

Syntax

public void RACEditData(HWND hwndParent,
char* filename)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwndParent</td>
<td>The parent window to contain this application.</td>
</tr>
<tr>
<td>filename</td>
<td>The name of a file to import.</td>
</tr>
</tbody>
</table>

Remarks

It is assumed that the window handle passed represents another application's main window. We'll create a frame window to contain the form set view and start the entry system after usual initialization.

This method calls RACEditData in RACLib.

RACGetDescription

Return description of error message

Syntax

public char* RACGetDescription(int ErrNo)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrNo</td>
<td>Error code.</td>
</tr>
</tbody>
</table>

Remarks

Get the description or message of an error code. This method calls RACGetDescription in RACLib.

RACGetMainWindow

Return frame window.

Syntax

public HWND RACGetMainWindow(void)

Remarks

Return the top level window handle created by RACThread. This method calls RACGetFrameWindow in RACLib.

RACGetSession

Get the session for this window
Syntax

public void* RACGetSession(HWND hwnd)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window</td>
</tr>
</tbody>
</table>

Remarks

Retrieve the session that matches on the main frame's window handle. This method calls RACGetSession in RACLib.

RACInitCtrl

Initialize the Entry system (menu and accelerator control).

Syntax

public void RACInitCtrl(HWND hwnd, char* userid, char* sysid, BOOL loadMenu, BOOL loadAccel)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>The external application's main (highest level) window.</td>
</tr>
<tr>
<td>userid</td>
<td>A pointer to a null-terminated string that corresponds to a valid user ID with access to the entry system.</td>
</tr>
<tr>
<td>sysid</td>
<td>Future expansion - not currently used.</td>
</tr>
<tr>
<td>loadMenu</td>
<td>If true, the menu is loaded.</td>
</tr>
<tr>
<td>loadAccel</td>
<td>If true, the accelerator table is loaded.</td>
</tr>
</tbody>
</table>

Remarks

This method calls RACInitCtrl in RACLib. This causes the entry system to initialize. Initialization includes memory management, FAP, WIP, user login, and the help system.

This function will call RACWorkingPath before starting the initialization process. If an error occurs, RACTerminate will be called and the path restored.

If the process initializes correctly and the load menu flag is true; the defined "menu" will be loaded (although not activated). In addition, the default accelerators will be loaded if the load accelerator flag is true.

If the calling application does not call RACModal, then it should eventually call RACTerminate to shutdown correctly.

During initialization, the login screen can appear if the user ID is not specified by a parameter and is not specified in the INI file.
RACLoadIni
Load the INI file.

Syntax
public void RACLoadIni(void)

Remarks
Load the INI file. The file name is defaulted to FSIUSER.INI if RACSetIniFile did not supply it.
This method calls RACLoadIni in RACLib.

RACLoadMenu
This function will query the INI file for the MEN.RES compatible file to load.

Syntax
public void RACLoadMenu(void)

Remarks
This method calls RACLoadMenu in RACLib. This function will query the INI file for the MEN.RES compatible file to load. The following INI options will be queried in the order shown. Once a valid option has been determined the menu is created but not activated. A call to RACEnableMenu is required to activate the menu.

Order of search to determine menu to load:
[AltMenu]
File
If not located then use:
[MENU]
File
If not located the file MEN.RES will be attempted.
If no menu is located, none will be created and an error is returned.
RACRestoreMenu will restore the original menu. RACTerminate automatically calls this function.

RACMain
Starts a modal session of the form entry system.
Syntax

```java
public void RACMain(HWND hwnd,
    char* transaction,
    char* company,
    char* lob,
    char* policy,
    char* description,
    char* userid,
    char* sysid,
    long mode)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
<tr>
<td>transaction</td>
<td>A string that represents one of the valid [TRANSACTIONS] abbreviations defined in the INI file. For instance: &quot;NB&quot; for New Business &quot;EN&quot; for Endorsement and so on.</td>
</tr>
<tr>
<td>company</td>
<td>A string that represents the first WIP key value (which is also the first component of a line from the FORM.DAT). This parameter is usually referred to as the &quot;Company&quot; value.</td>
</tr>
<tr>
<td>lob</td>
<td>A string that represents the second WIP key value (which is also the second component of a line from the FORM.DAT). This parameter is usually referred to as the &quot;Line of Business&quot; value.</td>
</tr>
<tr>
<td>policy</td>
<td>A string that represents the WIP key ID value. This parameter is usually referred to as the &quot;ID Number&quot; value.</td>
</tr>
<tr>
<td>description</td>
<td>A string that should be assigned as the WIP description value.</td>
</tr>
<tr>
<td>userid</td>
<td>A string that corresponds to a valid user ID with access to the Entry module.</td>
</tr>
<tr>
<td>sysid</td>
<td>Not currently used.</td>
</tr>
<tr>
<td>mode</td>
<td>A valid RACLib action. Currently defined: 1 = AFEACTION_CREATE = Create new WIP. 2 = AFEACTION_UPDATE = Update existing WIP.</td>
</tr>
</tbody>
</table>

Remarks

Starts a modal session that is similar to starting Documaker Workstation (AFEMNW32.EXE). No other initialization or termination functions need be called by the user’s application.

Since this function starts a modal session, it does not return until the session completes. It will take over the main window handling and replace the existing menu. The new menu will be created from the file specified by the [ALTMENU] setting in the INI file. If omitted, the menu will default to the file specified by the [MENU] setting in the INI file.

The original menu will be restored before returning. During initialization, the login screen can appear if the user ID is not specified by the parameter and is not specified in the INI file.

This method calls RACMain in RACLib.

RACNeed2Terminate

Gets terminate state.
Syntax
public BOOL RACNeed2Terminate(void)

Remarks
If no RACLib termination is needed, TRUE (1) will be returned, otherwise FALSE (0) is returned.

RACRemoveScrollbar
Remove the scroll bar from a window.

Syntax
public void RACRemoveScrollbar(HWND hwnd)

Parameter | Description
--- | ---
hwnd | Handle of window.

Remarks
Remove the scroll bar from a window. This method calls RACRemoveScrollbar in RACLib.

RACRestoreMenu
Restore the original menu.

Syntax
public void RACRestoreMenu(void)

Remarks
This method calls RACRestoreMenu in RACLib.

RACRetrieveArchive
View form set specified.

Syntax
public void RACRetrieveArchive(HWND hwndParent, char* Key1, char* Key2, char* KeyID, int startpage, int location)

Parameter | Description
--- | ---
hwndParent | The parent window to contain this application.
Key1 | Search data for Key1.
Key2 | Search data for Key2.
---|---
KeyID | Search data for KeyID.
startpage | set beginning page

| location | Set screen location.  
|---|---
| 0 | Left half display (default)  
| 1 | Right half display  
| 2 | Top half display  
| 3 | Bottom half display  
| 4 | normal window  
| 5 | maximized window

**Remarks**

This method calls [RACRetrieveArchive](#) in RACLlib. It is assumed that the window handle passed represents another application’s main window. We'll create a frame window to contain the form set view and start the entry system after the usual initialization.

See the ArcRet control group to determine which fields are matched with Key1, Key2, and KeyID.

**RACSetCaptionOff**

Set caption off for this window.

**Syntax**

```java
public void RACSetCaptionOff(HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

**Remarks**

Set caption off for this window. This method calls [RACSetCaptionOff](#) in RACLlib.

**RACSetCaptionOn**

Set caption on for this window.

**Syntax**

```java
public void RACSetCaptionOn(HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

**Remarks**

Set caption on for this window. This method calls [RACSetCaptionOn](#) in RACLlib.
**RACSetCurSession**

Set this window to be the current session.

**Syntax**

```java
public void RACSetCurSession(HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window</td>
</tr>
</tbody>
</table>

**Remarks**

Set the main frame’s window handle (hwndMainFrame variable) if we have a valid session for the handle passed. This method calls `RACSetCurSession` in RACLib.

**RACSetIniFile**

Sets the path that Documaker Workstation uses to locate the INI files.

**Syntax**

```java
public void RACSetIniFile(char* iniPath)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iniPath</td>
<td>Path name for INI files.</td>
</tr>
</tbody>
</table>

**Remarks**

The system loads the FSISYS.INI and FSIUSER.INI files from that location. This method calls `RACSetIniFile` in RACLib.

**RACSetProxyHandle**

Set the document handle of a proxy file for `RACViewProxy`.

**Syntax**

```java
public void RACSetProxyHandle(VMMHANDLE docH)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>docH</td>
<td>Document handle.</td>
</tr>
</tbody>
</table>

**Remarks**

Set the document handle of a proxy file for `RACViewProxy`.

This method calls `RACSetProxyHandle` in RACLib.
RACSetWorkingPath

Set working directory for RACLib DLL.

Syntax

public void RACSetWorkingPath(char* workDir)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workDir</td>
<td>Working directory path.</td>
</tr>
</tbody>
</table>

Remarks

This method also sets the FSIUSER environment variable. It calls RACSetWorkingPath in RACLib.

RACTerminate

Terminate Entry Session.

Syntax

public void RACTerminate(void)

Remarks

This method calls RACTerminate in RACLib. This will cause the session to release all resources used by the entry system. You must initialize the system before you call this function.

A non-modal session relies upon the controlling application to request termination. Exiting without terminating the system can cause unpredictable results.

RACTermSession

Terminate Entry Session.

Syntax

public void RACTermSession(BOOL mode, HWND hwnd)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>The parameter mode should be TRUE if the entry system should ask the user to save (if necessary) and FALSE to cause the system to shutdown without asking.</td>
</tr>
<tr>
<td>hwnd</td>
<td>Handle of window</td>
</tr>
</tbody>
</table>

Remarks

This method calls RACTermSession in RACLib.
This will cause the session to release all resources used by the entry system. You must initialize the system before you call this function.

A non-modal session relies upon the controlling application to request termination. Exiting without terminating the system can cause unpredictable results.

**RACThread**

Starts a new process operating within another window.

**Syntax**

```java
public void RACThread(HWND hndl)
```

**Parameter** | **Description**
---|---
*hndl* | Parent window to start this application

**Remarks**

It is assumed that the window handle passed represents the area that must contain the program. A frame window will be created inside this area. Then starts the Entry module after the usual initialization.

This method calls the `RACThread` in RACLib.

**RACViewByKey**

View form set specified.

**Syntax**

```java
public void RACViewByKey(HWND hwndParent, char* key, int startpage, int location)
```

**Parameter** | **Description**
---|---
*HwndParent* | The parent window to contain this application.
*Key* | The archives key for the transaction.
*Startpage* | The page number to start display (defaults to 1)
*Location* | Screen location.
0 = Left half display (default)
1 = Right half display
2 = Top half display
3 = Bottom half display
4 = normal window
5 = maximized window
**Remarks**

This method calls RACViewByKey in RACLib. It is assumed that the window handle passed represents another application’s main window. We'll create a frame window to contain the form set view and start the entry system after usual initialization.

**RACViewData**

View form set specified.

**Syntax**

```java
public void RACViewData(HWND hwndParent,
char* filename,
int startpage,
int location)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwndParent</td>
<td>The parent window to contain this application.</td>
</tr>
<tr>
<td>filename</td>
<td>The name of a file to import.</td>
</tr>
<tr>
<td>startpage</td>
<td>The page number to start display (defaults to 1)</td>
</tr>
<tr>
<td>location</td>
<td>Screen location.</td>
</tr>
<tr>
<td></td>
<td>0 = Left half display (default)</td>
</tr>
<tr>
<td></td>
<td>1 = Right half display</td>
</tr>
<tr>
<td></td>
<td>2 = Top half display</td>
</tr>
<tr>
<td></td>
<td>3 = Bottom half display</td>
</tr>
<tr>
<td></td>
<td>4 = normal window</td>
</tr>
<tr>
<td></td>
<td>5 = maximized window</td>
</tr>
</tbody>
</table>

**Remarks**

This method calls RACViewData in RACLib. It is assumed that the window handle passed represents another application's main window. We'll create a frame window to contain the form set view and start the entry system after usual initialization.

**RACViewProxy**

Display archive from proxy file.

**Syntax**

```java
public void RACViewProxy(HWND hwnd,
char* filename,
int startpage)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>must be a valid window handle to contain form set display</td>
</tr>
<tr>
<td>filename</td>
<td>path to a valid proxy file</td>
</tr>
<tr>
<td>startpage</td>
<td>page to begin display of form set</td>
</tr>
</tbody>
</table>
Remarks
This method calls RACViewProxy in RACLib.

SetParent
Use this function to set the parent window.

Syntax
public void SetParent(HWND hwnd)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

Remarks
This method calls RACSetParent in RACLib.
RACArchive2Proxy

Display transaction from archive.

Syntax

~~~
int RACArchive2Proxy(HWND hwnd,
char* Key1,
char* Key2,
char* KeyID,
int startpage,
char* proxyfilename)
~~~

Parameter | Description
---|---
hwnd | The parent window to contain this application.
Key1 | Search data for Key1.
Key2 | Search data for Key2.
KeyID | Search data for KeyID.
startpage | Page to begin display of form set.
proxyfilename | Character pointer that contains name of the proxy file

Remarks

Display an archived transaction and allow user to store a proxy file pointing to the transaction in archive. Transactions are filtered the same as RACRetrieveArchive.

Returns

RAC_SUCCESS (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

~~~
int _RACAPI RACArchive2Proxy(HWND hwnd,
char *Key1,
char *Key2,
char *KeyID,
int startpage,
char *proxyfilename);
~~~

Visual Basic Declaration

~~~
Private Declare Function RACArchive2Proxy Lib "racw32.dll" _
(ByVal hwnd As Long, _
ByVal Key1 As String, _
ByVal Key2 As String, _
ByVal KeyID As String, _
ByVal startpage As Integer)
~~~
typedef int (_RACAPIPTR RACARCHIVE2PROXY) (HWND hwnd, 
char *Key1, 
char *Key2, 
char *KeyID, 
int startpage, 
char *proxyfile);

See Also

RACSaveAsProxy, RACViewProxy

Calls

RACGetSession, RACGetStatus, RACModalEx, RACProcessError, RACRetrieve, RACSetCurSession, RACSetStatus
RACClose

Close.

Syntax

int RACClose(HWND hwnd,
             unsigned int querysave,
             unsigned int open)

Parameter | Description                      |
----------|---------------------------------|
hwnd      | Handle of window.               |
querysave | Boolean to indicate if a message should ask to save current form set. |
open      | Boolean to indicate if it should be opened. |

Remarks

Performs AFEClose, thus closing the form set and any windows that were associated with that form set. This lets you use the same session again without doing RACInit. You should call this function before performing a consecutive view function.

Files

Include: racapi.h
Source: RACTERM.C

Code Declarations

int _RACAPI RACClose(HWND hwnd, BOOL querysave, BOOL open);

Visual Basic Declaration

Private Declare Function RACClose Lib "racw32.dll" -
  (ByVal hwnd As Long,
   ByVal querysave As Long,
   ByVal openflag As String) As Long

typedef

typedef void (_RACAPIPTR RACCLOSE)(HWND hwnd,
                                  BOOL queryclose,
                                  BOOL open);

Calls

RACGetSession
RACCommand

Request command execution.

Syntax

int RACCommand(int comID)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>comID</td>
<td>Command ID</td>
</tr>
</tbody>
</table>

Remarks

This will send the main window a command message corresponding to the command ID passed as a parameter. Most often, this function will be used during a non-modal session to pass in the IDs associated with WM_COMMAND messages. You must initialize the system before you call this function.

Returns

The return value will be that returned from a WndProc in the given operating environment.

Files

Include: racapi.h
Source: RACPROC.C

Code Declarations

int _RACAPI RACCommand(int comID);

Visual Basic Declaration

Private Declare Function RACCommand Lib "racw32.dll" _
(ByVal comID As Long) As Long

typedef

typedef int (_RACAPIPTR RACCOMMAND)(int comID);

Example

The following code is an excerpt from an external program.

switch ( msg ) {
    case WM_COMMAND:
        rval = RACCommand(LOWORD(wParam));
        ....
}

Calls

RACCountPages
RACCountPages

Count form set pages

Syntax
    int RACCountPages(void)

Remarks
This will return the number of pages in the currently open form set.

Returns
This returns the number of pages. A zero return value indicates that the form set is empty or invalid.

Files
Include: racapi.h
Source: RACDATA.C

Code Declarations
    int _RACAPI RACCountPages(void);

Visual Basic Declaration
    Private Declare Function RACCountPages Lib "racw32.dll" () As Long
RACCreate

Create a New WIP Entry

Syntax

```c
int RACCreate(char* transaction,
    char* company,
    char* lob,
    char* policy,
    char* description)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction</td>
<td>A null-terminated string pointer that represents one of the valid [TRANSACTIONS] abbreviations defined in the INI file. For instance: &quot;NB&quot; for New Business; &quot;EN&quot; for Endorsement; and so on.</td>
</tr>
<tr>
<td>company</td>
<td>A null-terminated string pointer that corresponds to the first WIP key value (which is also the first component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Company&quot; value.</td>
</tr>
<tr>
<td>lob</td>
<td>A null-terminated string pointer that corresponds to the second WIP key value (which is also the second component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Line of Business&quot; value.</td>
</tr>
<tr>
<td>policy</td>
<td>A null-terminated string pointer that corresponds to the WIP KeyID value. This parameter is sometimes referred to as the &quot;ID Number&quot; value.</td>
</tr>
<tr>
<td>description</td>
<td>A pointer to a null-terminated string that should be assigned as the WIP description value.</td>
</tr>
</tbody>
</table>

Remarks

This will attempt to create a new WIP entry during a non-modal session by calling RACCreateEntry. You must initialize the system before you call this function.

Returns

RAC_SUCCESS (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

Files

Include: racapi.h
Source: RACCREAT.C

Code Declarations

```c
int _RACAPI RACCreate( char far *transaction,
    char far *company,
    char far *lob,
    char far *policy,
    char far *description);
```

Visual Basic Declaration

```vbnet
Private Declare Function RACCreate Lib "racw32.dll" _
```
Example

The following code is an excerpt from an external program.
switch ( msg ) {
case WM_COMMAND: 
    switch(Param1) {
        case ID_CREATE_NEWPOL:
            rval = RACCreate("NB",
                "ACME INSURANCE",
                "HEALTH",
                "1402001",
                "New Sample Policy");
            if (rval != RAC_SUCCESS){
                ... // handle error
                break;
            }
            ... // continue success
            break;
        }
    }
}

See Also

RACCreateEntry, RACInit

Calls

RACCreateEntry
RACCreateEntry

Create a New WIP Entry

Syntax

```c
int RACCreateEntry(char* transaction,
                    char* company,
                    char* lob,
                    char* policy,
                    char* description,
                    unsigned int RunEntry)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction</td>
<td>A null-terminated string pointer that represents one of the valid [TRANSACTIONS] abbreviations defined in the INI file. For instance: &quot;NB&quot; for New Business; &quot;EN&quot; for Endorsement; and so on.</td>
</tr>
<tr>
<td>company</td>
<td>A null-terminated string pointer that corresponds to the first WIP key value (which is also the first component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Company&quot; value.</td>
</tr>
<tr>
<td>lob</td>
<td>A null-terminated string pointer that corresponds to the second WIP key value (which is also the second component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Line of Business&quot; value.</td>
</tr>
<tr>
<td>policy</td>
<td>A null-terminated string pointer that corresponds to the WIP KeyID value. This parameter is sometimes referred to as the &quot;ID Number&quot; value.</td>
</tr>
<tr>
<td>description</td>
<td>A null-terminated string pointer that should be assigned as the WIP description value.</td>
</tr>
<tr>
<td>RunEntry</td>
<td>A Boolean that specifies whether Entry should begin or not. If Entry is started, the desktop and form windows are created. Whether Entry is started or not, the current form set will be defined in the pdata structure.</td>
</tr>
</tbody>
</table>

Remarks

Create a New WIP Entry. This will attempt to create a new WIP entry during a non-modal session. You must initialize the system before you call this function. RACLib should have been initialized before calling this function.

For additional information see: RACInitAll, RACModal

Returns

RAC_SUCCESS (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

Files

Include: racapi.h
Source: RACCREAT.C
Code Declarations

    int _RACAPI RACCreateEntry(char far *transaction,
                                 char far *company,
                                 char far *lob,
                                 char far *policy,
                                 char far *description,
                                 BOOL RunEntry);

Visual Basic Declaration

    Private Declare Function RACCreateEntry Lib "racw32.dll" _
        (ByVal trans As String, _
         ByVal Key1 As String, _
         ByVal Key2 As String, _
         ByVal KeyID As String, _
         ByVal description As String, _
         ByVal runEntry As Long) As Long

Example

    The following code is an excerpt from an external program.
    switch ( msg ) {
      case WM_COMMAND:
        switch(Param1){
          case ID_CREATE_NEWPOL:
            rval = RACCreateEntry("NB",
                                 "ACME INSURANCE",
                                 "HEALTH",
                                 "1402001",
                                 "New Sample Policy",
                                 TRUE);
            if (rval != RAC_SUCCESS){
              ... handle error
              break;
            }
            ... continue success
            break;
        }
    }

See Also

    RACInitAll, RACModal

Calls

    AFELoadPPSFormset, RACCreateWipEntry, RACFindTransaction
RACCreateWipEntry

Create a New WIP Entry

Syntax

```c
int RACCreateWipEntry(char* company,
char* lob,
char* policy,
char* description,
PAFEDATA pdata,
unsigned int uniqcheck)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>company</td>
<td>A null-terminated string pointer that corresponds to the first WIP key value (which is also the first component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Company&quot; value.</td>
</tr>
<tr>
<td>lob</td>
<td>A null-terminated string pointer that corresponds to the second WIP key value (which is also the second component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Line of Business&quot; value.</td>
</tr>
<tr>
<td>policy</td>
<td>A null-terminated string pointer that corresponds to the WIP KeyID value. This parameter is sometimes referred to as the &quot;ID Number&quot; value.</td>
</tr>
<tr>
<td>description</td>
<td>A null-terminated string pointer that should be assigned as the WIP description value.</td>
</tr>
</tbody>
</table>

Remarks

Create a New WIP Entry.

Files

Include: racapi.h
Source: RACCREAT.C

Code Declarations

```c
int _RACAPI RACCreateWipEntry(char far *company,
char far *lob,
char far *policy,
char far *description,
PAFEDATA pdata,
BOOL uniqcheck);
```

Visual Basic Declaration

```vb
Private Declare Function RACCreateWipEntry Lib "racw32.dll" _
(ByVal Key1 As String, _
ByVal Key2 As String, _
ByVal KeyID As String, _
ByVal description As String, _
ByVal pdata As Long, _
ByVal uniqcheck As Long) As Long
```
Calls

RACCheckUserEntry
RACEditData

View form set specified.

Syntax

HWND RACEditData(HWND hwndParent,
                  char* filename)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwndParent</td>
<td>The parent window to contain this application.</td>
</tr>
<tr>
<td>filename</td>
<td>The name of a file to import.</td>
</tr>
</tbody>
</table>

Remarks

It is assumed that the window's handle passed represents another application's main window. We'll create a frame window to contain the form set view and start the entry system after usual initialization.

Returns

If successful, the handle of our desktop window is returned. On failure, zero will be returned and the caller should use RACGetStatus to retrieve the last know error code. Error code values are defined in RACAPI.H.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

HWND _RACAPI RACEditData(HWND hwndParent,
                          char *filename);

Visual Basic Declaration

Private Declare Function RACEditData Lib "racw32.dll" _
                              (ByVal hwndParent As Long,
                              ByVal filename As String) As Long

typedef

  typedef HWND (_RACAPIPTR RACEDITDATA) 
              (HWND hwndParent, char *filename);

See Also

RACCreateMainWindow, RACInitAll

Calls

RACEditDataStat
RACEnableMenu

Change to the Entry Menu.

Syntax
   int RACEnableMenu(void)

Remarks
This will cause the session to switch the menu within the main window. You must initialize the system before you call this function. This function replaces the existing menu with a new menu created from the file loaded by RACLoadMenu.

Calling RACRestoreMenu will restore the original menu.

Returns
This returns RAC_SUCCESS upon success or one of the values defined in RACAPI.H.

Files
Include: racapi.h
Source: RACMENU.C

Code Declarations
   int _RACAPI RACEnableMenu(void);

Visual Basic Declaration
   Private Declare Function RACEnableMenu Lib "racw32.dll" As Long
**RACEntry**

Start the Entry (PPS) System.

**Syntax**

```c
int RACEntry(HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>The external application's main (highest level) window.</td>
</tr>
</tbody>
</table>

**Remarks**

This is the simplest method of causing the Entry Program start. The only parameter required is the handle to a main window that contains (or can contain) a menu. This entry point starts the application in a modal fashion and will return to the caller's application when the user "Exits" from our menu.

**Returns**

- **RAC_SUCCESS** (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

**Files**

Include: racapi.h  
Source: RACENTRY.C

**Code Declarations**

```c
int _RACAPI RACEntry(HWND hwnd);
```

**Visual Basic Declaration**

```vbnet
Private Declare Function RACEntry Lib "racw32.dll" _
(ByVal hwnd As Long) As Long
```

**Example**

The following code is an excerpt from an external C program.

```c
switch ( msg ) {
    case WM_COMMAND:
        switch(Param1){
            case ID_INIT_ENTRY:
                rval = RACEntry( hwnd );
                if (rval != RAC_SUCCESS){
                    ... handle errors
                    break;
                }
                ... continue success
                break;
        }
}
```

In Visual Basic the following code could be used.

```vbnet
Sub Action_Click (Index As Integer)
    retval% = RACEntry(Form1.hWnd)
    REM Handle errors (retval% != 0) if needed.
End Sub
```
See Also

RACInitAll, RACTerminate, RACModal

Calls

RACGetStatus, RACInitAll, RACModal, RACProcessError
RACFindCompanyLOB

Locate a Specified Key1 and Key2 values

Syntax

```
VMMHANDLE RACFindCompanyLOB(VMMHANDLE masterH,
                           char* company,
                           char* lob)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>masterH</td>
<td>Form set handle that will be searched.</td>
</tr>
<tr>
<td>company</td>
<td>first group name (a.k.a. Key1)</td>
</tr>
<tr>
<td>lob</td>
<td>second group name (a.k.a. Key2)</td>
</tr>
</tbody>
</table>

Remarks

Determines if the specified company and line of business (Key1 and Key2) are a valid form set combination. The search is case insensitive.

This function is somewhat antiquated and limited in functionality. It has been left intact for legacy systems.

Returns

This returns a VMMHANDLE to the located group or VMMNULLHANDLE if not located.

Files

Include: racapi.h
Source: RACFIND.C

Code Declarations

```
VMMHANDLE _RACAPI RACFindCompanyLOB(VMMHANDLE masterH,
                           char far *company,
                           char far *lob);
```

Visual Basic Declaration

```
Private Declare Function RACFindCompanyLOB Lib "racw32.dll" _
(ByVal masterH As Long, _
ByVal company As String, _
ByVal lob As String) As Long
```
RACFindTransaction

Locate a Specified Transaction

Syntax

```
VMMHANDLE RACFindTransaction(VMMHANDLE translistH,
    char* transaction)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>translistH</td>
<td>VMM list of valid transaction structures.</td>
</tr>
<tr>
<td>transaction</td>
<td>Transaction being searched.</td>
</tr>
</tbody>
</table>

Remarks

This searches the specified VMM list for the specified transaction. This search is case insensitive.

Returns

A VMMHANDLE to the located transaction or VMMNULLHANDLE if not located is returned.

Files

Include: racapi.h
Source: RACFIND.C

Code Declarations

```
VMMHANDLE _RACAPI RACFindTransaction(VMMHANDLE translistH,
    char far *transaction);
```

Visual Basic Declaration

```
Private Declare Function RACFindTransaction Lib "racw32.dll" _
    (ByVal translistH As Long, _
    ByVal transaction As String) As Long
```
RACGetAccelHandle

Get the Accelerator Table Handle

Syntax

HWND RACGetAccelHandle(void)

Remarks

This returns the accelerator tables handle if it was loaded by a function (directly or indirectly) within RACLib. This is provided for applications that manage non-modal sessions that wish to provide accelerator translations.

Returns

A HANDLE or NULL is returned.

Files

Include: racapi.h
Source: RACDATA.C

Code Declarations

HWND _RACAPI RACGetAccelHandle(void);

Visual Basic Declaration

Private Declare Function RACGetAccelHandle Lib "racw32.dll" As Long
typedef
typedef HWND ( _RACAPIPTR RACGETACCELHANDLE) (void);
RACGetAFEData

Get the Entry Data Structure

**Syntax**

```
PAFEDATA RACGetAFEData(void)
```

**Remarks**

This returns the AFEDATA structure created by RACInit.

This structure should not be altered by the calling application. This structure contains most of the system wide information used by the Entry system. This structure is defined by AFELIB.H.

**Returns**

A pointer to the current AFEDATA structure will be returned.

**Files**

Include: racapi.h  
Source: RACDATA.C

**Code Declarations**

```
PAFEDATA _RACAPI RACGetAFEData(void);
```

**Visual Basic Declaration**

```
Private Declare Function RACGetAFEData Lib "racw32.dll" As Long
```
RACGetClientWindow

Return client window.

**Syntax**

HWND RACGetClientWindow(void)

**Remarks**

Get the top client window's handle created by [RACThread](#).

**Returns**

The window's handle or NULLHANDLE is returned.

**Files**

Include: racapi.h
Source: RACTHRD.C

**Code Declarations**

```c
HWND _RACAPI RACGetClientWindow(void);

typedef HWND (_RACAPIPTR RACGETCLIENTWINDOW) (void);
```

**Visual Basic Declaration**

```vbnet
Private Declare Function RACGetClientWindow Lib "racw32.dll" As Long

typedef
typedef HWND (_RACAPIPTR RACGETCLIENTWINDOW) (void);
```
RACGetDescription

Return description of error message

Syntax

char* RACGetDescription(int RacErrNo)

Parameter | Description
--- | ---
RacErrNo | Error code.

Remarks

Get the description or message of an error code.

Returns

This returns a character pointer to the message.

Files

Include: racapi.h
Source: RACDATA.C

Code Declarations

char* _RACAPI RACGetDescription(int RacErrNo);

Visual Basic Declaration

Private Declare Function RACGetDescription Lib "racw32.dll" _
(ByVal errorCode As Long) As String

typedef

typedef char* (_RACAPIPTR RACGETDESCRIPTION) (int ErrNo);
RACGetFrameWindow

Return frame window.

Syntax

HWND RACGetFrameWindow(void)

Remarks

Return the top level Window Handle created by RACThread.

Files

Include: racapi.h
Source: RACTHRD.C

Code Declarations

HWND _RACAPI RACGetFrameWindow(void);

Visual Basic Declaration

Private Declare Function RACGetFrameWindow Lib "racw32.dll" As Long
RACGetMenuHandle

Get the Entry Menu Handle.

**Syntax**

```c
HWND RACGetMenuHandle(void)
```

**Remarks**

This returns the menu's handle of a menu created by RACLoadMenu. You must initialize the system before you call this function.

**Returns**

A handle to a menu will be returned or NULL.

**Files**

Include: racapi.h
Source: RACMENU.C

**Code Declarations**

```c
HWND _RACAPI RACGetMenuHandle(void);
```

**Visual Basic Declaration**

```vb
Private Declare Function RACGetMenuHandle Lib "racw32.dll" As Long
```
RACGetOrigMenuHandle

Get the Original Menu Handle.

Syntax

HWND RACGetOrigMenuHandle(void)

Remarks

This returns the original main window's menu handle if it was replaced by a call to RACEnableMenu. You must initialize the system before you call this function.

Returns

A handle to a menu will be returned or NULL. A NULL value means the main application window does not contain a menu or that the original menu has not been replaced by RACLib.

Files

Include: racapi.h
Source: RACMENU.C

Code Declarations

HWND _RACAPI RACGetOrigMenuHandle(void);

Visual Basic Declaration

Private Declare Function RACGetOrigMenuHandle Lib "racw32.dll" As Long
**RACGetSession**

Get the session for this window

**Syntax**

```c
struct racdata* RACGetSession(HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window</td>
</tr>
</tbody>
</table>

**Remarks**

Retrieve the session that matches on the main frame window's handle.

**Files**

Include: racapi.h
Source: RACDATA.C

**Code Declarations**

```c
c struct racdata* _RACAPI RACGetSession(HWND hwnd);
```

**Visual Basic Declaration**

```vb
Private Declare Function RACGetSession Lib "racw32.dll" _
    (ByVal hwnd As Long) As Long
```

typedef

```c
typedef void* (_RACAPIPTR RACGETSESSION) (HWND hwnd);
```
RACGetStatus

Get Last Error Status Code

Syntax
    int RACGetStatus(void)

Remarks
This returns the most recent error code assigned by RACLib functionality.

Returns
One of the values defined in RACAPI.H is normally returned.

Files
Include: racapi.h
Source: RACDATA.C

Code Declarations
    int _RACAPI RACGetStatus(void);

Visual Basic Declaration
    Private Declare Function RACGetStatus Lib "racw32.dll" As Long
typedef
    typedef int (_RACAPIPTR RACGETSTATUS) (void);
**RACHookProc**

Windows accelerator hook procedure

**Syntax**

```c
LRESULT RACHookProc(int code,
                     WPARAM wParam,
                     LPARAM lParam)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>a Window's hook code value</td>
</tr>
<tr>
<td>wParam</td>
<td>the first message parameter</td>
</tr>
<tr>
<td>lParam</td>
<td>the second message parameter</td>
</tr>
</tbody>
</table>

**Remarks**

This function is only useful for Windows applications. **RACHookProc** determines if the next queued message is an accelerator message for a RACLib application window or dialog.

The message hook is established by a call to **RACSetHook** and removed via **RACUnhook**.

**Returns**

Hooks are called in sequence; therefore, this hook will call the one installed before this hook. The return value will be the result of the last hook that executes. See Windows programming help for the meaning of the return values.

**Files**

Include: racapi.h
Source: RACHOOK.C

**Code Declarations**

```
LRESULT CALLBACK RACHookProc(int code,
                             WPARAM wParam,
                             LPARAM lParam);
```

**Visual Basic Declaration**

```vba
Private Declare Function RACHookProc Lib "racw32.dll" _
    (ByVal codeval As Long, _
     ByVal wParam As Long, _
     ByVal lParam As Long) As Long
```
RACInit

Initialize Entry System

Syntax

```c
int RACInit(HWND hwnd,
            char* userid,
            char* sysid)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>The external application's main (highest level) window.</td>
</tr>
<tr>
<td>userid</td>
<td>A null-terminated string pointer that corresponds to a valid user ID with access to the entry system.</td>
</tr>
<tr>
<td>sysid</td>
<td>Future expansion - not currently used.</td>
</tr>
</tbody>
</table>

Remarks

This causes the entry system to initialize. Initialization includes memory management, FAP, WIP, user login, and the help system.

Calling application will need to call RACTerminate to shutdown correctly at program end or during failure conditions.

When establishing a non-modal session, only RACWorkingPath can be called before this function.

During initialization, the login screen can appear if the user ID is not specified by the parameter and is not specified in the INI file.

Returns

RAC_SUCCESS (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

Files

Include: racapi.h
Source: RACINIT.C

Code Declarations

```c
int _RACAPI RACInit(HWND mainWnd,
            char far *userid,
            char far *sysid);
```

Visual Basic Declaration

```vb
Private Declare Function RACInit Lib "racw32.dll" _
    (ByVal mainWnd As Long, _
    ByVal userid As String, _
    ByVal sysid As String) As Long
```

Example

The following code is an excerpt from an external program.
switch ( msg ) {
    case WM_COMMAND:
        switch(Param1){
            case ID_INIT_ENTRY:
                rval = RACInit(hwnd, "USER1", ",");
                if (rval != RAC_SUCCESS){
                    RACTerminate(FALSE);
                    break;
                }
                ... // continue success
                break;
            }
            break;
        case WM_CLOSE:
            ... // if RAC running
            RACTerminate(FALSE);
            break;
        }
    break;
}

See Also

RACTerminate

Calls

RACmdLineArg2Afe, RACCreateNewSession, RACGetCmdLineArg, RACGetSession, RACSetCurSession,
RACSetIniFile, RACSetStatus
RACInitAll

Initialize Entire Entry System.

Syntax

```
int RACInitAll(HWND hwnd,
               char* userid,
               char* sysid)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>The external application's main (highest level) window.</td>
</tr>
<tr>
<td>userid</td>
<td>A null-terminated string pointer that corresponds to a valid user ID with access to the entry system.</td>
</tr>
<tr>
<td>sysid</td>
<td>Future expansion - not currently used.</td>
</tr>
</tbody>
</table>

Remarks

This causes the entry system to initialize. Initialization includes memory management, FAP, WIP, user login, and the help system.

This function will call RACWorkingPath before starting the initialization process. If an error occurs, RACTerminate will be called and the path restored.

If the process initializes correctly, the defined "menu" will be loaded (although not activated) and the default accelerators will be loaded.

If the calling application does not call RACModal, then it should eventually call RACTerminate to shutdown correctly.

During initialization, the login screen can appear if the user ID is not specified by parameter and is not specified in the INI file.

Returns

RAC_SUCCESS (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

Files

Include: racapi.h
Source: RACINIT.C

Code Declarations

```
int _RACAPI RACInitAll(HWND mainWnd,
               char far *userid,
               char far *sysid);
```

Visual Basic Declaration

```
Private Declare Function RACInitAll Lib "racw32.dll" _
(ByUrl mainWnd As Long, _
    ByVal userid As String, _
    ByVal sysid As String) As Long
```
Example

The following code is an excerpt from an external program.

```c
switch (msg) {
    case WM_COMMAND:
        switch(Param1){
            case ID_INIT_ENTRY:
                rval = RACInitAll(hwnd,
                                   "USER1",
                                   "");
                if (rval != RAC_SUCCESS){
                    ... handle error
                    break;
                }
                ... continue success
                break;
        }
    case WM_CLOSE:
        ... if RAC running
        RACTerminate(FALSE);
        break;
}
```

See Also

RACInit, RACWorkingPath, RACLoadMenu, RACLoadAccelerators, RACTerminate

Calls

RACGetStatus, RACInit, RACLoadAccelerators, RACLoadMenu, RACProcessError, RACRestorePath, RACSetStatus, RACWorkingPath
RACInitCtrl

Initialize Entry System (menu and accelerator control).

**Syntax**

```c
int RACInitCtrl(HWND hwnd,
           char* userid,
           char* sysid,
           unsigned int loadMenu,
           unsigned int loadAccel)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>The external application's main (highest level) window.</td>
</tr>
<tr>
<td>userid</td>
<td>A null-terminated string pointer that corresponds to a valid user ID with access to the entry system.</td>
</tr>
<tr>
<td>sysid</td>
<td>Future expansion - not currently used.</td>
</tr>
<tr>
<td>loadMenu</td>
<td>If true, the menu is loaded.</td>
</tr>
<tr>
<td>loadAccel</td>
<td>If true, the accelerator table is loaded.</td>
</tr>
</tbody>
</table>

**Remarks**

This causes the entry system to initialize. Initialization includes memory management, FAP, WIP, user login, and the help system.

This function will call RACWorkingPath before starting the initialization process. If an error occurs, RACTerminate will be called and the path restored.

If the process initializes correctly and the load menu flag is true; the defined "menu" will be loaded (although not activated). In addition, the default accelerators will be loaded if the load accelerator flag is true.

If the calling application does not call RACModal, then it should eventually call RACTerminate to shutdown correctly.

During initialization, the login screen can appear if the user ID is not specified by parameter and is not specified in the INI file.

**Returns**

RAC_SUCCESS (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

**Files**

Include: racapi.h
Source: RACINIT.C

**Code Declarations**

```c
int _RACAPI RACInitCtrl(HWND hwnd,
           char far *userid,
```
char far *sysid,
BOOL loadMenu,
BOOL loadAccel);

Visual Basic Declaration
Private Declare Function RACInitCtrl Lib "racw32.dll" _
(ByVal hwnd As Long, _
ByVal userid As String, _
ByVal sysid As String, _
ByVal loadMenu As Long, _
ByVal loadAccel As Long) As Long

typedef
typedef int (_RACAPIPTR RACINITCTRL)
(HWND hwnd,
char far *userid,
char far *sysid,
BOOL accel,
BOOL menu);

Calls
RACGetStatus, RACInit, RACLoadAccelerators, RACLoadMenu, RACProcessError, RACRestorePath,
RACSetStatus, RACWorkingPath
RACLibVersion

Get library version information.

**Syntax**

```c
FSI_VERSION* RACLibVersion(void)
```

**Remarks**

This returns a pointer to a structure containing version information. The structure is composed of four elements: a numeric version value, a character string, a date, and a time.

**Returns**

This returns a pointer to a structure containing version information.

**Files**

Include: racapi.h  
Source: RACVERSN.C

**Code Declarations**

```c
FSI_VERSION* _VMMAPI RACLibVersion(void);
```

**Visual Basic Declaration**

```vb
Private Declare Function RACLibVersion Lib "racw32.dll" As Long
```
RACLoadIni

Load the INI file.

Syntax

    int RACLoadIni(void)

Remarks

Load the INI file. The file name is defaulted to FSIUSER.INI if RACSetIniFile did not supply it.

Files

Include: racapi.h
Source: RACTHRD.C

Calls

RACGetIniFile, RACSetIniFile, RACSetStatus
RACLoadMenu

This function will query the INI file for the MEN.RES compatible file to load.

Syntax
   int RACLoadMenu(void)

Remarks

This function will query the INI file for the MEN.RES compatible file to load. The following INI options will be queried in the order shown. Once a valid option has been determined the menu is created but not activated. A call to RACEnableMenu will be required to activate the menu.

Order of search to determine menu to load:

[AltMenu]
File
If not located then use:

[MENU]
File
If not located the file MEN.RES will be attempted.
If no menu is located, none will be created and an error returned.
RACRestoreMenu will restore the original menu. RACTerminate automatically calls this function.

Returns

This returns RAC_SUCCESS upon success or one of the values defined in RACAPI.H.

Files

Include: racapi.h
Source: RACMENU.C

Code Declarations

   int _RACAPI RACLoadMenu(void);

Visual Basic Declaration
   Private Declare Function RACLoadMenu Lib "racw32.dll" As Long

typedef
   typedef int (_RACAPIPTR RACLOADMENU) (void);

Calls

RACGetCmdLineArg, RACSetStatus
RACMain

Remote Start of Entry (MODAL)

Syntax

    int RACMain(HWND hwnd,
        char* transaction,
        char* company,
        char* lob,
        char* policy,
        char* description,
        char* userid,
        char* sysid,
        int entrymode)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>The external application's main (highest level) window.</td>
</tr>
<tr>
<td>transaction</td>
<td>A null-terminated string pointer that represents one of the valid [TRANSACTIONS] abbreviations defined in the INI file. For instance: &quot;NB&quot; for New Business; &quot;EN&quot; for Endorsement; and so on.</td>
</tr>
<tr>
<td>company</td>
<td>A null-terminated string pointer that corresponds to the first WIP key value (which is also the first component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Company&quot; value.</td>
</tr>
<tr>
<td>lob</td>
<td>A null-terminated string pointer that corresponds to the second WIP key value (which is also the second component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Line of Business&quot; value.</td>
</tr>
<tr>
<td>policy</td>
<td>A null-terminated string pointer that corresponds to the WIP KeyID value. This parameter is sometimes referred to as the &quot;ID Number&quot; value.</td>
</tr>
<tr>
<td>description</td>
<td>A null-terminated string pointer that should be assigned as the WIP description value.</td>
</tr>
<tr>
<td>userid</td>
<td>A null-terminated string pointer that corresponds to a valid user ID with access to the entry system.</td>
</tr>
<tr>
<td>sysid</td>
<td>Future expansion - not currently used.</td>
</tr>
</tbody>
</table>
| entrymode    | A valid RALib action. Currently defined:
    AFEACTION_CREATE = Create new WIP.
    AFEACTION_UPDATE = Update existing WIP. |

Remarks

Starts a modal session that is similar to starting AFEMAIN.EXE. No other initialization or termination functions need be called by the user's application. The function does not return until the session completes.

This modal session will take over the main window handling and replace the existing menu. The new menu will be created from the file specified by the [ALTMENU] setting in the INI file. If omitted, the menu will default to the file specified by the [MENU] setting in the INI file. The original menu will be restored before returning.
During initialization, the login screen can appear if the user ID is not specified by the parameter and is not specified in the INI file.

**Returns**

**RAC_SUCCESS** (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

**Files**

Include: racapi.h  
Source: RACMAIN.C

**Code Declarations**

```c
int _RACAPI RACMain(HWND hwnd,  
                     char far *transaction,  
                     char far *company,  
                     char far *lob,  
                     char far *policy,  
                     char far *description,  
                     char far *userid,  
                     char far *sysid,  
                     int mode);
```

**Visual Basic Declaration**

```vb
Private Declare Function RACMain Lib "racw32.dll" _  
    (ByVal hwnd As Long, _  
     ByVal Key1 As String, _  
     ByVal Key2 As String, _  
     ByVal KeyID As String, _  
     ByVal description As String, _  
     ByVal userid As String, _  
     ByVal sysid As String, _  
     ByVal modeval As Long) As Long
```

**typedef**

```c
typedef void (_RACAPIPTR RACMAIN)(HWND hwnd,  
                                  char *transaction,  
                                  char *company,  
                                  char *lob,  
                                  char *policy,  
                                  char *description,  
                                  char *userid,  
                                  char *sysid,  
                                  long mode);
```

**Example**

The following code is an excerpt from an external program.

```c
switch ( msg ) {
  case WM_COMMAND:  
      switch(Param1) {  
        case ID_CREATE_NEWPOL:  
            rval = RACMain( hwnd,  
                            "NB",  
                            "ACME INSURANCE",  
                            "HEALTH",  
                            "1402001",  
                            "New Sample Policy",  
                            "USER1",  
                            "",  
```
AFEACTION_CREATE);
if (rval != RAC_SUCCESS){
    ... // handle error
    break;
}
... // continue success
break;
}

See Also

RACInitAll, RACCreate, RACUpdate, RACTerminate, RACModal

Calls

RACCreate, RACEnableMenu, RACGetStatus, RACInitAll, RACModal, RACProcessError, RACRestoreMenu, RACUpdate
RACMainWndProc

The Documaker Workstation message handler.

Syntax

\[
\text{MRESULT} \ RACMainWndProc(\text{HWND} \ hwnd, \\
\text{UINT} \ msg, \\
\text{WPARAM} \ mp1, \\
\text{LPARAM} \ mp2)
\]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
<tr>
<td>msg</td>
<td>Message</td>
</tr>
<tr>
<td>mp1</td>
<td>Message parameter one.</td>
</tr>
<tr>
<td>mp2</td>
<td>Message parameter two.</td>
</tr>
</tbody>
</table>

Remarks

Is a main window message handler used by the entry system. During a non-modal session, it is possible to pass messages to this function that should be handled by the entry system. Most often, this function will be passed WM_COMMAND messages associated with menu commands. You must initialize the system before you call this function.

The prototype for this function should conform to that required for a standard window within the operating environment.

Returns

The return value will be that returned from a WndProc in the given operating environment.

Files

Include: racapi.h
Source: RACPROC.C

Code Declarations

\[
\text{MRESULT} \ \text{EXPENTRY} \ RACMainWndProc(\text{HWND} \ hwnd, \\
\text{MMSG} \ msg, \\
\text{MPARAM1} \ mp1, \\
\text{MPARAM2} \ mp2);
\]

Visual Basic Declaration

Private Declare Function RACMainWndProc Lib "racw32.dll" _
(ByVal hwnd As Long, _
ByVal msg As Long, _
ByVal mp1 As Long, _
ByVal mp2 As Long) As Long

Example

The following code is an excerpt from an external program.
switch ( msg ) {
    case WM_COMMAND:
        rval = RACMainWndProc( hwnd, msg, Param1, Param2);
        ...

See Also

RACInit

Calls

RACGetCommandLineArg, RACGetSession, RACServeAsProxy, RACSetDesktop, RACSetFocus, RACTermSession
RACModal

Start Modal Entry.

Syntax

```c
int RACModal(void)
```

Remarks

This causes the session to take over the main window handling. You must initialize the system before you call this function.

This function replaces the existing menu with a new menu if one was loaded. The original menu will be restored before returning.

The function does not return until the Entry system completes. The entry system is terminated but the session is not terminated.

Returns

`RAC_SUCCESS` (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

Files

Include: racapi.h
Source: RACMODAL.C

Code Declarations

```c
int _RACAPI RACModal(void);
```

Visual Basic Declaration

```vbnet
Private Declare Function RACModal Lib "racw32.dll" As Long
```

Example

The following code is an excerpt from an external program.

```c
if ( RACInitAll(hwnd, NULL, NULL) != RAC_SUCCESS ) {
    RACProcessError(RACGetStatus());
    return(rval);
}
RACModal();
```

See Also

RACInit

Calls

RACModalEx
RACPackDatabase

Pack the database.

Syntax

```c
int RACPackDatabase(HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

Remarks

Packs the database associated with the window session.

Files

Include: racapi.h
Source: RACDATA.C

Code Declarations

```c
int _RACAPI RACPackDatabase(HWND hwnd);
```

Visual Basic Declaration

```vbnet
Private Declare Function RACPackDatabase Lib "racw32.dll" _
               (ByVal hwnd As Long) As Long
```

Calls

[RACGetSession](#)
RACRemoveScrollbar

Remove the scroll bar from window.

Syntax

    void RACRemoveScrollbar(HWND hwnd)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

Remarks

Remove the scroll bar from window.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

    void RACRemoveScrollbar(HWND hwnd);

    Visual Basic Declaration
    Private Declare Sub RACRemoveScrollbar Lib "racw32.dll" _
        (ByVal hwnd As Long)

typedef

    typedef void (_RACAPIPTR RACREMOVESCROLLBAR) (HWND hwnd);
RACRestoreMenu

Restore the Original Menu.

Syntax

\[
\text{int RACRestoreMenu(void)}
\]

Remarks

This will cause the session to restore the original main window's menu if it was replaced by a call to RACEnableMenu. You must initialize the system before you call this function.

RACTerminate automatically calls this function.

Returns

This returns RAC_SUCCESS upon success or one of the values defined in RACAPI.H.

Files

Include: racapi.h
Source: RACMENU.C

Code Declarations

\[
\begin{align*}
\text{int } &\_\text{RACAPI RACRestoreMenu(void);} \\
\text{Visual Basic Declaration} &\ \\
&\text{Private Declare Function RACRestoreMenu Lib } \"\text{racw32.dll}\" \text{ As Long}
\end{align*}
\]

typedef

\[
\text{typedef int } (\_\text{RACAPIPTR RACRESTOREMENU})(\text{void});
\]
RACRestorePath

Restore original working directory.

Syntax

void RACRestorePath(void)

Remarks

This function will restore the working directory to the original one saved by calling RACWorkingPath.

Files

Include: racapi.h
Source: RACINIT.C

Code Declarations

void _RACAPI RACRestorePath(void);

Visual Basic Declaration

Private Declare Sub RACRestorePath Lib "racw32.dll"

See Also

RACWorkingPath
RACRetrieve

Display transaction from archive.

Syntax

```c
int RACRetrieve(HWND hwnd,
    char* Key1,
    char* Key2,
    char* KeyID,
    int startpage)
```

Parameter | Description
---|---
hwnd | The parent window to contain this application.
Key1 | Search data for Key1.
Key2 | Search data for Key2.
KeyID | Search data for KeyID.
startpage | Page to begin display of form set.

Remarks

Display transaction from archive, similar to [RACRetrieveArchive](#) except that it does not call [RACInit](#), [RACInit](#) must be called before this function.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

```c
int _RACAPI RACRetrieve(HWND hwnd,
    char *Key1,
    char *Key2,
    char *KeyID,
    int startpage);
```

Visual Basic Declaration

```vbnet
Private Declare Function RACRetrieve Lib "racw32.dll" _
    (ByVal hwnd As Long, _
    ByVal Key1 As String, _
    ByVal Key2 As String, _
    ByVal KeyID As String, _
    ByVal startpage As Long) As Long
```

Calls

[RACFilterPageForDisplay](#), [RACProcessError](#), [RACSetCurSession](#), [RACSetStatus](#)
RACRetrieveArchive

View form set specified.

Syntax

HWND RACRetrieveArchive(HWND hwndParent,
                        char* Key1,
                        char* Key2,
                        char* KeyID,
                        int startpage,
                        int location)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwndParent</td>
<td>The parent window to contain this application.</td>
</tr>
<tr>
<td>Key1</td>
<td>Search data for Key1.</td>
</tr>
<tr>
<td>Key2</td>
<td>Search data for Key2.</td>
</tr>
<tr>
<td>KeyID</td>
<td>Search data for KeyID.</td>
</tr>
<tr>
<td>startpage</td>
<td>Set beginning page</td>
</tr>
<tr>
<td>location</td>
<td>Set screen location.</td>
</tr>
<tr>
<td></td>
<td>0 = Left half display (default)</td>
</tr>
<tr>
<td></td>
<td>1 = Right half display</td>
</tr>
<tr>
<td></td>
<td>2 = Top half display</td>
</tr>
<tr>
<td></td>
<td>3 = Bottom half display</td>
</tr>
<tr>
<td></td>
<td>4 = normal window</td>
</tr>
<tr>
<td></td>
<td>5 = maximized window</td>
</tr>
</tbody>
</table>

Remarks

It is assumed that the window's handle passed represents another application's main window. We'll create a frame window to contain the form set view and start the entry system after usual initialization.

See the <ArcRet> INI group to determine which fields are matched with Key1, Key2, and KeyID.

Returns

If successful, the handle of our desktop window is returned. On failure, zero will be returned and the caller should use RACGetStatus to retrieve the last know error code. Error code values are defined in RACAPI.H.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

HWND _RACAPI RACRetrieveArchive(HWND hwndParent,
                                 char *Key1,
                                 char *Key2,
                                 char *KeyID,
int startpage,
int location);

Visual Basic Declaration
Private Declare Function RACRetrieveArchive Lib "racw32.dll" _
(ByVal hwndParent As Long, _
ByVal Key1 As String, _
ByVal Key2 As String, _
ByVal KeyID As String, _
ByVal startpage As Long, _
ByVal location As Long) As Long

Calls
RACRetrieveArchiveStat
RACRetrieveArchiveHab

View form set specified.

Syntax

HWND RACRetrieveArchiveHab(HINSTANCE hab,
char* Key1,
char* Key2,
char* KeyID,
int startpage,
int location)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hab</td>
<td>Anchor block. In Windows, it is the instance handle.</td>
</tr>
<tr>
<td>Key1</td>
<td>Search data for Key1.</td>
</tr>
<tr>
<td>Key2</td>
<td>Search data for Key2.</td>
</tr>
<tr>
<td>KeyID</td>
<td>Search data for KeyID.</td>
</tr>
<tr>
<td>startpage</td>
<td>set beginning page</td>
</tr>
<tr>
<td>location</td>
<td>Set screen location.</td>
</tr>
<tr>
<td></td>
<td>0 = Left half display (default)</td>
</tr>
<tr>
<td></td>
<td>1 = Right half display</td>
</tr>
<tr>
<td></td>
<td>2 = Top half display</td>
</tr>
<tr>
<td></td>
<td>3 = Bottom half display</td>
</tr>
<tr>
<td></td>
<td>4 = normal window</td>
</tr>
<tr>
<td></td>
<td>5 = maximized window</td>
</tr>
</tbody>
</table>

Remarks

Is identical to RACViewByKey except the application instance is passed instead of a window's handle. Actually, RACRetrieveArchiveHab uses the window's handle only to obtain the application instance.

Returns

If successful, the handle of our desktop window is returned. On failure, zero will be returned and the caller should use RACGetStatus to retrieve the last know error code. Error code values are defined in RACAPI.H.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

HWND _RACAPI RACRetrieveArchiveHab(HAB hab,
char *Key1,
char *Key2,
char *KeyID,
int startpage,
int location);
Visual Basic Declaration

    Private Declare Function RACRetrieveArchiveHab Lib "racw32.dll" _
        (ByVal hInstance As Long, _
        ByVal Key1 As String, _
        ByVal Key2 As String, _
        ByVal KeyID As String, _
        ByVal startpage As Long, _
        ByVal location As Long) As Long

Calls

RACRetrieveArchiveStat
RACSave

Save transaction.

Syntax

int RACSave(void)

Remarks

Save the current transaction returns SUCCESS if the transaction was saved or if no transaction was current and needed to be saved.

Files

Include: racapi.h
Source: RACTERM.C

Code Declarations

    int _RACAPI RACSave(void);

Visual Basic Declaration

    Private Declare Function RACSave Lib "racw32.dll" () As Long

typedef

    typedef int (_RACAPIPTR RACSAVE) (void);

Calls

RACGetSession
RACSaveAsProxy

Save the current form set in the current AFEData structure into a proxy file.

Syntax

```c
int RACSaveAsProxy(HWND hwnd,
                   VMMHANDLE menuH)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hab</td>
<td>Anchor block or instance handle for Windows.</td>
</tr>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
<tr>
<td>menuH</td>
<td>Menu handle.</td>
</tr>
</tbody>
</table>

Remarks

Save the current form set in the current AFEData structure into a proxy file.

Files

Include: racapi.h
Source: RACDATA.C

Code Declarations

```c
int _VMMAPI RACSaveAsProxy(HAB hab, HWND hwnd, VMMHANDLE menuH);
```

Visual Basic Declaration

```vb
Private Declare Function RACSaveAsProxy Lib "racw32.dll" _
  (ByVal hInstance As Long, _
   ByVal hWnd As Long, _
   ByVal menuH As Long) As Long
```

Calls

RACGetSession
**RACSetCaptionOff**

Set caption off for this window.

**Syntax**

```c
void RACSetCaptionOff(HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

**Remarks**

Set caption off for this window.

**Files**

Include: racapi.h  
Source: RACPROC.C

**Code Declarations**

```c
void _RACAPI RACSetCaptionOff(HWND hwnd);
```

**Visual Basic Declaration**

```vbnet
Private Declare Sub RACSetCaptionOff Lib "racw32.dll" _
(ByVal hwnd As Long)
```

**typedef**

```c
typedef void (_RACAPIPTR RACSETCAPTIONOFF) (HWND hwnd);
```

**Calls**

[RACGetSession](#)
RACSetCaptionOn

Set caption on for this window.

Syntax
void RACSetCaptionOn(HWND hwnd)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window.</td>
</tr>
</tbody>
</table>

Remarks
Set caption on for this window.

Files
Include: racapi.h
Source: RACPROC.C

Code Declarations
```
void _RACAPI RACSetCaptionOn(HWND hwnd);
```

Visual Basic Declaration
```
Private Declare Sub RACSetCaptionOn Lib "racw32.dll" _
   (ByVal hwnd As Long)
```

typedef
typedef void (RACAPIPTR RACSETCAPTIONON) (HWND hwnd);

Calls
RACGetSession
RACSetCmdLineArg

This allows historical command line options for AFEMAIN program to be passed to RACInit.

Syntax

```c
int RACSetCmdLineArg(int argc, char* argv[])
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>argc</td>
<td>Number of parameters.</td>
</tr>
<tr>
<td>argv</td>
<td>Array of pointers that contain the parameters.</td>
</tr>
</tbody>
</table>

Remarks

This allows historical command line options for AFEMAIN program to be passed to RACInit. Parameters are identical to those of a c program's main function.

The pointer to the structure is initially NULL but it's filled by the RACSetCmdLineArg function. The memory for the structure is static but there is a linked list's handle that is created via VMMCreateList. This handle will not survive RACTerminate. Therefore, RACSetCmdLineArg will need to be called before each RACInit.

Returns

SUCCESS or FAIL

Files

Include: racapi.h
Source: RACCMDL.C

Code Declarations

```c
int _RACAPI RACSetCmdLineArg(int argc, char *argv[]);
```

Visual Basic Declaration

```vbnet
Private Declare Function RACSetCmdLineArg Lib "racw32.dll" _
    (ByVal argc As Long, ByVal argv As Long) As Long
```

typedef

```c
typedef int (_RACAPIPTR RACSetCmdLineArg)(int argc, char *argv[]);
```

Example

Valid command line options

-INI or /INI - set the INI FILE.
-MODE or /MODE - set to one of the following to etrieve WIP
-USERID or /USERID - set the user ID
-HD or /HD - set help debug flag
-MENU or /MENU - set the menu file.
Calls

RACSetIniFile
RACSetCurSession

Set this window to be the current session.

Syntax

```c
int RACSetCurSession(HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window</td>
</tr>
</tbody>
</table>

Remarks

Set the main frame window's handle (hwndMainFrame variable) if we have a valid session for the handle passed.

Files

Include: racapi.h
Source: RACDATA.C

Code Declarations

```c
int _RACAPI RACSetCurSession(HWND hwnd);
```

Visual Basic Declaration

```vbnet
Private Declare Function RACSetCurSession Lib "racw32.dll" -
    (ByVal hWnd As Long) As Long
```

typedef

```c
typedef int (_RACAPIPTR RACSETCURSESSION)(HWND hwnd);
```

Calls

RACGetSession, RACSetStatus
RACSetHook

Establish Windows message hook.

Syntax

    void RACSetHook(void)

Remarks

This installs RACHookProc as a Windows compatible message hook. See that function for more information.

Files

Include: racapi.h
Source: RACHOOK.C

Code Declarations

    void _RACAPI RACSetHook(void);

Visual Basic Declaration

    Private Declare Sub RACSetHook Lib "racw32.dll" ()
**RACSetIniFile**

Sets the INI file path name used by RACLib functions.

**Syntax**

```c
int RACSetIniFile(char* iniFile)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iniFile</td>
<td>File path name of INI file.</td>
</tr>
</tbody>
</table>

**Remarks**

Sets the INI file path name used by RACLib functions. A string is set within RACLib that is used by the RACInit function.

**Returns**

RAC_SUCCESS (0) is always returned.

**Files**

Include: racapi.h  
Source: RACINIT.C

**Code Declarations**

```c
int _RACAPI RACSetIniFile(char *iniFile);

Visual Basic Declaration
Private Declare Function RACSetIniFile Lib "racw32.dll" _
    (ByVal iniFile As String) As Long
```

typedef
```
typedef int (_RACAPIPTR RACSETINIFILE) (char *iniFile);
```

**See Also**

RACGetIniFile
**RACSetParent**

Use this function to set the parent window. This function lets you size the frame window which contains the form sets you display with these functions:

- RACViewData
- RACViewDataHab
- RACViewByKey
- RACViewByKeyHab
- RACRetrieveArchive
- RACRetrieveArchiveHab

If you omit the RACSetParent function, the size of the frame window is 1/3 of the screen size. In some cases, the text on the form set may be too small to read at this size.

Keep in mind that the form must be open before you can set it as a parent.

**Syntax**

```c
void RACSetParent(HWND hwnd)
```

**Remarks**

Set the parent window for **RACThread** if the parent window has been set and is a valid window's handle then we create a child window instead of using the desktop.

**Files**

Include: racapi.h
Source: RACTHRD.C

**Code Declarations**

```c
void _RACAPI RACSetParent(HWND hwnd);

typedef void (_RACAPIPTR RACSETPARENT) (HWND hwnd);
```

**Visual Basic Declaration**

```vb
Private Declare Sub RACSetParent Lib "racw32.dll" _
                                (ByVal hwnd As Long)
```

```c
typedef void (_RACAPIPTR RACSETPARENT) (HWND hwnd);
```
RACSetProxyHandle

Set the document handle of a proxy file for RACViewProxy.

Syntax

    void RACSetProxyHandle(VMMHANDLE docH)

Remarks

Set the document handle of a proxy file for RACViewProxy.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

    void _RACAPI RACSetProxyHandle(VMMHANDLE docH);

    Visual Basic Declaration
    Private Declare Sub RACSetProxyHandle Lib "racw32.dll" _
        (ByVal docH As Long)

typedef

    typedef void (_RACAPIPTR RACSETPROXYHANDLE)(VMMHANDLE docH);
RACSetSessionMenu

Set the menu's handle in the session structure

Syntax

```c
unsigned short RACSetSessionMenu(VMMHANDLE hwnd,
        VMMHANDLE newMenuH)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Menu Handle.</td>
</tr>
<tr>
<td>newMenuH</td>
<td>New Menu handle.</td>
</tr>
</tbody>
</table>

Remarks

This should be a call back function from FWMSetztCurrentMenu in GUILIB.

Files

Include: racapi.h
Source: RACDATA.C

Code Declarations

```c
WORD _VMMAPI RACSetSessionMenu(VMMHANDLE oldMenuH,
                                         VMMHANDLE newMenuH);
```

Visual Basic Declaration

```vb
Private Declare Function RACSetSessionMenu Lib "racw32.dll" _
      (ByVal oldMenuH As Long, _
       ByVal newMenuH As Long) As Long
```

Calls

RACGetFirstSession, RACGetNextSession
RACSetStatus

Set the Error Status Code

Syntax

int RACSetStatus(int status)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status code</td>
</tr>
</tbody>
</table>

Remarks

Assigns the current error code that will be returned via RACGetStatus. Normally, this function will be called by functions within RACLib, but can be called by the master application during a non-modal session.

Returns

The value returned is the same as the parameter passed. This usually will be one of the values defined in the RACAPI.H file.

Files

Include: racapi.h
Source: RACDATA.C

Code Declarations

int _RACAPI RACSetStatus(int sts);

Visual Basic Declaration

Private Declare Function RACSetStatus Lib "racw32.dll" _
(ByVal stsCode As Long) As Long

typedef

typedef int (_RACAPIPTR RACSETSTATUS) (int status);
RACSetWorkingPath

Set current working directory.

Syntax

```c
void RACSetWorkingPath(char* path)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>path to make the current directory</td>
</tr>
</tbody>
</table>

Remarks

Set current working directory to whatever is in the parameter. The old parameter is saved and can be restored by the RACRestorePath function.

Files

Include: racapi.h
Source: RACINIT.C

Code Declarations

```c
void _RACAPI RACSetWorkingPath(char *path);
```

Visual Basic Declaration

```vbnet
Private Declare Sub RACSetWorkingPath Lib "racw32.dll" -
(ByVal filename As String)
```

typedef

```c
typedef int (_RACAPIPTR RACSETTINGPATH) (char *path);
```
RACSubClass

Subclass a window’s procedure with RACMainWndProc.

Syntax

PFNWP RACSubClass(HWND hwnd)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>Handle of window</td>
</tr>
</tbody>
</table>

Remarks

Subclass window specified with RACMainWndProc.

Returns

A pointer to the original window procedure is returned. If error, then NULL is returned.

Files

Include: racapi.h
Source: RACTHRD.C

Code Declarations

PFNWP _RACAPI RACSubClass(HWND hwnd);

Visual Basic Declaration

Private Declare Function RACSubClass Lib "racw32.dll" _
(ByVal hwnd As Long) As Long
RACTerminate

Terminate Entry Session.

Syntax

int RACTerminate(unsigned int mode)

Parameter

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
</tr>
</tbody>
</table>

The parameter mode should be TRUE if the entry system should ask the user to save (if necessary) and FALSE to cause the system to shutdown without asking.

Remarks

This will cause the session to release all resources used by the entry system. You must initialize the system before you call this function.

A non-modal session relies upon the controlling application to request termination. Exiting without terminating the system can cause unpredictable results.

Returns

Upon success, returns RAC_SUCCESS or one of the values defined in RACAPI.H.

Files

Include: racapi.h
Source: RACTERM.C

Code Declarations

typedef int _RACAPI RACTerminate(BOOL mode);

Visual Basic Declaration

Private Declare Function RACTerminate Lib "racw32.dll" _
(Val modeval As Long) As Long

typedef int (_RACAPIPTR RACTERMINATE) (BOOL flag);
typedef int (*RACTERMFUNC)(BOOL mode);

Calls

RACClearSessions, RACGetFirstSession, RACGetNextSession, RACInitData, RACResetCmdLineArg, RACRestorePath, RACTermSession, RACUnhook
RACTermSession

Terminate Entry Session.

Syntax

```
int RACTermSession(unsigned int mode,
                    HWND hwnd)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>The parameter mode should be TRUE if the entry system should ask the user to save (if necessary) and FALSE to cause the system to shutdown without asking.</td>
</tr>
<tr>
<td>hwnd</td>
<td>Handle of window</td>
</tr>
</tbody>
</table>

Remarks

This will cause the session to release all resources used by the entry system. You must initialize the system before you call this function.

A non-modal session relies upon the controlling application to request termination. Exiting without terminating the system can cause unpredictable results.

Returns

Upon success, returns RAC_SUCCESS or one of the values defined in RACAPI.H.

Files

Include: racapi.h
Source: RACTERM.C

Code Declarations

```
int _RACAPI RACTermSession(BOOL mode, HWND hwnd);
```

Visual Basic Declaration

```
Private Declare Function RACTermSession Lib "racw32.dll" _
    (ByVal modeval As Long,
     ByVal hwnd As Long) As Long
```

typedef

```
typedef void (_RACAPIPTR RACTERMSESSION)
    (BOOL mode, HWND hwnd);
```

Calls

RACGetSession, RACReleaseSession, RACSetParent
RACThread

New Process operating within another window.

Syntax

```c
int RACThread(HWND hwndParent)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwndParent</td>
<td>The parent window to contain this application.</td>
</tr>
</tbody>
</table>

Remarks

In this entry point, it is assumed that the window's handle passed represents the area that must contain the program. We'll create a frame window that we are familiar with inside this area and start the entry system after usual initialization.

Returns

If successful, **RAC_SUCCESS** is returned otherwise a value from RACAPI.H is returned.

Files

Include: racapi.h
Source: RACTHRD.C

Code Declarations

```c
int _RACAPI RACThread( HWND hwndParent );
```

```vbnet
Visual Basic Declaration
Private Declare Function RACThread Lib "racw32.dll" _
    (ByVal hwndParent As Long) As Long
```

typedef
typedef int (_RACAPIPTR RACTHREAD) (HWND hwnd);

See Also

RACThread, RACInitAll, RACSetHook, RACEnableMenu

Calls

RACCreateMainWindow, RACEnableMenu, RACGetDescription, RACGetIniFile, RACGetStatus, RACInitAll, RACLoadMenuToolbar, RACPostInit, RACProcessError, RACSetHook, RACSetIniFile, RACSetStatus
RACUnhook

Remove Windows message hook.

Syntax

    void RACUnhook(void)

Remarks

This De-installs RACHookProc as a Windows compatible message hook. See that function for more information.

Files

Include: racapi.h
Source: RACHOOK.C

Code Declarations

    void _RACAPI RACUnhook(void);

Visual Basic Declaration

    Private Declare Sub RACUnhook Lib "racw32.dll" ()
**RACUpdate**

Edit an existing WIP entry.

**Syntax**

```c
extern int RACUpdate(char* transaction,
                      char* company,
                      char* lob,
                      char* policy,
                      char* description,
                      int entryMode)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction</td>
<td>A null-terminated string pointer that represents one of the valid [TRANSACTIONS] abbreviations defined in the INI file. For instance: &quot;NB&quot; for New Business; &quot;EN&quot; for Endorsement; and so on.</td>
</tr>
<tr>
<td>company</td>
<td>A null-terminated string pointer that corresponds to the first WIP key value (which is also the first component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Company&quot; value.</td>
</tr>
<tr>
<td>lob</td>
<td>A null-terminated string pointer that corresponds to the second WIP key value (which is also the second component of a line from the FORM.DAT). This parameter is sometimes referred to as the &quot;Line of Business&quot; value.</td>
</tr>
<tr>
<td>policy</td>
<td>A null-terminated string pointer that corresponds to the WIP KeyID value. This parameter is sometimes referred to as the &quot;ID Number&quot; value.</td>
</tr>
<tr>
<td>description</td>
<td>A null-terminated string pointer that should be assigned as the WIP description value.</td>
</tr>
<tr>
<td>entryMode</td>
<td><strong>AFEACTION_UPDATE</strong> - Update existing WIP.</td>
</tr>
</tbody>
</table>

**Remarks**

Attempts to reload for edit an existing WIP entry during a non-modal session. You must initialize the system before you call this function.

**Returns**

**RAC_SUCCESS** (0) is returned if the session completes successfully, otherwise one of the values defined in RACAPI.H is returned.

**Files**

Include: racapi.h  
Source: RACUPDAT.C

**Code Declarations**

```c
int _RACAPI RACUpdate( char far *transaction,
                      char far *company,
                      char far *lob,
                      char far *policy,
                      char far *description,
                      int entryMode)
```
char far *description,
int       mode);

Visual Basic Declaration
Private Declare Function RACUpdate Lib "racw32.dll" _
(ByVal transaction As String, _
ByVal Key1 As String, _
ByVal Key2 As String, _
ByVal KeyID As String, _
ByVal description As String, _
ByVal mode As Long) As Long

Example
The following code is an excerpt from an external program.

switch ( msg ) {
    case WM_COMMAND:
        switch(Param1){
            case ID_UPDATE_OLDPOL:
                rval = RACUpdate("NB",
                "ACME INSURANCE",
                "HEALTH",
                "1402001",
                "New Sample Policy",
                AFEACTION_UPDATE);
                if (rval != RAC_SUCCESS){
                    ... // handle error
                    break;
                }
                ... // continue success
                break;
        }

See Also
RACInit

Calls
RACWipSelection
**RACViewByKey**

View form set specified.

**Syntax**

```
HWND RACViewByKey(HWND hwndParent, 
                 char* key, 
                 int startpage, 
                 int location)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwndParent</td>
<td>The parent window to contain this application.</td>
</tr>
<tr>
<td>key</td>
<td>The archive's key for the transaction.</td>
</tr>
<tr>
<td>startpage</td>
<td>The page number to start display (defaults to 1)</td>
</tr>
<tr>
<td>location</td>
<td>Screen location.</td>
</tr>
<tr>
<td></td>
<td>0 = Left half display (default)</td>
</tr>
<tr>
<td></td>
<td>1 = Right half display</td>
</tr>
<tr>
<td></td>
<td>2 = Top half display</td>
</tr>
<tr>
<td></td>
<td>3 = Bottom half display</td>
</tr>
<tr>
<td></td>
<td>4 = normal window</td>
</tr>
<tr>
<td></td>
<td>5 = maximized window</td>
</tr>
</tbody>
</table>

**Remarks**

It is assumed that the window's handle passed represents another application's main window. We'll create a frame window to contain the form set view and start the entry system after usual initialization.

**Returns**

If successful, the handle of our desktop window is returned. On failure, zero will be returned and the caller should use RACGetStatus to retrieve the last know error code. Error code values are defined in RACAPI.H.

**Files**

Include: racapi.h
Source: RACVIEW.C

**Code Declarations**

```
HWND _RACAPI RACViewByKey( HWND hwndParent, 
                         char *key, 
                         int startpage, 
                         int location );
```

**Visual Basic Declaration**

```
Private Declare Function RACViewByKey Lib "racw32.dll" _
(ByVal hwndParent As Long, _
 ByVal key As String, _
 ByVal startpage As Long, _
 ByVal location As Long) As Long
```

typedef
typedef HWND (_RACAPIPTR RACVIEWBYKEY)(HAB hab,
    char *key,
    int startpage,
    int location);

See Also

RACCreateMainWindow, RACInitAll

Calls

RACViewByKeyStat
**RACViewByKeyHab**

View form set specified.

**Syntax**

```c
HWND RACViewByKeyHab(HINSTANCE hhab,
                     char* key,
                     int startpage,
                     int location)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hhab</td>
<td>Anchor block or instance handle for Windows.</td>
</tr>
<tr>
<td>key</td>
<td>The archive's key for the transaction.</td>
</tr>
<tr>
<td>startpage</td>
<td>The page number to start display (defaults to 1)</td>
</tr>
<tr>
<td>location</td>
<td>Screen location.</td>
</tr>
<tr>
<td></td>
<td>0 = Left half display (default)</td>
</tr>
<tr>
<td></td>
<td>1 = Right half display</td>
</tr>
<tr>
<td></td>
<td>2 = Top half display</td>
</tr>
<tr>
<td></td>
<td>3 = Bottom half display</td>
</tr>
<tr>
<td></td>
<td>4 = normal window</td>
</tr>
<tr>
<td></td>
<td>5 = maximized window</td>
</tr>
</tbody>
</table>

**Remarks**

Is identical to **RACViewByKey** except the application instance is passed instead of a window's handle. Actually, **RACViewData** uses the window's handle only to obtain the application instance.

**Returns**

If successful, the handle of our desktop window is returned. On failure, zero will be returned and the caller should use **RACGetStatus** to retrieve the last know error code. Error code values are defined in RACAPI.H.

**Files**

Include: racapi.h  
Source: RACVIEW.C

**Code Declarations**

```c
HWND _RACAPI RACViewByKeyHab( HAB hhab,
                             char *key,
                             int startpage,
                             int location );
```

**Visual Basic Declaration**

```vbnet
Private Declare Function RACViewByKeyHab Lib "racw32.dll" _
(type=H) (ByVal hInstance As Long, _
          ByVal key As String, _
          ByVal startpage As Long, _
          ByVal location As Long) As Long
```

typedef
typedef HWND (_RACAPIPTR RACRETRIEVEARCHIVE)
(HAB hab,
 char *Key1,
 char *Key2,
 char *KeyId,
 int startpage,
 int location);

See Also
RACCreateMainWindow, RACInitAll

Calls
RACViewByKeyStat
RACViewData

View form set specified.

Syntax

HWND RACViewData(HWND hwndParent,
char* filename,
int startpage,
int location)

Parameter | Description
---|---
hwndParent | The parent window to contain this application.
filename | The name of a file to import.
startpage | The page number to start display (defaults to 1)
location | Screen location.
0 = Left half display (default)
1 = Right half display
2 = Top half display
3 = Bottom half display
4 = normal window
5 = maximized window

Remarks

It is assumed that the window’s handle passed represents another application’s main window. We'll create a frame window to contain the form set view and start the entry system after usual initialization.

Returns

If successful, the handle of our desktop window is returned. On failure, zero will be returned and the caller should use RACGetStatus to retrieve the last know error code. Error code values are defined in RACAPI.H.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

HWND _RACAPI RACViewData( HWND hwndParent,
char *filename,
int startpage,
int location );

Visual Basic Declaration

Private Declare Function RACViewData Lib "racw32.dll" _
(ByVal hwndParent As Long, _
ByVal filename As String, _
ByVal startpage As Long, _
ByVal location As Long) As Long

typedef
typedef int (*RACVIEWFUNC)(HWND hwnd, char *filename, int startpage);

See Also
RACCreateMainWindow, RACInitAll

Calls
RACViewDataStat
### RACViewDataHab

View form set specified.

**Syntax**

```c
HWND RACViewDataHab(HINSTANCE hhab,
                   char* filename,
                   int startpage,
                   int location)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hhab</td>
<td>Anchor block or instance handle for Windows.</td>
</tr>
<tr>
<td>filename</td>
<td>The name of a file to import.</td>
</tr>
<tr>
<td>startpage</td>
<td>The page number to start display (defaults to 1)</td>
</tr>
<tr>
<td>location</td>
<td>Screen location.</td>
</tr>
<tr>
<td></td>
<td>0 = Left half display (default)</td>
</tr>
<tr>
<td></td>
<td>1 = Right half display</td>
</tr>
<tr>
<td></td>
<td>2 = Top half display</td>
</tr>
<tr>
<td></td>
<td>3 = Bottom half display</td>
</tr>
<tr>
<td></td>
<td>4 = normal window</td>
</tr>
<tr>
<td></td>
<td>5 = maximized window</td>
</tr>
</tbody>
</table>

**Remarks**

Is identical to `RACViewData` except the application instance is passed instead of a window's handle. Actually, `RACViewData` uses the window's handle only to obtain the application instance.

**Returns**

TRUE or FALSE (0) is returned depending upon whether the record matches the search criteria.

**Files**

Include: racapi.h
Source: RACVIEW.C

**Code Declarations**

```
HWND _RACAPI RACViewDataHab( HAB hhab,
                             char *filename,
                             int startpage,
                             int location );
```

**Visual Basic Declaration**

```
Private Declare Function RACViewDataHab Lib "racw32.dll" _
           (ByVal hInstance As Long, _
            ByVal filename As String, _
            ByVal startpage As Long, _
            ByVal location As Long) As Long
```

typedef
typedef HWND (_RACAPIPTR RACVIEWDATA) 
(HAB hab, 
char *filename, 
int startpage, 
int location);

Calls

RACViewDataStat
RACViewProxy

Display archive from proxy file.

Syntax

\[
\text{int RACViewProxy(HWND hwnd, char* filename, int startpage)}
\]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwnd</td>
<td>must be a valid window's handle to contain form set display</td>
</tr>
<tr>
<td>filename</td>
<td>path to a valid proxy file</td>
</tr>
<tr>
<td>startpage</td>
<td>page to begin display of form set</td>
</tr>
</tbody>
</table>

Remarks

Display archive from proxy file, RACInit must be called before this function.

Files

Include: racapi.h
Source: RACVIEW.C

Code Declarations

\[
\text{int _RACAPI RACViewProxy(HWND hwnd, char *filename, int startpage);}\]

Visual Basic Declaration

\[
\text{Private Declare Function RACViewProxy Lib "racw32.dll" _ (ByVal hwnd As Long, ByVal filename As String, ByVal startpage As Long) As Long}\]

typedef

\[
\text{typedef int (_RACAPIPTR RACVIEWPROXY)}\]
\[
\text{(HWND hwnd, char *filename, int startpage);}\]

Calls

RACFilterPageForDisplay, RACGetDescription, RACGetIniFile, RACGetStatus, RACProcessError, RACSetCurSession, RACSetStatus
RACWipSelectFunction

WIP selection support function.

Syntax

```c
unsigned int RACWipSelectFunction(void* WipRec)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WipRec</td>
<td>Pointer to a WIP record that is to be evaluated.</td>
</tr>
</tbody>
</table>

Remarks

This is a support function for RACUpdate. The WIP record is first evaluated by the AFEWipSelectFunction to see if it is a valid record for this user. A second test is then performed via RACWipSelection to see if the record matches the remaining search criteria.

Returns

TRUE or FALSE (0) is returned depending upon whether the record matches the search criteria.

Files

Include: racapi.h
Source: RACUPDAT.C

Code Declarations

```c
BOOL _VMMAPIRACWipSelectFunction(void FAR *WipRec);
```

Visual Basic Declaration

Cannot be called from VB

Calls

RACWipSelection
RACWipSelection

Support function for WIP selection

Syntax

unsigned int RACWipSelection(void* WipRec,
char* transaction,
char* company,
char* lob,
char* policy)

Parameter | Description
---|---
WipRec | Pointer to a WIP record that is to be evaluated.
transaction | A null-terminated string pointer that represents one of the valid [TRANSACTIONS] abbreviations defined in the INI file. For instance: "NB" for New Business; "EN" for Endorsement; and so on.
company | A null-terminated string pointer that corresponds to the first WIP key value (which is also the first component of a line from the FORM.DAT). This parameter is sometimes referred to as the "Company" value.
lob | A null-terminated string pointer that corresponds to the second WIP key value (which is also the second component of a line from the FORM.DAT). This parameter is sometimes referred to as the "Line of Business" value.
policy | A null-terminated string pointer that corresponds to the WIP KeyID value. This parameter is sometimes referred to as the "ID Number" value.

Remarks

This is a support function for RACUpdate used to compare a WIP record for matches search criteria.

When a NULL WipRec parameter is used, the remaining parameters are saved as the compare test information. Afterwards the function can be called with valid WipRec parameters to test for matches.

When a valid WipRec parameter is used, the remaining parameters are not used. Rather, the WIP record is compared to the static test information stored previously.

Returns

TRUE or FALSE (0) is returned depending upon whether the record matches the search criteria.

Files

Include: racapi.h
Source: RACUPDAT.C

Code Declarations

```c
BOOL _RACAPI RACWipSelection(void far *WipRec,
char far *transaction,
char far *company,
char far *lob,
```
Visual Basic Declaration

Private Declare Function RACUpdate Lib "racw32.dll" _
(ByVal WipRec As Long, _
ByVal transaction As String, _
ByVal Key1 As String, _
ByVal Key2 As String, _
ByVal KeyID As String) As Long

RACWorkingPath

Establish correct working directory.

Syntax

void RACWorkingPath(void)

Remarks

This function queries the environment for FSIPATH. If a path is specified, the original path is saved and the working directory is changed to the one specified by the environment variable. Calling RACRestorePath can restore original working directory.

If no FSIPATH environment variable is available, the working directory is not changed.

Files

Include: racapi.h
Source: RACINIT.C

Code Declarations

void _RACAPI RACWorkingPath(void);

Visual Basic Declaration

Private Declare Sub RACWorkingPath Lib "racw32.dll" ()

See Also

RACRestorePath

Calls

RACSetWorkingPath
Global Types
**CMDLINE**

typedef struct CMDLINE

This structure contains information parsed from the command line that is used by RACInit.

**Files**

**Source:** RACAPI.H

struct CMDLINE

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char ini[ 129 + 1 ]</td>
<td>The INI file name supplied by the /INI command line switch.</td>
</tr>
<tr>
<td></td>
<td>INI=\fap\mstrres\sampco\fsiuser.ini</td>
</tr>
<tr>
<td>int gMode</td>
<td>The mode state supplied by the /MODE command line switch.</td>
</tr>
<tr>
<td></td>
<td>MODE=retrieve</td>
</tr>
<tr>
<td></td>
<td>MODE=WIP</td>
</tr>
<tr>
<td>char CmdLineUserID[ 129 + 1 ]</td>
<td>The user ID set by the /USERID command line switch.</td>
</tr>
<tr>
<td></td>
<td>USERID=&quot;FORMAKER&quot;</td>
</tr>
<tr>
<td>BOOL bHelpDebug</td>
<td>True if the help system debug /HD command line switch is supplied.</td>
</tr>
<tr>
<td>char menufile[ 129 + 1 ]</td>
<td>The menu file name supplied by the MENU command line switch.</td>
</tr>
<tr>
<td></td>
<td>MENU=men.res</td>
</tr>
<tr>
<td><strong>VMMHANDLE</strong> argvH</td>
<td>Handle to argument list.</td>
</tr>
<tr>
<td>char progName[ 129 + 1 ]</td>
<td>Program name.</td>
</tr>
</tbody>
</table>

**Files**

**Source:** RACAPI.H
MRESULT

typedef long MRESULT

Files

Source: RACAPI.H
PFNWP

typedef WNDPROC PFNWP

Files

Source: RACAPI.H
RAC_ENTRYMODES
typedef enum RAC_ENTRYMODES

Files

Source: RACAPI.H
enum RAC_ENTRYMODES

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFEACTION_CREATE=1</td>
<td>Successful operation</td>
</tr>
<tr>
<td>AFEACTION_UPDATE=2</td>
<td>The main window handle is invalid or 0</td>
</tr>
</tbody>
</table>

Files

Source: RACAPI.H
RAC_ERRNO

typedef enum RAC_ERRNO

Error codes supplied as return values and are used in the RACSetStatus and RACGetStatus functions.

Files

Source: RACAPI.H

enum RAC_ERRNO

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAC_SUCCESS=0</td>
<td>Successful operation.</td>
</tr>
<tr>
<td>RAC_ERR_HWND=-1</td>
<td>The main window handle is invalid or 0.</td>
</tr>
<tr>
<td>RAC_ERR_ACCESS=-2</td>
<td>DLL is currently accessed.</td>
</tr>
<tr>
<td>RAC_ERR_FAP=-3</td>
<td>FAP Failed to initialize.</td>
</tr>
<tr>
<td>RAC_ERR_AFE=-4</td>
<td>Failed to initialize AFELib.</td>
</tr>
<tr>
<td>RAC_ERR_MENU=-5</td>
<td>Failed to load menu.</td>
</tr>
<tr>
<td>RAC_ERR_MODE=-6</td>
<td>The entry mode specified is not valid.</td>
</tr>
<tr>
<td>RAC_ERR_TRANS=-7</td>
<td>Transaction error transaction invalid.</td>
</tr>
<tr>
<td>RAC_ERR_COMPANYLOB=-8</td>
<td>Company or Line of Business error.</td>
</tr>
<tr>
<td>RAC_ERR_ENTRY=-9</td>
<td>Entry failed to start.</td>
</tr>
<tr>
<td>RAC_ERR_WIPCREATE=-10</td>
<td>Cannot create WIP record.</td>
</tr>
<tr>
<td>RAC_ERR_WIPSTATUS=-11</td>
<td>WIP status invalid.</td>
</tr>
<tr>
<td>RAC_ERR_NEWWIP=-12</td>
<td>WIP NEW type not specified in INI.</td>
</tr>
<tr>
<td>RAC_ERR_POLICY=-13</td>
<td>Policy number not specified.</td>
</tr>
<tr>
<td>RAC_ERR_WIPSEL=-14</td>
<td>WIP selection not completed.</td>
</tr>
<tr>
<td>RAC_ERR_FORMSET=-15</td>
<td>Form set load failed.</td>
</tr>
<tr>
<td>RAC_ERR_AFEDLLE=-16</td>
<td>Cannot locate AFE DLL.</td>
</tr>
<tr>
<td>RAC_ERR_NOWIP=-17</td>
<td>No WIP to select from.</td>
</tr>
<tr>
<td>RAC_ERR_AFEDATA=-18</td>
<td>AFEDATA incorrect.</td>
</tr>
<tr>
<td>RAC_ERR_AFEFUNCION=-19</td>
<td>AFE function failed.</td>
</tr>
<tr>
<td>RAC_ERR_AFEINI=-20</td>
<td>AFE failed to load INI.</td>
</tr>
<tr>
<td>RAC_ERR_NEEDTERM=-21</td>
<td>Session already active.</td>
</tr>
<tr>
<td>RAC_ERR_NODLL=-22</td>
<td>Could not load DLL used by CRacLib.</td>
</tr>
<tr>
<td>RAC_ERR_MAXSESSIONS=-23</td>
<td>Maximum &lt;R&gt;RACLib sessions.</td>
</tr>
<tr>
<td>RAC_ERR_SESSION=-24</td>
<td>Invalid RACLib session requested.</td>
</tr>
<tr>
<td>RAC_ERR_MISC=-25</td>
<td>Unknown error in RACLib.</td>
</tr>
</tbody>
</table>
### Files

**Source:** RACAPI.H

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAC_ERR_CANCEL=-26</td>
<td>User canceled.</td>
</tr>
<tr>
<td>RAC_ERR_TERM=-27</td>
<td>Termination error.</td>
</tr>
<tr>
<td>RAC_ERR_NOTUNIQUE=-101</td>
<td>WIP Key components are not unique.</td>
</tr>
<tr>
<td>RAC_ERR_ARC=-102</td>
<td>Archive file failure.</td>
</tr>
<tr>
<td>RAC_ERR_ARCDFD=-103</td>
<td>Archive DFD file failure.</td>
</tr>
<tr>
<td>RAC_ERR_WIPADD=-104</td>
<td>WIP failed to add.</td>
</tr>
<tr>
<td>RAC_ERR_PROXY=-105</td>
<td>Proxy error.</td>
</tr>
</tbody>
</table>
# RACDATA

struct RACDATA

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWND hwndMainFrame</td>
<td>handle to the main frame window</td>
</tr>
<tr>
<td>int mainWndCreated</td>
<td>Was the main window created?</td>
</tr>
<tr>
<td>int loginfailed</td>
<td>Did login fail?</td>
</tr>
<tr>
<td>int racUserClosed</td>
<td>Has the user closed?</td>
</tr>
<tr>
<td>HHOOK hHook</td>
<td>Hook procedure</td>
</tr>
<tr>
<td>char CommandLineUserID[129+1]</td>
<td>Command line user identification</td>
</tr>
<tr>
<td>PAFEDATA pAFEData</td>
<td>AFE data structure.</td>
</tr>
<tr>
<td>short RACRetVal</td>
<td>Latest RACLib return value.</td>
</tr>
<tr>
<td>HWND hwndMain</td>
<td>The main client window</td>
</tr>
<tr>
<td>HWND hwndHelpInstance</td>
<td>The help instance</td>
</tr>
<tr>
<td>int closeBar</td>
<td>Flag indicates whether close bar window was created</td>
</tr>
<tr>
<td>int needTerminate</td>
<td>Flag indicates whether we need to call RACTerminate</td>
</tr>
<tr>
<td>int terminateAllowed</td>
<td>Flag indicates whether we can terminate the active session</td>
</tr>
<tr>
<td>PFNW oldProc</td>
<td>Old window procedure</td>
</tr>
<tr>
<td>PFNW clientProc</td>
<td>New client window procedure</td>
</tr>
<tr>
<td>HWND oldMenu</td>
<td>Old menu handle.</td>
</tr>
<tr>
<td>HWND newMenu</td>
<td>New menu handle.</td>
</tr>
<tr>
<td>char* m_ErrorMsg</td>
<td>Error message string.</td>
</tr>
<tr>
<td>char oldworkdir[129]</td>
<td>Old working directory.</td>
</tr>
<tr>
<td>long ulDriveNum</td>
<td>Drive number.</td>
</tr>
<tr>
<td>int RACTermSession</td>
<td>true after RACClose has executed on this instance.</td>
</tr>
<tr>
<td>VMMHANDLE curmenuH</td>
<td>Current menu handle.</td>
</tr>
<tr>
<td>int modal</td>
<td>Is this a modal session?</td>
</tr>
<tr>
<td>char proxyfilename[129]</td>
<td>Proxy file name.</td>
</tr>
<tr>
<td>char Title[129]</td>
<td>Title for window.</td>
</tr>
<tr>
<td>int NoCaptionWindow</td>
<td>Is there a caption?</td>
</tr>
<tr>
<td>long orgWindowStyle</td>
<td>Original window style</td>
</tr>
</tbody>
</table>

## Files

**Source:** RACLlib.H
# RACWNDCMDS

typedef enum RACWNDCMDS

Window Commands

*Files*

**Source:** RACAPI.H

enum RACWNDCMDS

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD_ID_ZOOMDLG=1013</td>
<td>Bring up zoom dialog.</td>
</tr>
<tr>
<td>CMD_ID_ZOOMIN=1010</td>
<td>Zoom in.</td>
</tr>
<tr>
<td>CMD_ID_ZOOMOUT=1011</td>
<td>Zoom out.</td>
</tr>
<tr>
<td>CMD_ID_ZOOMNORM=1012</td>
<td>Zoom normal.</td>
</tr>
<tr>
<td>CMD_ID_FITWIDTH=1014</td>
<td>Fit to width.</td>
</tr>
<tr>
<td>CMD_ID_FITWINDOW=1072</td>
<td>Fit to window.</td>
</tr>
<tr>
<td>CMD_ID_PAGETOP=1022</td>
<td>Go to top of page.</td>
</tr>
<tr>
<td>CMD_ID_PAGEBOTTOM=1023</td>
<td>Go to bottom of page.</td>
</tr>
<tr>
<td>CMD_ID_PAGENEXT=1033</td>
<td>Go to next page.</td>
</tr>
<tr>
<td>CMD_ID_PAGEPREV=1034</td>
<td>Go to previous page.</td>
</tr>
<tr>
<td>CMD_ID_PAGELEFT=1024</td>
<td>Move page left.</td>
</tr>
<tr>
<td>CMD_ID_PAGERIGHT=1025</td>
<td>Move page right.</td>
</tr>
<tr>
<td>CMD_ID_FORMFIRST=1016</td>
<td>Go to first form.</td>
</tr>
<tr>
<td>CMD_ID_FORMLAST=1017</td>
<td>Go to last form.</td>
</tr>
<tr>
<td>CMD_ID_FORMNEXT=1007</td>
<td>Go to next form.</td>
</tr>
<tr>
<td>CMD_ID_FORMPREV=1008</td>
<td>Go to previous form.</td>
</tr>
<tr>
<td>CMD_ID_LINEUP=1020</td>
<td>Go up a line.</td>
</tr>
<tr>
<td>CMD_ID_LINEDOWN=1021</td>
<td>Go down a line.</td>
</tr>
<tr>
<td>CMD_ID_REFRESH=1009</td>
<td>Refresh display.</td>
</tr>
<tr>
<td>CMD_ID_CASCADE=1002</td>
<td>Cascade windows.</td>
</tr>
<tr>
<td>CMD_ID_TILE=1003</td>
<td>Tile windows.</td>
</tr>
<tr>
<td>CMD_ID_STACK=1004</td>
<td>Stack windows.</td>
</tr>
<tr>
<td>CMD_ID_PRT_FORMSET=1065</td>
<td>Print form set.</td>
</tr>
<tr>
<td>CMD_ID_PRT_FORM=1066</td>
<td>Print form.</td>
</tr>
<tr>
<td>CMD_ID_PRT_PAGE=1067</td>
<td>Print page.</td>
</tr>
<tr>
<td>CMD_ID_PRT_NODLG=1100</td>
<td>Print.</td>
</tr>
<tr>
<td>CMD_ID_COUNTPAGES=1</td>
<td>Count pages.</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>

Files

Source: RACAPI.H
**VMMHANDLE**

typedef void *VMMHANDLE

*Files*

*Source: RACAPI.H*