Peregrine

ServiceCenter

Rapid Application Development (RAD)

Release 6

PART NO: DSC-6-EN35

Peregrine SYSTEMS®

040512

Peregrine Systems, Inc.
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

Contents

Introducing Rapid Application Development (RAD)

Welcome to the Peregrine Systems Rapid Application Development (RAD) for ServiceCenter. The following is an overview of the application development environment, an organizational view of this guide, and information on how to contact Peregrine Systems.

The application development environment serves as a central point in ServiceCenter for all functions and utilities used in building, maintaining, and documenting ServiceCenter applications.

The RAD environment is a set of tools used to develop ServiceCenter applications. All ServiceCenter applications (Incident Management, Change Management, Inventory Management, Service Management, and utilities such as Forms Designer and Database Manager) are written in the RAD language.

RAD is a powerful, yet easy to use environment that is best categorized as a fourth generation language (4GL). The syntax of the language is similar to Basic, with a number of extensions.

Sample screens and examples

The sample screens and examples included in this guide are for illustration only, and may differ from those at your site.

RAD elements

RAD contains the following elements.

Element	Description
Application	A ServiceCenter application is a collection of RAD command panels linked together to represent a *business process flow*, such as opening a problem ticket.
Command panel	A command panel is a visual, fill-in-the-blanks form that represents a discrete processing unit, one that interacts with a user, or updates a data base record. Some panels contain a number of statements and expressions for computing results or making decisions.

Statement	A statement controls how and in what order objects are manipulated. Typical statements are:
	`$X = $Y + 1`
	`IF $X = $Y THEN $UPDATE = 0 ELSE $UPDATE = 1`

Expressions	Expressions are sequences of operators and operands that compute a value. Typical RAD expressions are:
	`$X * 25`
	`$Y > 32`

Operators	Operators designate operations to perform with operands. Typical operators are +, *, /, indicating addition, multiplication, and division respectively.
Operands	Operands are either variables, literals, or function calls. Variables start with a $ sign.
Literals	Literals represent constants in a program, such as the number 3.1415, or a name "Your Name Here".

Structure of this guide

The RAD Guide contains the following sections:

- *Introducing Rapid Application Development (RAD)* provides an overview of RAD, a discussion of terms used in this guide, and definitions for common system tray buttons used in the program.
- *Chapter 1, Application Development Editor* describes the starting point for all application development activity. It involves using and accessing the Application Development Encyclopedia and outlines the steps for creating an application.
- *Chapter 2, Application Development Tools* provides instructions for using the Compare Application Utility and how to unload, copy, rename, delete, and print an application.
- *Chapter 3, Command Panels* provides a list of the RAD Command Panels, plus the definition, format, parameters, programming considerations, and examples of each command panel.
- *Chapter 4, Display Panel Conversions* describes the procedures for converting custom RAD applications using rio and fdisp panels to use the display panel.
- *Appendix A, Command Panel List* is an alphabetical list of RAD command panels.

For information about system language, go to the **Reference** > **System language** topics in *Administering ServiceCenter* online help. Topics include data types, expressions and assignment statements, forming literals, and using operators. Included are all current RAD functions and their definitions with a detailed discussion of each function, including the programming format, factors to consider when using, and examples for each function.

For information about the RAD Debugger and the procedures for converting custom RAD applications using rio and fdisp panels to use the display panel, go to the **Concepts** > **Debugger** topics in *Administering ServiceCenter* online help.

1

CHAPTER

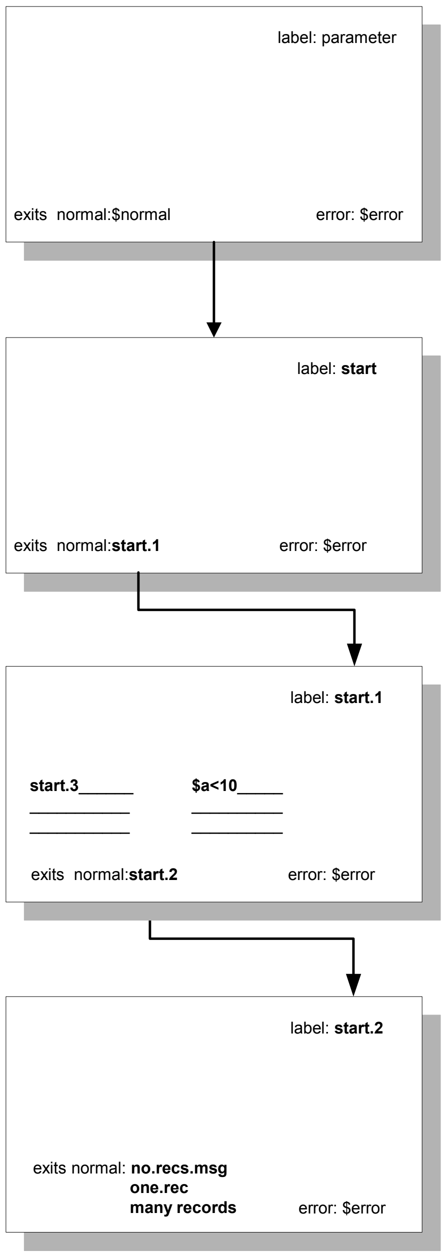
Application Development Editor

A RAD Application contains application panels (records) linked together in a logical flow. Each panel performs a specific function. When the function is complete, the application exits to another panel. The field value within the exit is a **label name** of another panel.

The **parameter** panel defines local variables passed to it by a calling application. In general, two kinds of local variables exist: those used within an application, and those used as exits. Typically, the exit variables are **$normal** (in the normal exit) and **$error** (in the error exit).

The first **command** panel**,** where execution begins, is always labeled **start**. Execution continues at a panel's normal exit (or another exit depending on conditions). When that exit is defined as **$normal**, execution of the panel is complete. If an error occurs, then the **$error** exit is taken.

The following flowchart provides an example of how an application moves from panel to panel.



Parameter Panel

The parameter panel is always the first panel in an application. This panel defines all parameters (local variables) received or returned by the application.

Process Panel

The first command panel in an application always has a label of start.

Decision Panel

Sample execution from a decision panel:
If the condition statement on a decision panel (for example, $a<10) evaluates to true, execution continues at start.3.
If the condition evaluates to false, the panel takes its normal exit.

Select Panel

Sample execution from a select panel:
If no records are selected, execution continues at no.recs.msg.
If one record is selected, execution continues at one.rec.
Otherwise, execution continues at many.records.

RAD Development Editor

The RAD Development Editor is an online software development tool that:

- Allows controlled modification of application records.
- Provides a development environment capable of quick and efficient code generation.
- Has a complete set of tools to maintain and develop RAD applications.

The RAD Development Editor has two modes, Edit and View.

Note: The Edit mode is only available when you purchase a RAD license.

Edit mode

The RAD Edit mode allows access to all RAD Editor functions, giving you complete application creation and modification capabilities. If any changes are made to the application, you can:

- Update the application and store the changes. Modifications are not actually executed until the application is compiled.
- Re-edit the changes before continuing.
- Ignore the changes.

These choices lessen the chance of modifications being lost accidentally and help alert you to any unintentional changes. You are always notified and must confirm any changes to the application records.

View mode

The View mode restricts functionality. You cannot make changes to any panels in the application. Also, none of the system tray buttons that control editing are present when in View mode.

Application Development Encyclopedia

The Application Development Encyclopedia is the reference point for all application development activity. From this record, you can:

- Create and edit applications with the Application Development Editor.
- Interface with ServiceCenter utilities such as the Database Manager.
- Use application support tools such as the Compare Application Utility and Map Utility.

Each RAD application contains an Encyclopedia Record. ServiceCenter automatically maintains the history information related to creations, edits, and compiles whenever one of those operations occurs. The arrays for files, formats, menus and sub-applications can be updated during a ServiceCenter applications upgrade by adding data, but not removing any of your own. The remaining fields are currently used for documentation purposes only.

Data fields

The following table contains the field label, field name, and a brief description.

Label	Field Name	Description
Name	application.name	A required field containing the name of the application.
Type	type	An optional field containing the type of application (utility, sub-application, and so on). This can be any text.
Title	application.title	An optional field containing the title of the application. This can be any text.
Description	appl.desc	An optional array containing descriptive information about the application in free-form text. Text information placed on the first full line carries forward to the comments of the **parameter** panel when the application is created.
Help Category	help.category	An optional field containing the help category of the application.
System	system	An optional field containing the system type of the application.

Label	Field Name	Description
Files	files	An optional array containing the names of ServiceCenter files that the application uses. When in the Edit Application mode, the Database Dictionary option can access the Database Dictionary record of a file listed in this field.
Data	data	An optional field containing a true or false flag determining whether data is contained in the file.
Formats	formats	An optional array containing the names of ServiceCenter formats that this application uses. When in the Edit Application mode, the Forms Designer option can access the Format record of a form listed in this field.
Menus	menus	An optional array containing the names of ServiceCenter menus that call this application.
Sub Apps	sub.appl	An optional array containing the names of ServiceCenter sub-applications that this application calls.
Authored By	author	A required field containing the ServiceCenter operator ID of the original author of the application. The system fills in this field at the time the Create Application option is executed.
Creation Date	creation.date	A required field containing the date and the time the application was created. The system fills in this field at the time the Create Application option is executed.
Edited By	editor	A required field containing the ServiceCenter operator ID of the last operator to edit the application. The system fills in this field when any updates are saved.
Compiled By	compiled.by	A required field containing the ServiceCenter operator ID of the last operator to compile the application. The system fills in this field when the application is compiled.
Last Edit Date	last.edit.date	A required field containing the date and the time the application was last edited. The system fills in this field when any updates are saved.

Label	Field Name	Description
Last Compile Date	last.compile.date	A required field containing the date and time the application was last compiled. The system fills in this field when the Compile Application option is executed.
Current Release Level	current.release.level	The application version level. This field is updated during a ServiceCenter upgrade.
Release Date	current.release.date	The application version release date corresponding to the current release level. This triggers the prompt to enter revision notes whenever you exit any newly compiled application. This file is updated during a ServiceCenter upgrade.

Options menu items

The following table shows the option and a brief description.

Option	Description
Print	Starts the printing process.
Copy/Rename	Starts the copying or renaming process.
dbDict Utility	Accesses the Database Dictionary.
Database	Accesses the Database Manager.
Forms Designer	Accesses the Forms Designer Utility.
Revision History	Allows you to view revision comments.
Compare Application	Starts the Compare Application Utility process.
Export/Unload	Allows you to export this record into a file for importing into a spreadsheet, or unload this data set for loading into another ServiceCenter system.
Automatic Update	Updates the Encyclopedia Record to match the current version of the application. Automatically updates the Files, Data, Formats, Menus, and Sub Apps arrays in the record. Also enters the Release Date, which in turn activates the revision notes.

Option	Description
Create Revision	Takes you to the Revision Tracking Record screen, where you can enter comments and Software Change Request (SCR) information to associate with a change.
View Revisions	Allows you to view a QBE list of revisions, if more than one is present, otherwise displays the revision.

Accessing an application

Once you access an existing application, you can browse, edit, delete, and test.

To access an existing ServiceCenter application

1 From the System Navigator, click **Toolkit** > **RAD Editor**.

2 Open the Encyclopedia Record for the selected application:

- Enter the name of the application you want to open in the **Application** field and click **Search**.

  or

- Click **Search** and select an application from the displayed QBE list.

Viewing an application

The View mode allows you to look through an application without making any changes. No editing options are available.

To view an application

1 Click **View** in the Encyclopedia record to display the application in View mode.

The **parameter** panel of the application opens. There are labels after each field showing the input parameter name that calls the application when outside RAD.

2 Click the buttons to navigate the panels.

Button	Action
Next	Moves through the application panel by panel using normal exits unless you place the cursor in a field with a different exit, such as a decision panel.
Previous	Displays the preceding panel.
Goto	Accesses the search dialog box that allows you to search for any panel in the application.

3 Type the name of a RAD panel in the field and click the appropriate button.

Button	Action
Cancel	Cancels the action.
Search	Locates panels by label. If more than one panel label beginning with the search word exists, the system displays a QBE list of possible matches. **Enter** performs the same function.
Params	Returns to the application's **parameter** panel.
Start	Returns to the application's **start** panel.
Exit	Locates panels by an exit name. Use this query to locate all panels in an application that exit to a specific panel, such as a **display** panel or an **msg** panel. **Note:** Use exact strings.
Type	Goes to the **type** of panel named in the search screen (for example, **select**, **rinit**, **display**). If more than one panel of the requested type is in the application or if the search field is left empty, a QBE list is shown.

Editing an application

The **Edit** mode allows you to modify an existing application. Additional system tray buttons and Options menu items are available in Edit mode that are not available in View mode.

To edit an application

1. Access the Application Encyclopedia for the application you want to edit.
2. Click **Edit**.

Note: The **Edited by** and **Last edit date** fields in the Encyclopedia Record automatically populate with the operator ID of the editor and the date and time of the edit. An exclusive lock is applied to the application, preventing others from accessing it using the Edit mode.

The **parameter** panel of the application is displayed first. Use the buttons in the system tray to navigate through the application.

Important: If you do not intend to edit the application, return to the Application Encyclopedia and click **View**. Unintended changes to an application may have unpredictable results.

System tray buttons

The following system tray buttons are options on various RAD panels:

Button	Function
Compare All	Compares all the panels of the new version of the application named with all the panels of the old version.
Comp Panel	Compares the old and new versions of a single panel named in the Panel field of the Compare Application form.
View Old	Displays the old version of the panel named in the Panel field of the Compare Application format.
View New	Displays the new version of the panel named in the Panel field of the Compare Application format.
Clear	Clears all data from the panel comparison fields (Unmatched, Deleted, New and Matched), and prepares the utility to perform another comparison.
Print Panel	Prints the Detail Listing of Differences screen.

Button	Function
Enter	Stores the data entered in the format into the data record of the current application.
Print All	Prints all the panel records currently displayed.
Delete Encl	From the Delete option, deletes only the Encyclopedia Record.
Delete Appl	From the Delete option, deletes only the Application and Code Records.
Delete All	From the Delete option, deletes all Records Associated with the Application (application, code, format, and encyclopedia).
Appl Only	From the Delete option, deletes only the Application Records.
Back	Returns to the Encyclopedia record. If any changes occur to the panel prior to selection, a prompt screen appears to confirm the update of that panel.
Previous	Accesses the last panel viewed regardless of cursor position. If any changes occur to the currently viewed panel, a prompt screen appears to confirm the changes prior to the display of the previous panel. A list of the last 200 panels viewed within the current application is maintained. The Flow Prev function uses this list to access the previous panel.
Next	Accesses the next panel in the process flow named in the **normal** exit field. If any changes occur to the currently viewed panel, a prompt screen appears to confirm the changes prior to the display of the next panel. From the parameter panel, this function always locates and displays the **start** panel.
New	Adds a new panel to the application. If any changes occur to the panel prior to selecting **New**, a prompt screen appears to confirm the update of that panel. This is not available in view mode.
Save	Updates and stores any changes made to the currently displayed panel. This is not available in view mode.
Parameter	Begins the process of adding new parameters to the **parameter** panel. This is not available in view mode.
Compile	Compiles the application and stores the compiled version in the Code record for the application. If the Code record already exists, the record is updated. If the record does not exist, a new Code record is added. Any errors that occur during the compile are displayed at the bottom of the screen. Click **M** (**Message**) in the toolbar at the top of your screen to view messages pertaining to the Compile operation that are hidden. The **Compile** button is not available in view mode.

Button	Function
Goto	Locates and allows access to various panels by panel or form name, panel type (parameter, start), or any panel exiting to a named panel. If any changes occur to the panel prior to selection, a prompt screen appears to confirm the update of that panel.
Delete	Deletes the current application.
Find	Searches for a panel or string. **Note:** For the Find button to execute the standard ServiceCenter **Find** feature, a link record for application must exist.
View	Displays the application in View mode. No editing is available in this mode.
Edit	Displays the application in Edit mode. The application can be updated and compiled.

The parameter panel

Each RAD Application must have its own definition panel, called the **parameter** panel. The **parameter** panel is always the first panel in an application (label: parameter). This panel defines all parameters (also known as parameter variables) that the application receives or returns, along with the exits defined for all available completion conditions.

When an application is created, a default parameter panel is built and added to the format file. In addition, a parameter application record is added to the application file, referencing the parameter panel record previously created.

The parameter panel provides definitions for these standard fields:

Field	Definition
application	Name of application.
label	Always *parameter* on this panel.
comments	Free form comments.
normal exit	Exit if successful completion occurs.
error exit	Exit if unrecoverable error occurs.

Use the default parameter panel for a very simple application, one which neither receives nor returns any parameters when executed. For more complicated applications, you can add fields to the parameter panel, defining additional data to received or return. All fields used on a parameter panel must be the name of an existing field in the Application Database Dictionary.

Note: Be sure to choose an **application** field defined with the data type matching the type you need.

Parameter panels allow you to pass data to and from an application in a controlled method using local variables. All parameter panels have at least two default parameters, the normal exit and error exit.

Conventions

Use the following guidelines when creating parameter panels.

- Use upper case variable names for local variables.
- Preface application and variable names with a code that describes the general application area. For example, use us. for universal ServiceCenter applications.
- Preface all input variables as $INTO.<*variable*> and all return variables as $RETURN.<*variable*>. This makes tracing an application easier.
- Add all parameters *before* you begin to write your application.

Options menu items

Option	Action
Print RAD	Prints the application file to the printer defined in the ServiceCenter Operator record.
comments	Accesses the Extended Comments screen for a particular panel. This allows for more detailed documentation regarding the functionality and purpose of the panel.

Command panels

The first command panel used in an application has a default label of **start** regardless of what type of command panel is added.

Enter the required information into the fields on the command panel.

Best practices indicate that you always use a **process** panel for your **start** panel. This allows you to modify the application or to add debugging statements later.

Options menu items

Option	Action
Print Panel	Sends the panel currently being viewed to the printer defined in the current user's operator record.
Copy Panel	Copies a particular panel to a new panel within the application. This option is not available in View mode.
comments	Accesses the Extended Comments screen for the current panel. The comments function allows you to enter more detailed information regarding the functionality and purpose of that panel.
Find String	Finds all the panels within the application that contain a certain string.
Unload Fix	Unloads the current panel, as file application_panel, to the fix directory you define.
Text Edit	Accesses Text Edit mode. This option is not available in View mode.
Find/Replace String	Allows you to replace all instances of a string within the current panel.
Insert Process	Click this button to insert a process panel into the RAD application.

Note: **Copy Panel** and **Text Edit** are in the Edit Mode of a Parameter panel, but not in the View Mode.

Locating command panels

The procedures for moving through an application in the View Mode are the same as in the Create/Edit Mode.

Navigation buttons

Use the navigation buttons in the system tray to move backwards and forwards sequentially through the command panels in a RAD application.

- Click **Next** to access the panel named in the **normal** exit. To access panels using **conditional** exits, the cursor must be in the **exit** field. For example, place the cursor in a field in the **Exit** array of a **decision** panel.
- Click **Previous** to access the last panel viewed regardless of cursor position. If you made changes to the currently viewed panel, a system prompt confirms the changes (*update*) before returning to the previous panel.

Goto feature

Use the **Goto** feature to locate command panels by panel or format name, panel type (*parameter, start*), or any panel exiting to a named panel.

To locate a specific panel from within a RAD application

1 From any panel within an application, click **Goto**.

2 Type the full name, exit name, or panel type you want to **display**, or enter a partial string with which to do a *starts with* search. When searching by name, click on the appropriate command button to locate the panel.

   If the search finds more than one panel, a QBE list of all matching entries is returned.

3 Double-click on the panel you want to display.

Find String option—single application

Use the **Find String** option to search an application for all command panels in which a specified string is found.

To initiate a string search

1 Select **Options** > **Find String** in a command panel.

2 Type the text you want to locate in the **Search string** field.

3 Click **Search** to display a QBE list of panels where the string is found.

4 Select a panel from the list by clicking on the labeled button.

5 Select **Options** > **View Last Find** to display the QBE list of matching strings again.

Note: The system saves this list until the next string search is executed.

Find String option—application suite

You can search for all instances of a string in applications that start with a particular string (for example, **apm**, **cm3r**). This is helpful when looking for all instances of a variable or literal within an application suite.

To initiate a string search within an application suite

1 Select **Options** > **Find String** in any command panel.

2 Enter the string to search for in the **Search string** field.

3 Replace the name of the current application in the **RAD Application** field with a pound sign (#) followed by the first few characters of an application *suite* name (for example, `cm3`, `apm`).

4 Click **Search** to display a QBE list of all panels where the string is found.

Note: The list of panels returned cannot be used for direct selection. To print the screen, select **Print Screen** from the File menu.

5 To search for all instances of a string in ALL applications, leave the **RAD Application** field blank and click the **Search** button.

These operations may take some time because all RAD panels are searched for the string.

Editing array elements in a command panel

To edit elements of an array in a command panel

1 From the command panel, select **Options** > **Text Edit.**

A new set of buttons appears in the system tray.

2 Choose the operation you want to perform from the following buttons:

Button	Action
Insert Line	Adds a blank line within an array. Position the cursor on a line within an array in any panel and click **Insert Line**. A blank line is inserted *above* the line where the cursor appears, and the existing lines are moved down accordingly.
Delete Line	Removes a line within an array. Position the cursor on the line within an array to be deleted and click **Delete Line.** The line on which the cursor had been located is deleted, and the remaining lines are moved up accordingly.
Mv/Cpy Line	Moves or copies a line within an array to another position within the same array. To move or copy a line within an array to another line within the same array, position the cursor at the beginning of the line and click **Mv/Cpy Line**. The system prompt instructs you to place the cursor on the destination line and click either **Move** or **Copy**. Move—Moves the selected line from its original position to a position indicated by the cursor. Copy—Copies the line to the position indicated by the cursor and leaves the line in its original position. **Note:** When moving and copying, you must first use the **insert line feature** to create a vacant destination line.

3 Click **Back** to return to the View Mode of the panel.

4 Click **Save** to enter your changes in the Application Development Encyclopedia.

Creating a new application

You can build a new application with or without a system-supplied template.

Note: If you do not intend to modify an application, access it using the View mode.

Building an application without a template

To create a new application without the template

1. Click Toolkit > **RAD Editor** to open the RAD Editor dialog box.
2. Enter the name of your new application in the **Application** field and click **New**.

   A prompt asks if you want to use an application template to create your new application.

   Note: If you do not provide a name, the system prompts you for one and takes you directly to the Application Development Encyclopedia.

3. Click **No**.

   A parameter panel with blank exits opens.

4. Create exits and enter any additional pertinent data.
5. Click **Save** to save your changes.
6. Click **New**.

   You are prompted to select the next type of panel in the flow.

7. Enter a panel type and click **Add**.

Building an application with the template

To use the template to create an application

1. Click Toolkit > **RAD Editor** to open the RAD Editor dialog box.
2. Enter the name of your new application in the **Application** field and click **New**.
3. Click **Yes** when prompted about the template.

   The template panels include:

   - Parameter panel
   - Process panels:
     - Start
     - Cleanup
     - Normal exit panel—sets `$exit` to **normal**
     - Error exit panel—sets `$exit` to **error**
     - Error exit message panel—sets actual error message text
   - Decision panel (for exit)—branches based on value of `$exit`

- Message panel (for error exit)—sends the error message

   A parameter panel for your new application is displayed with exits already established.

4. Click **Next.**

   A **start** panel is displayed with no **normal** exit defined.

5. Click **Goto.**
6. Enter **decision** in the Goto search field.
7. Click the button that locates this type of panel.

   The template **decision** panel is displayed with the default panel name of **decide.exit** in the label field.

8. Select a descriptive name for this panel.
9. Enter appropriate exits and conditions for this panel.
10. Click **New.**

    A prompt asks if you want to save the application.

11. Click **Yes.**

    A dialog box asks what type of panel you want to create.

12. Enter the name of the next panel in your flow.
13. Click **Add.**

    The specified panel opens.

14. Enter a name for the normal exit.
15. Click **Add.**

    The following message is displayed in the status bar:
    ```
    Application record successfully added.
    ```

16. Proceed to build your application using this base.

    Note: You can also create an application from the Application Development Encyclopedia form. Enter the name of the application and any other pertinent information and click Add.

Extended Comments option

Use the Extended Comments option to enter more detailed information regarding the panel and application.

To access the Extended Comments feature

1 Select **comments** from the Options menu.

Note: If an asterisk appears after the **comments** option, extended comments already exist for that panel.

The Extended Comments form opens.

Note: You must be in Edit mode for the **comments** option to be available.

2 Enter text describing this particular panel.

3 Click **Back** to return to the panel itself.

4 Click **Save** to save your Extended Comments.

Note: The values in the **application** and **panel** name fields cannot be modified.

Compiling an application

Use **Compile** to compile an application. After saving your application, click **Compile** to generate the executable.

Errors

Errors that occur during the compilation generate messages at the bottom of the screen after the attempt.

The following are typical error messages:

Message	Definition
exit to unknown label: <*label*>	The system detected an exit with no corresponding panel label.
panel not reached: <*label*>	The system detected a panel in the application where no exit exists.

Incorrect exit

To correct an exit error

1 Click **Goto** to locate the panel named in the error message.

2 Do one of the following to correct the error:

- Modify the exit field to connect it to a panel within the application.
- Delete the panel if it unnecessary.

3 Recompile the application.

4 Click **Back**.

A Revision History form for the application opens.

5 Complete the description of the revision you made in the **Description of Change** field.

6 Click **Update** to return to the Application Encyclopedia.

Unconnected panels

When the compiler fails to find an exit to one of the panels in the application, a form opens listing the panels that cannot be reached.

1 Place the cursor in the field of a panel you want to delete.

- Click **Delete All** to delete all the command panels listed from the application.

  A list of all panels on the form is displayed.

  or

- Click **Delete Panel** to delete a single, selected panel.

  A read-only copy of the command panel you selected is displayed.

2 Click on the **Confirm** button.

The panel listed is deleted and you return to the application.

3 Click **Compile** to recompile the application.

4 Click **Back** to leave the RAD Editor.

The Revision History form opens.

5 Enter an explanation of any changes you made to the application and assign a **Change #**, if applicable.

6 Click **Update** to return to the Encyclopedia Record for the application.

Testing an application

Once you successfully compile an application, you can test it. The **Test Application** function verifies that the application performs as expected (executing file updates, deleting, displaying forms).

Note: Compile the application at least once for this option to execute.

To test an application

1. Click **Test** in the Application Development Encyclopedia record of the application.
2. In the **name** array (actual field names and not the variables used), type any parameter names to pass to the application for this test.
3. In the **value** array, type any values to be assigned to those fields.
4. Click **Proceed** or press **Enter** to run the application.

Adding new parameters to the parameter panel

By adding additional parameters to a parameter panel, values can pass into an application from a calling application and then passed back.

The following example shows how you can add a general-purpose application that has the following specifications:

- The System Administrator wants to track all queries executed against the Incident database.
- The administrator want to track all databases that Database Manager accesses for performance monitoring.
- In the future, the administrator wants to track queries from other sources.
- The System Administrator creates the **trackquery** database containing four fields:
  - filename
  - time
  - query
  - operator

In designing the application **track.queries**, you can obtain values for the **time** and **operator** fields using the **tod**() and **operator**() functions, respectively, from within the application.

Values for the fields **filename** and **query** are dependent on the calling application. In the Database Manager **QBE** application, the fields are `$FILE.LOCAL` and `$query.select`. In the `apm.search.problems` application, the fields are `$L.file` and `$L.sql`. For future sources, you can do one of the following:

- Hardcode a specific RAD program for every case.
- Write one general-purpose application, passing parameters for fields whose source may vary (the preferred method).

New field names

To add new fields to a parameter panel

1. Access the parameter panel.

   The only parameters defined at this point are the **normal** and **error** exits.
2. Click **Parameter**. The new parameter dialogue box appears.
3. Enter a name for the new parameter in the **Label** field
4. Select a value for the **Input Field** field from the drop-down list. Select a field name that corresponds to the type of field you are passing. (A **file** variable has a **type** of record.)
5. Click **Add** to display the parameter panel containing the new field.
6. Repeat the process to add more fields.
7. Assign variables to the newly added parameter fields, then build the application.
8. Click **Save** to store the data in the Encyclopedia Record for the application.

Adding new command panels to an application

To add a new command panel

1. From the System Navigator, click **Toolkit** > **RAD Editor**.
2. Access the application where you want to add command panels.

   Note: You must be in Edit mode to add panels to an application.

3. Use **Next, Previous**, and **Goto** to locate where you want your new panel.
4. Click **New**.

   A dialog box asks you to choose the type of panel you want to add.

5. Type the name of the panel you want to create in the field provided.
6. Click **Add**.

   The label field of the new command panel contains the value in the **normal** exit field (or the exit field currently selected with cursor) from the previous panel if that panel is not already created. If a panel with this label already exists, the **label** field is left blank.

Copying a panel

To copy a particular panel to a new panel within the current application

1. From Edit mode, select **Options** > **Copy Panel**.

   The Application Panel Copy prompt screen appears. The **Copy From Application** and **Panel** fields default to the current application and panel. You can override these values.

   The **Copy To Application** field is set to the current application. You cannot override this.

   Note: If the cursor is located on a panel name (including the normal exit), that name appears in both the Copy From and Copy To fields. If the cursor is located in any other field, the name of the current panel is displayed.

2. Enter the panel name to which this panel is to be copied.
3. Do one of the following:
   - Click **Copy** to copy the panel.
   - Click **Cancel** to cancel the process and return to the previous screen.

Note: If you leave the **Copy To Application** panel name blank, ServiceCenter uses the **Copy From Application** panel name.

Important: Panels copied from other applications may contain references to local variables. Check all statements carefully to ensure that all variables are properly referenced.

2 Application Development Tools

CHAPTER

Topics in this section include:

- *Compare Application Utility*—the utility that compares different versions of the same application.
- *Unloading an application*—procedure for unloading all the elements of an application to an external file.
- *Copying or renaming an application*—procedure for copying or renaming an existing RAD application.
- *Deleting an application*—procedure for deleting a RAD application from your system. Controls allow you to delete individual elements of the application or the entire application.
- *Printing an application*—procedure for printing the panels of a RAD application.

Compare Application Utility

The RAD Comparison Utility is a tool that compares one version of a ServiceCenter RAD application to a different version of the same application. Neither application needs to reside in the same Application Library. The **Compare Application Utility** quickly and accurately determines what changes were made to a RAD application. This utility is helpful during the ServiceCenter upgrade process for users with customized changes to ServiceCenter RAD applications.

To access the compare application utility

- From any Application Encyclopedia, click **Options** > **Compare Application.**

System tray buttons

The Compare Application Utility uses the following command buttons.

Button	Action
End (F1)	Returns you to the previous screen.
Compare All (F2)	Compares all the panels of the new version of the application named with all the panels of the old version.
Comp Panel (F4)	Compares the old and new versions of a single panel named in the Panel field of the Compare Application form.
View Old (F5)	Displays the old version of the panel named in the Panel field of the Compare Application format.
View New (F6)	Displays the new version of the panel named in the Panel field of the Compare Application format.
Clear (F7)	Clears all data from the panel comparison fields (Unmatched, Deleted, New and Matched), and prepares the utility to perform another comparison.

Data fields

The Compare Application Utility contains the following data fields.

Fields	Description
Old Version Source File	Defines the logical file containing the application records (RAD panels) of the application specified in the field Old Application Name. The default is application.
New Version Source File	Defines the logical file containing the application records (RAD panels) of the application specified in the New Application Name field. The default is application.

Fields	Description
Old Application Name	Defines the name of the application that resides in the Old Version Source File. The application specified in the New Application Name field uses the Old Version Source File as a model. **Note:** If the Comparison Utility is called from an Encyclopedia Record, the Old Application Name field defaults to the application name displayed in the Encyclopedia Record.
New Application Name	Defines the name of the application residing in the New Version Source File that is compared against the application specified in the Old Application Name field. **Note:** If the Comparison Utility is called from an Encyclopedia Record, the New Application Name field defaults to the application name displayed in the Encyclopedia Record.
Panel	Names a panel to use for comparison checking. The Panel field views a specified panel in either the old or new versions of the application.
Matched panels †	Contains a list (array) of panel names that detects no differences between the old and new version of the RAD application.
Unmatched panels †	Contains a list (array) of panel names with differences between the old and new version of the RAD application.
Deleted panels †	Contains a list (array) of panel names present in the old version of the application that are not present in the new version.
New panels †	Contains a list (array) of all panel names present in the new version of the application that are not present in the old version.

† The list completes after you click **Compare All**.

Important: Do not make manual changes to the contents of the † fields.

Detail Level results are available for all panels that have differences between the old and new versions. You can view the comparison results online and print the results at the operator's default printer. You also can view the old and new versions of each panel online.

Defining or modifying source file definitions

The **Old Version Source File** and **New Version Source File** fields define the name of the logical file that contains the RAD panels for the applications to be compared. Although both fields default to **application**, you can override the defaults.

Note: The standard version of ServiceCenter contains one Application Library. If you maintain two or more Application Libraries, you are responsible for allocating and controlling the functionality of these files and for defining those routines necessary for the exchange of data between files.

After accessing the RAD Comparison Utility, tab to the **Old Version Source File** input field and type the name of the logical file that contains the panels of the old application.

Follow the same procedure for specifying a different **New Version Source File**, if necessary.

Defining application names

The definitions for the old and new application names must be in place before attempting to compare either the entire application or a single application panel.

To define application names

1. In the initial Compare Application form, type the name of the old version of the application in the **Old Application Name** field.
2. Type the name of the new version of the application in the **New Application Name** field.

Comparing entire applications

To compare entire applications

1. From the RAD Comparison Utility, add the proper data to the following:
   - **Old Version Source File**
   - **New Version Source File**
   - **Old Application Name**
   - **New Application Name**

2 Click **Compare All**.

ServiceCenter displays summary lists of Matched Panels, Unmatched Panels, Deleted Panels, and New Panels where they apply.

3 To make selections from these lists and display additional panel information, use the command buttons in the system tray.

Print All appears in the system tray after you click **Compare All**.

4 Click **Print All** to print all the panel records currently displayed.

You can view old and new versions of application panels, review the detailed comparison results for panels in the Unmatched Panels list, and print all the panels.

Note: If you decide to compare a different application, you must click **Clear** before proceeding to reset all controls before processing begins.

Printing a report

To get a printed report of the Comparison Results, click **Print All**. The print job routes to your default printer.

The report displays the following information.

This panel	Displays a summary page listing of
Matched Panels	■ The names of all panels that matched.
Unmatched Panels	■ The names of all panels showing a change. ■ Printouts of the old and new versions of each unmatched panel. ■ The comparison results of each panel.
Deleted Panels	■ The names of all deleted panels. ■ A printout of each deleted panel.
New Panels	■ The names of all new panels. ■ A printout of each panel.

Comparing single panels

You can compare a single panel either before or after the entire application has been compared. Make sure that you enter valid data on the new and old versions of the application in all the source file and application name fields.

To view a detailed comparison of a single panel after the entire application has been compared

1. Select the desired panel in the Unmatched Panels array in the comparison utility.
2. Click **Comp Panel** to view the results of comparing a single, unmatched panel.

To view a detailed comparison of a single panel before comparing the entire application

1. In the **Panel** field, type the name of the panel.
2. Click **Comp Panel**.

Important: You must select the **Panel** field for ServiceCenter to locate the specified panel. A border surrounds the selected field.

The Detail Listing of Differences screen shows the exact differences between the old and new versions of the panel. The results are presented in the same manner for both online viewing and in printed form. Every page shows the name of the Old and New Version Source File names and the names of the Old and New Application Name fields.

System tray buttons

A single panel contains the following buttons.

Button	Action
Old Panel	Displays the old version of the application panel being compared.
New Panel	Displays the new version of the application panel being compared.
Print Panel	Prints the Detail Listing of Differences screen.

Scalar field differences

When differences are noted between the old and new version of a **scalar** field, the values have the following form:

```
normal CHANGED FROM invalid.assign
normal CHANGED TO invalid.assign.test
```

Normal, in this example, is the name of the scalar field, and invalid.assign is the value of that field in the old version.

Array field differences

Differences between the old and new version of an array field contain the following syntax.

Syntax	Description
Line	Constant
XX	Element number of the added or deleted array.
Action	Changes to arrayed field elements always have the ADDED TO or DELETED FROM phrase. If you make a change to an element of an array, the old version of the element is considered deleted and the new version of the element is considered added.
Prompt	Name of the prompt on the RAD panel which corresponds to the changed array field.

Viewing versions of a panel

You view old and new versions of a panel from the following screens:

- **Compare Application**—After the entire application has been compared, type the name of the panel in the **Panel** field or select the panel from one of the lists. Click **View Old** or **View New**.
- **Detail Listing of Differences**—Click **Old Panel** or **New Panel** to open a version of the current panel.

Printing a Detail Listing of differences

To print the Detail Listing

▶ Click **Print Panel**.

If Active Notes is enabled, a dialog box indicates that the report spooled and is scheduled to print on your default printer.

Continuation lines

If the contents of a scalar field or an element of an arrayed field are greater than 72 bytes, the comparison results for those lines are displayed in their entirety with the use of continuation lines. All continuation lines start with three asterisks (***).

Continuation lines are in both online and printed comparison results. The last two characters of a Detail Line are the first two characters of the next continuation line.

```
LINE 2 DELETED FROM: statements (old version)
   if ($assignment.orig="") then ($agm="No assignment group entered;us
  ***ng default \""+name in $assignment+"\".")
```

Unloading an application

The **Unload Application** option unloads the entire application into an external file. Each application component is unloaded: Parameter panel, Encyclopedia Record, command panels, and the Code Record (the compiled version of the application).

Note: Any scmessage record that is referenced in the RAD application is also unloaded.

To unload an existing application

1. From the application Encyclopedia Record, select **Options** > **Export/Unload** to open the Unload/Export Facility form.
2. In the **External File Name** field, type the name of the destination file for your application.

   While a file extension is optional, adding one such as .UNL makes the unload file easier to identify. ServiceCenter unloads the file to either the server or client, depending on the preference set in the **Window** > **Preferences** > **Client side Load/Unload** option. If you select the Client side option, the system automatically puts the unload file in the client installation directory unless you specify a different path.

   Note: If you do not select the **Window** > **Preferences** > **Client side Load/Unload** check box, the system uses the server RUN directory as the default unload location unless you specify a different server path.

3. Set the Export Mode, Dbdict Load and Record Load parameters as desired.

   For more information on these fields, go to the **Concepts** > **Database** topics in *Administering ServiceCenter* online help.

4. Click **Unload Appl** in the system tray to unload your application to the destination file specified.

Copying or renaming an application

Use the RAD Editor to access the Encyclopedia Record for the application.

To copy or rename an application

1 From the application Encyclopedia Record, select **Options** > **Copy/Rename** to open the Application Copy/Rename form.

2 Type a new name in the **NEW Application name** field.

3 Choose either **Copy** or **Rename**.

You return to the Application Encyclopedia. A message in the status bar describes the operation, either copy or rename.

Warning: When you rename an application, references to it in other applications are NOT renamed.

Deleting an application

Use the RAD Editor to access the Encyclopedia Record for the application.

To delete an existing application

1 From the application Encyclopedia Record, click **Delete** in the system tray.

The Application Deletion prompt screen shows the name of the application to delete and the description from the Encyclopedia Record.

2 Select one of the delete options from the buttons in the system tray.

Warning: When you delete an application, references to it in other applications are NOT deleted.

Button	Action
Abort	Ends the delete process and returns you to the Encyclopedia record.
Delete Encl	Deletes only the Encyclopedia record.
Delete Appl	Deletes only the Application and Code records.
Delete All	Deletes all records associated with the application: application, code, format, and encyclopedia.
Appl Only	Deletes only the Application records.

Printing an application

The printout of an application consists of a copy of every panel in the application file. The panels print in alphabetical order by label name.

To print an existing application

1 From the application the Encyclopedia Record, select **Options** > **Print** to open the Run Report Exerciser form.

  The **Primary File Query** field automatically contains the name of the report.

2 Do one of the following:

  - To print the report immediately, clear the **Print** (**false=spool only**) check box.
  - To spool the report for printing later, select the **Print** (**false=spool only**) check box.

3 Click **Run**.

  If you designate the client printer or the **Print** (**false=spool only**) option is clear, the job spools immediately to that printer. This completes the print function.

  If you designate the server printer or select the **Print** (**false=spool only**) option, the Report Scheduler opens.

4 In the **Time to Print** field, type the date and time to print the report.

  The default is the current date and time.

5 To schedule reports to print automatically throughout the year, select an option from the Repeat Interval structure.

6 Select a printer.

7 Click **Run** to activate the Report Scheduler.

  The Report Maintenance panel opens, showing you a read-only overview of the report you scheduled. Click **View** to page through the report.

3 CHAPTER Command Panels

This section describes RAD command and parameter panels that combine to create applications. You create a RAD application by stringing together a series of panels to make a single, compiled unit that performs a process. Each panel in a RAD application performs a task, such as displaying a form, declaring a variable, or performing calculations. A complete RAD application contains the following components:

- Format—parameter panels
- Application—command panels
- Code—compiled application
- Encyclopedia—encyclopedia record

While you can use command panels in any order to create an application, all RAD applications must have two initial panels:

- Parameter
- Start

Parameter panel

The parameter panel passes variable information and declares exits. The parameter panel's format field contains the name of the application and contains the word **parameter** in its **label** field.

Other applications can use parameter panels as a command panel. This feature allows modular applications and effectively extends the RAD command set. To call another application from any other application:

- Use the **call** command panel if the name of the application is not known until run-time.
- Use that application's parameter panel.

Start panel

The start panel must contain the word **start** in its label. The start panel serves as an entry point of an application, and is the first panel that the application runs.

Common fields

Each panel has five common fields.

Field	Description
application	Name of the application. The system automatically populates this field.
label	User-generated label for the command panel. Each label in an application must be unique. Labels cannot contain spaces.
comments	User-generated comments about the function and purpose of each labeled command panel. This field is optional.
normal	Exit the application takes if no errors occur.
error	Exit the application takes if an unexpected error occurs.

Note: These are the first five fields in any command panel.

Functional groups

The command panels in this section are in groups that represent their programming functions for Basic and Advanced use. See *Appendix A, Command Panel List* for an alphabetical listing of all available panels.

The definition of each command panel includes the following information:

Term	Description
Format name	Syntax used to access the command panel in the Encyclopedia Record. The format name is entered in the name field of the New Panel dialog box. The format name is displayed in the heading at the top of each page.
Parameters	Fields in which information is entered to configure the command panel. Note that within the parameters, the type entry (for example, character, array, and so on) is listed as well as the elements of the entry.
Factors	Programming considerations for configuring and using the command panel.
See also	Related command panels.

Basic facilities

User interaction panels

Panel	Function
rio	Displays a form.
fdisp	Displays a list of records in a set.
mb.ok	Displays a modal dialog box with a message in it.
mb.yes.no	Displays a modal dialog box that asks the user a question.
msg	Sends a message.
thread.start	Enables multiple windowing.
wopen	Opens a window.
wselect	Selects a specific window.
wclose	Closes a window.
print	Prints a database record.

Processing panels

Panel	Function
process	Executes a list of BASIC-like statements.
decision	Takes an exit when a condition is true.
call	Calls another application.
loop	Repeatedly executes statements while a condition is true.
lock	Locks a system resource.
unlock	Unlocks a system resource.

Database panels

Panel	Function
rinit	Initializes a file.
genquery	Introduced with ServiceCenter 4.0 to replace the **qbe** panel. It translates data entered in the form to an SQL-like query.
select	Selects the records that satisfy a query.
next	Reads the next record in a set.
previous	Reads the previous record in a set.
count	Counts the number of records in a set.
radd	Adds a record.
rupdate	Updates a record.
rdelete	Deletes a record.

Advanced facilities

Processing—Advanced

Panel	Function
attach	Attaches another process to create a task.
compile	Compiles an application.
configure	Configures the terminal.
event.send	Sends a system event to a thread.
fmt	Defines a form by painting a screen.
priority	Sets a task's priority.
return	Returns an application to the prior state of the last **rio** panel.
signal	Creates a software interrupt in an attached task.
sleep	Pauses for a designated number of seconds.
user.login	Logs a user on the system.

Database—Advanced

Panel	Function
fcreate	Creates a database file.
fremove	Removes a database file.
freset	Deletes all data records from a database file.
detect.keyed	Determines if the query passed is fully keyed.
fregen	Regenerates the indices of a database file.
project	Copies like-named fields from one record to another.

External communication

Panel	Function
connect	Begins a communication session with an external device or file.
dde	Calls any of the supported DDE functions.
read	Reads data from a communication session.
write	Writes data to a communication session.
disconnect	Terminates a communication session.

SQL interface

Panel	Function
sqlcrt	Coverts a P4 file to a DDL file.
sqlddl	Displays the DDL (Data Definition Language).
SQLexecute	Executes SQL commands on the RDBMS.
SQLprocedure	Executes a stored procedure on the RDBMS.
SQLselect	Selects data from a table on the RDBMS.
SQLfetch	Retrieves the next row returned by a query from a RDBMS table.
SQLgeterr	Retrieves error codes and messages from the RDBMS.
sqlunl	Unloads a P4 file.

User interaction panels

The user interaction grouping has 10 command panels: **rio**, **fdisp**, **mb.ok**, **mb.yes.no**, **msg**, **wopen**, **wselect**, **wclose**, **print**, and **rdelete**.

rio

	Description
Definition	The **rio** (Record Display/Input) command panel displays a record using a specified format. Use the **rio** panel to: ■ Declare file variables. ■ Identify the form to display. ■ Define button options.

Parameter	Field	Definition
	File Variable	Record variable (file variable) you want to display or input. The field is optional except when you are working with a Database.
	Form	Name (character) of the format through which you want to display the record. The field is required.
	Window Title	Prompting message (character) that appears at the top of the screen.
	I/O Condition	Condition (boolean) that determines whether record input can occur. **True** allows record input, **False** does not allow it. **Important**: Do not leave this field blank.
	option #	Number (array of numbers) that identifies the option; for example, *3* for Back.
	description	Description (array of characters) of an Options function.
	exit	Exit (array of labels) to take if the option is chosen.
	condition for option	Condition (array of boolean) that must evaluate to **true** before you can use the option.

	Description
Factor	■ If you leave the **Form** field blank, or specify a form that does not exist, the error exit is taken. ■ Use Forms Designer to create a form that the rio panel uses. Do not end the form name with *.g* unless you create both a text and a GUI form. In either case, do not specify the form in the **rio** panel as the *.g* version. ■ The <**enter**> key uses the normal **exit**. The normal **exit** is also the next panel you reach if the **condition for record input** field is left blank. If you leave the **condition for record input** field blank, and the **normal** exit name is the same as this panel name, you create an infinite loop. ■ Use **-1** as the first option number to enable bitmaps on system tray buttons. Set condition to **gui**(). ■ Only options 1-12 are available to the **rio** panel in the text mode. ■ Option numbers greater than 199 appear in the Options menu in GUI environments. You can define balloon help as the text after a semicolon (;) in the descriptor field.
See also	N/A

fdisp

	Description	
Definition	The **fdisp** (File Display) command panel shows a list of records using the default QBE format or a specified format. Records are listed using the specified format.	
Parameter	Field	Definition
	File	File variable containing the list of records to display.
	Form	Form name (character) used to display the record (defaults to filename.qbe).
	Window Title	Window title that appears at the top of the screen.
	Time Limit	Time for a partial keyed query.
	option #	Number (array of numbers) of the option key.
	description	Description (array of characters) of the function of the option key that displays on the bottom of the screen in the option line.
	exit	Exit (another panel label) the application takes when the option is chosen (array of labels).
	condition for option	Condition (array of boolean) that determines whether the corresponding option key is to be displayed.

	Description
Factor	■ If you leave the **Form** field blank, and no form exists named *<filename>*.qbe, the system creates a temporary default form to display the records. This form displays the first 15 characters of the first fields of each of the first five keys. ■ If you want to create your own file display form, use Forms Designer to create one. Do not end the form name with .g unless you create both a text and a GUI form. In either case, do not specify the form in the **fdisp** panel as the .g version. ■ Lines at the top of the form that do not contain input fields are treated as headers and are not repeated for each record in the list. For gui versions, the table widget most commonly is used for a file display form. ■ The list of records is created using the **select** operation. ■ The <**enter**> key is the normal **exit.** The record that the cursor selects becomes the current record in the file. ■ If a form name is specified and that form does not exist, the **error** exit is taken.
See also	**rio**, **select**, **previous**, **next**

mb.ok

	Description	
Definition	The **mb.ok** (Message Box-OK) panel creates a standard modal dialog box containing a message. The user must acknowledge this message by clicking OK before proceeding.	
Parameter	Field	Definition
	Message	Message you want displayed in the dialog box. There is a limit of 200 characters once all the variables, if any, expand.
	Type	Dialog box types: ■ information ■ warning ■ error The only difference is an icon before the message.
Factor	■ The normal exit is followed when the user clicks OK in the dialog box. ■ The error exit is followed if there is an error in processing this panel. ■ This is a GUI-mode only panel. If you access this panel while in text mode, a normal text message is displayed.	
See also	msg, mb.yes.no	

mb.yes.no

	Description
Definition	The **mb.yes.no** (Message Box-Yes/No) panel creates a modal dialog box that asks the user a question. The dialog box has two buttons: **Yes** and **No**, with the possibility of a third button, Cancel.

Parameter

Field	Definition
Message	Message you want displayed in the dialog box. There is a limit of 200 characters once all the variables, if any, expand.
Button On?	Logical field that determines if a Cancel button is displayed in the dialog box. If this field evaluates to **true**, the button is displayed. If this field evaluates to **false** (the default), the button is not displayed.

Factor

- The **Yes exit** is followed when the user clicks **Yes**.
- The **No exit** is followed when the user clicks **No**.
- The **Cancel exit** is followed if the user clicks **Cancel**.
- If there is an error processing the panel, the **error exit** is taken.
- This is a GUI-mode only panel. If you access this panel while in text mode, the error **exit** is taken.
- If the user closes the window using either the System menu or pressing the <Esc> key, the panel takes the **Cancel exit** if turned on; otherwise it takes the **No exit.**

See also **msg, mb.ok**

msg

	Description	
Definition	The msg command panel displays a statement on the last line of the screen that the user specifies.	
Message level	Message	Description
	1 = information message	This message provides information only. The user need not take any action.
	2 = action message	This message requires the user to take some sort of action.
	3 = error message	This message indicates that an error occurred in an application.
Parameter	Field	Definition
	enter message level (1,2, or 3)	Appropriate level (number) for the message.
	Text	Text of the message (character).
	send message to users	Name of the users (character) receiving the message. If you want to send a message only to a local user, leave this field blank. To send the message to: ■ A specific user, enter the attached resource name of the user. ■ Multiple users, enter the list of attached resource names. If the field is left blank, the message is sent to you.
	message name	Name (character) of the message source, for example, **database**, for the database application. If entered, this optional field is included with the application and panel names in parentheses at the end of the message.
	message number	Message number (number) referencing the source message. If entered, this optional field is included with the application and panel names in parentheses at the end of the message.

	Description
Factor	■ If the user is logged on, the message appears on the screen. ■ The attached resource name is set by the login application to be the user's login name. ■ Messages are stacked on the message line in the order of error, action and information.
See also	**mb.yes.no**, **mb.ok**

wopen

	Description	
Definition	The **wopen** (Open Window) command panel creates a new window within a parent window.	
Parameter	Field	Definition
	parent window name	Name (character) of the parent window.
	create window name	Name (character) of the window you want to create.
	position of new window top: right: bottom: left:	Indicates whether the new window is in the top, bottom, left, or right side of the parent window.
	create as either percent of parent or lines or columns	Percentage of the size of the parent window or the number of lines or columns for the new window.
Factor	■ The new window must be greater than 0% of the parent window and less then, or equal to, 100% of the parent window. ■ The system selects the newly opened window for future I/O and displays all subsequent **rio** and **fdisp** command panels in the new window until it encounters a **wselect** or **wclose** in the application. ■ The screen contains three system-defined windows automatically: Main Window, Option Window, and Message Window. ■ If the parent window name is left blank, it defaults to the Main Window. ■ If a **wopen** panel is used in an application, all possible exits, including error exits, must encounter a **wclose** panel.	
See also	**wclose**, **wselect**, **rio**, **fdisp**	

wselect

	Description	
Definition	The **wselect** (Select Window) command panel chooses a specified window.	
Parameter	Field	Definition
	window name	Name (character) of the window to select.
Factor	■ The system displays all subsequent **rio** and **fdisp** command panels in the selected window until it encounters another **wselect** or **wclose** in the application.	
See also	**wopen, wclose, rio, fdisp**	

wclose

	Description	
Definition	The **wclose** (Close Window) command panel closes a specified open window.	
Parameter	Field	Definition
	window name	Name (character) of the window you want to close.
Factor	N/A	
See also	**wopen, wselect, rio, fdisp**	

print

	Description	
Definition	The **print** (Print to Spool File) command panel allows you to print a database record to a spool file.	
Parameter	Field	Definition
	file variable	Record variable (file variable) you want to print.
	format	Name (character) of the format to use to print the record.
	spool file	Spool record variable (file variable) to put in the record.
	skip to line	Line number (number) on which to begin printing the format. This parameter defaults to the current line.
	lines per page	Total number of lines per page.
	maximum line width	Maximum line width.
	exit at top of page	Label of the panel to exit when the top of the page is reached.
Factor	■ When you execute a print command that skips to the next page, the system automatically adds the old page to the spool file. ■ The **print** command panel keeps its own page counter (field name **page**) and line counter (field name **eop**) in the spool file variable. ■ Run the **print.start** application before starting a print, and run **print.end** when you finish using the **print** command panel. ■ The **print.detail** application handles long records (those exceeding **lines.per.page**).	
See also	N/A	

thread.start

	Description	
Definition	Use the **thread.start** command panel to run multiple RAD applications independently of one another in different windows.	
Parameter	Field	Definition
	Application Name	Name of the RAD application to run.
	Thread ID	Thread ID returned as a variable.
	Pane Name	No longer used.
	Parameters	Appropriate parameter names and values for the RAD application being called; similar to the use of parameters in the **call** panel.
Factor	■ The **thread.start** panel differs from a **call** panel, which can return a value from the application being called. ■ Threads are not preemptive. Each thread must yield to the previous thread as it generates. ■ There is no communication between threads; each thread operates independently. ■ This panel is available on all platforms.	
See also		

Processing panels

The processing grouping has six command panels: **process**, **decision**, **call**, **loop**, **lock**, and **unlock**.

process

	Description	
Definition	The **process** panel initalizes or sets variables, or processes expressions.	
Parameter	Field	Definition
	statements	Processing **statements** (array of statements) you can use in the application.
Factor	■ Executes top to bottom and left to right. ■ A semicolon (;) separates **Statements** on the same line.	
See also	N/A	

decision

	Description	
Definition	Use the **decision** command panel to branch to any of a set of panels, depending on conditions set at the time you execute the **decision** panel, without user interaction. This command panel takes exits corresponding to a given true condition. If none of the conditions are true, the normal exit is taken. **Note:** This command panel is the only way to establish conditional branching in RAD. Specifically, you cannot use a **process** panel to force branching.	
Parameter	Field	Definition
	exit	Exit (another panel label) to take if the corresponding condition is **true**.
	condition for exit	Condition (array of booleans) that must evaluate to **true** before the application takes the corresponding **exit**.
Factor	■ This command panel is the equivalent of a nested IF-THEN-ELSE statement in BASIC.	
See also	N/A	

call

	Description	
Definition	The **call** command panel summons another command panel or application whose name is unknown or may change based upon values (that is, **menu.manager**) at run time. You also can use this panel to call another subroutine whose name is known at run time. (It can replace the parameter panel normally used to call a subroutine.)	
Parameter	Field	Definition
	name of application to call	Full name (character) or variable of the command panel or application to be called.
	parameters to pass: names	Name (array of characters) of each parameter in the command panel or application to be called. The name is the field name in the application file.
	parameters to pass: values	Value (array of characters) of each parameter in the command panel or application to be called.
Factor	N/A	
See also	N/A	

loop

	Description	
Definition	The **loop** command panel repeatedly executes statements while the specified condition is true.	
Parameter	Field	Definition
	initialization expressions	Statements to execute one time before the condition is evaluated (for example, $L.i=1; $1=lng($L.list)).
	condition for loop execution	Condition to evaluate (boolean) to determine if the body should be executed (for example, $L.i=>$1).
	body of loop	List of statements (arrays of statements) to execute repeatedly as long as the condition evaluates to true.

	Description
Factor	■ Be sure that the condition for the loop execution is set to false somewhere in the loop statements; otherwise, the loop may continue to run indefinitely. (for example, `$L.i+=1`)
See also	N/A

lock

	Description	
Definition	The **lock** (Lock Resources) command panel requests exclusive or shared use of a system resource, or checks to see if a system resource already has a lock on it.	
Parameter	Field	Definition
	Lock Resource Name	Name (character) of the system resource.
	Exclusive?	Indicate whether you are making a request for exclusive use of the resource (boolean).
	Immediate?	Indicate whether you wish to wait for use of this resource.
	Location	ServiceCenter distributed site where the lock will reside.
	Lock Denied	Name (label) of the **exit** to take if the resource is in use and you do not wish to wait.
Factor	■ Be sure to unlock a resource after locking is completed. ■ Any exit, including **error** exits, must unlock a locked resource. ■ By convention, the **Lock Denied** exit must call the application **lock.denied.msg**.	
See also	**unlock**	

Using system resources

When more than one program modifies the same resource, you can prevent simultaneous use of the resource. RAD automatically controls all resource sharing in its built-in functions. However, if you increase this control (for example, by locking an entire file, a single record, or list of records for an update application), you must use the lock panel to keep others from using a resource while you are modifying it.

The lock panel ensures that system resources are used serially. The lock panel, by itself, cannot prevent simultaneous use of a system resource. The lock panel marks the record as locked, but does not actually lock the record.

The lock panel asks the system to assign control of a system resource to the active task. The system determines the status of the resource, and either grants the request by returning control to the active task or delays assignment of control by placing the active task in the wait condition.

Exclusive and shared requests

You can request exclusive or shared control of the resources for a task by using the different options available on the lock panel. If use of the resource results in the resource being modified, you must request exclusive control of the resource.

Processing the request

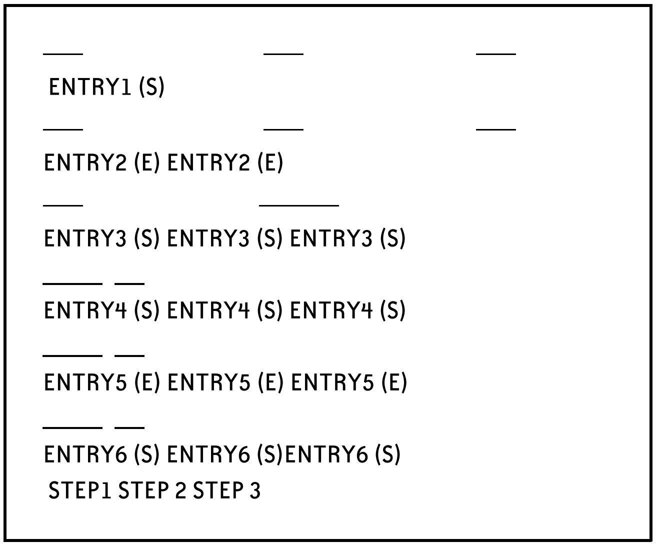
The system constructs a list and enters a request in the list for the task that is active when the lock panel is issued. When the system receives a request, it is entered in an existing list. If no list exists, the system builds a new list. The system places the requests in the order they are received. A task receives control of a resource according to two factors:

- The position of that task's request on the list.
- The exclusive control or shared control requirements of the request which caused the entry to be added to the list.

If the **Immediate?** field is `false`, then the request remains in the list until it is granted. If the conditional request field is `true`, then the application continues to the panel specified in the **exit if lock denied** field if the system fails to grant the lock immediately.

The following example provides the status of a list. The **S** or **E** next to the entry indicates that the request is **S**hared or **E**xclusive control. The task that `ENTRY1` (Step 1) represents is assigned the resource. The task that established `ENTRY2` is for exclusive control. The corresponding task is placed in the wait condition, along with the tasks represented by all the other entries in the list.

Eventually, control of the resource is released for the task represented by ENTRY1, and the entry is removed from the list. In Step 2, ENTRY2 is first on the list, and the corresponding task is assigned control of the resource. Since the request that established ENTRY2 was for exclusive control, the tasks represented by the other entries in the list are kept in the wait condition.


ENTRY1 (S)
ENTRY2 (E) ENTRY2 (E)
ENTRY3 (S) ENTRY3 (S) ENTRY3 (S)
ENTRY4 (S) ENTRY4 (S) ENTRY4 (S)
ENTRY5 (E) ENTRY5 (E) ENTRY5 (E)
ENTRY6 (S) ENTRY6 (S)ENTRY6 (S)
STEP1 STEP 2 STEP 3

The system uses the following general rules:

- A shared task runs if no tasks with an exclusive lock on that resource are running.
- A task with an exclusive lock runs only if no task is running which has locked that resource. When ENTRY2 releases the resource, both ENTRY3 and ENTRY4 run, since they both have shared locks. This is represented by Step 3. Again, ENTRY5 and ENTRY6 are waiting because ENTRY5 has requested exclusive control.

Note: From the time a resource is locked until it is unlocked, all error exits go to an unlock panel.

unlock

	Description	
Definition	The **unlock** (Unlock Resource) command panel unlocks a system resource.	
Parameter	Field	Definition
	Unlock resource name	Name (character) of the resource you want to unlock.
	Location	ServiceCenter distributed site where the lock resides.
Factor	■ For more information, see *lock* on page 69.	
See also	lock	

Database panels

The database grouping has nine command panels: **rinit**, **genquery**, **select**, **next**, **previous**, **count**, **radd**, **rupdate**, and **rdelete**.

rinit

	Description	
Definition	The **rinit** (or Record Initialize) command panel initializes a file variable and variable and binds it to a specific database file.	
Parameter	Field	Definition
	file variable	File variable to associate with the file being initialized
	filename	Database file (character) to which the file variable will be bound
	Location	ServiceCenter distributed site, if applicable
	Read Only?	Boolean statement or expression to indicate whether this panel should be read only.
Factor	■ A rinit must be executed before taking any other actions with a ServiceCenter file.	
See also	N/A	

genquery

	Description	
Definition	The **genquery** (Translate QBE) command panel changes data entered as a QBE query into a query expression.	
Parameter	Field	Definition
	file variable	File variable containing the QBE query.
	query	Name (character) of the variable to contain the query string.
Factor	■ The query returned is suitable for use by the **select** command panel. ■ If the contents of the file variable is empty, the query returned is **true**. ■ This panel replaces the **qbe** panel, used in versions prior to ServiceCenter 4.0.	
See also	**rio**, **select**	

select

	Description	
Definition	The **select** (Select) command panel chooses the records in a file which satisfy a query. The selected records can be sorted by any field or combination of fields. The first record in the list is immediately available for processing.	
Parameter	Field	Definition
	file variable	Enter the file variable from which to select records. This file variable has been initialized with the Record Initialization (**rinit**) command panel.
	query	Query expression (string) that specifies the criteria for selecting records for the source file.
	sort fields	Names of the fields (array of characters) used to sort the selected records (optional). If these sort fields are filled in, the query uses this as the key, which may adversely affect response time.
	sort descending	Reverses the normal sort order. This only works for files mapped to SQL.
	exit if no records selected	Exit (label) to take if there are no records matching the selection criteria.
	exit if one selected by query	Exit (label) to take if there is only one matching record.

	Description
Factor	■ A true or **true** query selects all records; a false or **false** query selects none. ■ You can sort on one or more fields. ■ SQL follows its own sorting rules. ■ You can sort efficiently only on keys. ■ To use a pre-defined key of multiple fields, you must enter the same field names in the same order as entered in the dbdict. ■ The query field is parsed at run time. Since parameter (local) variables cannot be parsed at run time, query expressions with parameter variables must be input in a special manner. The procedure is to enter an expression that is parsed at run time which generates a string with no local variables. Therefore, to check for the field name being equal to the local variable `$menu.name`, you enter “`name=\”“+$menu.name+”\“` ”. If the local variable $menu.name contains the value MAIN MENU at run time, this results in the query: name=“MAIN MENU”. ■ If you issue a true query against a file where the first key allows NULLs, or specifically sort on a key that allows NULL entries, you do not retrieve any records whose key is NULL. These records are not in the index. ■ Use caution when defining the query, since any NULL values appended to or inserted in a NULL query result in a NULL query, which is treated as **true**. ■ Invalid queries take the **error** exit.
See also	**project**, **rinit**, **next**, **previous**

next

	Description	
Definition	The **next** (Read Next Record) command panel reads the next record in a list of records.	
Parameter	Field	Definition
	file variable	Name (**file variable**) containing the list of records.
	exit if no next record	Label (label) of the **exit** to take if no *next record* is found.
Factor	■ **Select** must be executed before the first use of this step. The list of records are selected using the **select** command panel.	
See also	**previous**, **select**	

previous

	Description	
Definition	The **previous** (Read Previous Record) command panel reads the previous record in a list of records.	
Parameter	Field	Definition
	file variable	File variable (file variable) to read
	exit if no previous record	Label (label) of the exit to take if no previous panel is found
Factor	■ **Select** and **next** must be executed before the first use of this step. The list of records selected using the **select** command panel.	
See also	**next**, **select**	

count

	Description	
Definition	The **count** (Count) command panel counts the number of records in a file.	
Parameter	Field	Definition
	file variable	Name (file variable) of the file you want to count.
	query	Query string used to select the records.
	record count variable	Variable in which to store the output of the panel (number).
Factor	■ The file variable must be initialized prior to using this panel. This will reposition the record pointer to the last record in the set. ■ Use fully keyed queries for the best performance.	
See also	**rinit**, **select**	

radd

	Description	
Definition	The **radd** (Record Add) command panel inserts a record into the database.	
Parameter	Field	Definition
	file variable	Record variable (**file variable**) you want to add.
	exit if error detected by trigger	Exit (another panel label) to take if a trigger runs and encounters an error.
	exit if record contains all null keys	Exit (another panel label) to take if the record has all null keys.
	exit if record contains invalid null keys	Exit (another panel label) to take if the record has invalid null keys.
	exit if record contains invalid duplicate keys	Exit (another label panel) to take if the record has invalid duplicate keys.

	Description
Factor	■ Before you can add a record to a file, the key fields must be filled in correctly. ■ You can use the RAD application **bu.record.add** in place of the radd command panel. ■ This panel executes any triggers designated `before add` and `after add`.
See also	**rupdate**, **rdelete**

rupdate

	Description	
Definition	The **rupdate** (Record Update) command panel incorporates new information into a record in the database.	
Parameter	Field	Definition
	file variable	File variable to update.
	exit if error detected by trigger	Exit (another panel label) to take if a trigger runs and encounters an error.
	exit if record contains all null keys	Exit (another panel label) to take if all keys in the record are null.
	exit if record contains invalid null key	Exit (another panel label) to take if the record contains an invalid null key.
	exit if record contains invalid duplicate key	Exit (another panel label) to take if the record contains an invalid duplicate key.
	exit if record has been changed	Exit to take if the record has been modified since the last time it was read.
	exit if record deleted	Exit (another panel label) to take if the record has been deleted since last read.
Factor	■ You must use **select** or **next** to get a record before you can update that record. ■ You cannot update a record unless you have the current version of the record. ■ You can use the RAD application **bu.record.update** in place of the **rupdate** command panel. ■ This panel executes any triggers designated before update and after update.	
See also	**radd**, **rdelete**	

rdelete

	Description	
Definition	The **rdelete** (Record Delete) command panel removes a record from a database file.	
Parameter	Field	Definition
	file variable	Record variable (file variable) to delete.
	exit if error detected by trigger	Exit (another panel label) to take if a trigger runs and encounters an error.
	exit if record had been changed	Exit (another panel label) to take if the record has been changed since it was last read.
	exit if record has been deleted	Exit (another panel label) to take if the record has been deleted since it was read.
Factor	■ You cannot delete a record from the database if you do not have the most current version of that record. ■ You can use the RAD application **bu.record.delete** in place of the **rdelete** command panel. ■ This panel executes any triggers designated before delete and after delete.	
See also	**radd**, **rupdate**	

Processing—Advanced panels

The advanced processing grouping has 10 command panels: **attach**, **compile**, **configure**, **event.send**, **fmt**, **priority**, **return**, **signal**, **sleep**, and **user.login**.

attach

	Description	
Definition	The **attach** (Attach Another Process) command panel starts a program as a subtask. For example, the **attach** panel can start RAD's scheduler.	
Parameter	Field	Definition
	name of program to attach	Name (character) of the program you want to start.
	name of attached resource	Name (character) assigned to this task. This name can be used by the **signal** panel to terminate a task.
	parameters to pass	Parameters (array of characters) to pass to the program, if any.
	wait for completion?	Indicates whether or not to complete the task before returning control (boolean). This option is only supported on Unix and Windows NT/2000 servers.
Factor	■ If **true** is entered in the wait for completion? field, your program does not stop until the attached task is complete. ■ You can send messages to attached tasks using the **msg** panel. ■ The login application attaches all user processes with attached resources set to the user's login name. ■ On Windows-based clients, the **attach** panel issues a WinExec to start a new process on the client machine.	
See also	**signal**, **msg**	

compile

	Description	
Definition	The **compile** command panel translates the application you specify into RAD pseudo code that executes at run time.	
Parameter	Field	Definition
	name of application	Name (character) of the application to be compiled
	Set local symbol table only	Establishes a local symbol table based on the parameter panel **Note:** The above parameters convert the `$1`, `$2`, and `$3` variables used to position local variables on the stack to their variable names inside RAD.
	Cleanup local symbol table only	Causes the local symbol table to be freed
Factor	■ The RAD Editor uses this panel to compile an application. Do not use it in any applications.	
See also	N/A	

configure

	Description
Definition	The **configure** command panel sets up the terminal running the RAD Application.
Parameter	N/A
Factor	■ The login application uses this panel to ensure that the terminal you login on to has the correct terminal definition.
See also	**user.login**

event.send

	Description	
Definition	The **event.send** command panel sends a system event to a thread.	
Parameter	Field	Definition
	Thread ID	ID of thread to which you are sending the event.
	Event Name	Unique name of the event.
	Names	Name (array of characters) of each parameter in the command panel or application to be called. The name is the field name in the application file.
	Values	Value (array of characters) of each parameter in the command panel or application to be called.
Factor	■ Use **event.send** for internal ServiceCenter communication to pass information to another thread, most commonly with the parent thread. ■ On the receiving thread, this action can either: ● Trigger a system event ● Displayevent ■ For the Thread ID, -1 is the parent thread.	
See also	**thread.start**	

fmt

	Description	
Definition	The **fmt** (Format Definition) command panel invokes a screen painter to create a form.	
Parameter	Field	Definition
	format name	Name (character) of the format you want to define.
	abort (exit)	Name (label) of the exit to take if the **format definition** is aborted.
Factor	■ This panel only works for text mode applications.	
See also	Go to the **Concepts** > **Tailoring** topics in *Administering ServiceCenter* online help for more information.	

priority

	Description	
Definition	Use the **priority** (Task Priority) command panel to change the Operating System processing priority of the current task.	
Parameter	Field	Definition
	relative task priority number	Positive or negative number indicating relative change in the processing priority.
Factor	■ A positive number raises the priority and a negative number lowers the priority. ■ On some operating systems, you may need special authority to change the priority.	
See also	N/A	

return

	Description
Definition	The return command panel returns an application to the prior state of the last rio panel.
Parameter	N/A
Factor	■ You must use a **display** panel must when the **on form modified** RAD Event is selected. If another **rio** panel is called, the `$file` variable does not contain the user's typing, and all updates are lost. ■ The application continues normally when the user saves the updated record.
See also	N/A

signal

	Description	
Definition	The **signal** (Signal Process) command panel creates a software interrupt which usually terminates an attached task.	
Parameter	Field	Definition
	name/pid of resource to signal	Name (character) of the task to signal. It is the attach resource name used to attach the task.
Factor	■ The software interrupt terminates the attached task unless it is caught and handled by the task. All RAD attached users terminate cleanly upon receipt of this software interrupt.	
See also	**attach**	

sleep

	Description	
Definition	The **sleep** (Sleep for Interval) command panel makes an application pause or sleep for a designated number of seconds before continuing.	
Parameter	Field	Definition
	sleep interval in seconds	Number of seconds you want the application to pause.
Factor	■ The application continues from the interruption point after the sleep interval has passed.	
See also	N/A	

user.login

	Description	
Definition	The **user.login** (User Login) panel logs a user onto the system.	
Parameter	Field	Definition
	UserID	User name (character) of the person logging on to the system.
	Password	Password (character) of the person logging on to the system.
	New Password	New password (character) for the person logging on to the system.
	Errcode	Variable (number) to contain any error code returned by the login process. Error codes may be returned by external security applications, such as RACF or SAF.
Factor	■ You must call the Login application to access the system. ■ The User ID and Password fields are defined in ServiceCenter's operator table.	
See also	N/A	

Database—Advanced Panels

The advanced database grouping has six command panels: **fcreate**, **fremove**, **freset**, **detect.keyed**, **fregen**, and **project**.

fcreate

	Description	
Definition	The **fcreate** (File Create) command panel creates a database file.	
Parameter	Field	Definition
	descriptor record	Database Dictionary record variable (file variable) from which a file is created.
	data pool number	Defines the pool where data records for the new database should be stored. Valid pool numbers depend on your configuration. Typically, **3** refers to SCDB.DB1, **4** to SCDB.DB2, and so on.
	index pool number	Defines the pool where index records for the new database should be stored. Valid pool numbers depend on your configuration. Typically, **3** refers to SCDB.DB1, **4** to SCDB.DB2, and so on.
Factor	■ Normally, use the File Create function in Forms Designer and the Database Dictionary to accomplish this. ■ Remove files with the **fremove** command panel. ■ Set the **root.record** field in the descriptor record to 0 (zero) when creating the dbdict in P4, or set to -1 (negative one) when creating the dbdict in SQL.	
See also	**fregen**, **fremove**, **freset** Go to **Concepts** > **Tailoring** in *Administering ServiceCenter* online help for further information.	

fremove

	Description	
Definition	The **fremove** (File Remove) command panel removes the specified file.	
Parameter	Field	Definition
	file to remove	Name (character) of the file to remove from the database.
Factor	■ The **fremove** command panel does not prompt for confirmation before deleting the file. The file is reset before it is removed.	
See also	**freset, fcreate**	

freset

	Description	
Definition	The **freset** (File Reset) command panel erases all records from a database file, but does not remove the file in the database so you can use it again.	
Parameter	Field	Definition
	file name	Name (character) of the file you want to reset.
Factor	■ The **freset** command panel does not ask for any confirmation before resetting the file.	
See also	**fremove**	

detect.keyed

	Description	
Definition	The **detect.keyed** command panel checks if the query passed is fully keyed.	
Parameter	Field	Definition
	file	File variable from which to select the records. The file variable will have been initialized with the **rinit** command panel.
	query	SQL query (character) that specifies the criteria for selecting records from the source file.
	sort fields	Names of the fields (array of characters) used to sort the selected records (optional). If these sort fields are filled in, the query uses this as the key, which may adversely affect response time.
	exit if partial or non-keyed query	Exit to take if the query is not fully keyed. It normally exits to a message and a confirmation screen that allows the query to continue.

	Description
Factor	■ A true or **true** query selects all records; a false or **false** query selects none. ■ You can sort on one or more fields. ■ You can sort efficiently only on keys. ■ To use a pre-defined key of multiple fields, you must enter the same field names in the same order as entered in the dbdict. ■ The query field is parsed at run time. Since parameter (local) variables cannot be parsed at run time, query expressions with parameter variables must be input in a special manner. The procedure is to enter an expression that is parsed at run time which generates a string with no local variables. Therefore, to check for the field name being equal to the local variable `$menu.name`, enter “`name=\"“+$menu.name+”\“` ”. If the local variable `$menu.name` contains the value MAIN MENU at run time, this results in the query: `name=“MAIN MENU”`. ■ If you specifically sort on a key that allows NULL entries, you do not retrieve any records whose key is NULL. These records are not in the index. ■ Use caution when defining the query, since any NULL values appended to or inserted in a NULL query, result in a NULL query, which is treated as **true**. ■ Invalid queries take the **error** exit. ■ Use the RAD application **bu.detect.keyed** in place of the **detect.keyed** command panel.
See also	**select**, **project**, **join**

fregen

	Description	
Definition	The **fregen** (File Regenerate) command panel regenerates all indexes for the specified file.	
Parameter	Field	Definition
	file name	Name (character) of the file whose indexes (keys) you want to regenerate.
	IR flag	Condition (boolean) that, if true, will cause only the IR index to be regenerated.
Factor	■ The File Regenerate command panel causes the system to regenerate all the indexes in a file. This is required only after you make a change in the file descriptor that affects the file's keys. ■ During any regeneration, the entire table is locked. ■ Index regeneration should never be abnormally terminated.	
See also	**freset**	

project

	Description
Definition	The **project** (Project) command panel copies specified fields from a source record to a target record. The fields must have the same names in the source and target records. Other fields in the target file are not altered.

	Description	
Parameter	Field	Definition
	source file variable	Name of the record variable (file variable) from which you want to copy the fields.
	target variable	Name of the record (file variable) to which you want to copy the fields.
	Project Using Labels?	A condition (boolean) that, if true, fields will be fully qualified for the project, matching name, level and data type.
	Condition for NULL	A condition (boolean) that regulates the behavior of project with respect to NULL fields. true—do not project NULL fields from source file. false—do not project from source unless target is NULL. NULL/unknown—project fields as specified above.
Factor	N/A	
See also	select	

External communication panels

The external communication grouping has five command panels: **connect**, **dde**, **read**, **write**, and **disconnect**.

connect

	Description	
Definition	The **connect** command panel establishes a communication session with an external device (for example, printer), file, or program	
Parameter	Field	Definition
	file variable	Connection variable (file variable) assigned to the communication session.
	device address	Device address (character) for the device, file or program (for example, qsam, popen, jes).
	type	Session **type** (character). There are five supported types: ■ printer ■ jes ■ qsam ■ file ■ popen
	options	Configuration **options** (array of characters), if any, needed for communication.
	exits	This array is not used on this panel.
Factor	■ The **options** and **exits** that are available depend upon the connect type. ■ For type File, Option 1 is the file type (text, raw, binary) and Option 2 is the connection type (append, read, write). Only the first character of the option is checked (for example, t for text, r for raw) and either upper or lower case is accepted. No exits are honored except normal and error. ■ From the time a connection is made, until it is disconnected, all error messages should lead to a disconnection.	
See also	**read**, **write**, **disconnect**	

dde

	Description	
Definition	The **dde** command panel calls any of the supported DDE functions. Use this panel to initiate any of the seven different actions associated with a DDE client conversation (see the following table of parameters).	
Parameter	Field	Definition
	DDE Action	Any DDE action you want to take. For example: ■ initiate—start a session. ■ terminate—end the session. ■ poke—set the value of the named item. ■ request—get the value of a named item and return it as a string. ■ execute—ask ServiceCenter to execute a transaction or set the focus to a named item. ■ advise—tell an external application that you contact to notify RAD when a named item is changed. ■ unadvise—tell an external application not to notify you of changes to a named item.
	Return Value	Value returned from the DDE call, such as $L.channel.
	Input Values	Additional parameters sent with the DDE action.
Factor	N/A	
See also	N/A	

read

	Description	
Definition	The **read** command panel reads a record from a communication session.	
Parameter	Field	Definition
	file variable	Connection variable (file variable) associated with the communication session
	record	Variable (character) in which to store the results
	options	Connecting type specific options (array of characters)
	exits	Connecting type specific exits (array of labels).

Connection Type	Exit Type
qsam	■ End of file ■ SYNAD error
file	End of file
popen	End of file

	Description
Factor	■ This panel does not read RAD internal files. ■ Before you can use the **read** panel in an application, you must use the **connect** command panel to establish a communication session. ■ For **file** type, read in binary mode: **option 1**=bytes to read. **exit 1**=end of file. ■ In text mode: **option 1**=bytes to read. **exit 1**=end of file. ■ In raw mode: **option 1**=separator character(s) if string, or bytes to read if number. **exit 1**=end of file. ■ When the read operations complete, use the disconnect command panel to terminate the communication session.
See also	**control**, **connect**, **disconnect**, **write**

write

	Description	
Definition	The **write** (Write External Record) command panel writes a record to a communication session.	
Parameter	Field	Definition
	file	Connection variable (file variable) where the panel writes.
	record	Record (character) to send to the communication session.
	options	Options (arrays of characters) for write.
	exits	Not used in this panel.
Factor	■ You cannot write to RAD internal files. ■ Before you can use the **write** command panel, you must use the **connect** command panel to establish a communication session. ■ In raw mode: **option 1**—separator characters of string or bytes to write ■ The following options apply in raw mode only: NULL—writes data without a separator <*number*>—write exactly this number of bytes without a separator STRING—writes data with the STRING as the separator ■ When the write operations complete, use the **disconnect** command panel to terminate the communication session.	
See also	**read**, **control**, **connect**, **disconnect**	

disconnect

	Description	
Definition	The **disconnect** command panel terminates a communication session.	
Parameter	Field	Definition
	file variable	Connection variable.
	options	Not used on this panel.
	exits	Not used on this panel.
Factor	N/A	
See also	**connect**, **read**, **write**	

In-line SQL RAD command panels

In-line SQL RAD command panels provide further interfaces to the RDBMS. Use these command panels in RAD applications to implement SQL commands to execute SQL statements and stored procedures, select and fetch rows, and get errors returned by the RDBMS. You must have knowledge of the SQL language and RDBMS conventions to use these command panels.

The in-line SQL RAD grouping has eight command panels: **sqlcrt**, **sqlddl**, **SQLexecute**, **SQLfetch**, **SQLgeterr**, **SQLprocedure**, **SQLselect**, and **sqlunl**.

sqlcrt

	Description	
Definition	Use the **sqlcrt** (SQL Advanced Conversion-Create DDL) command panel to convert a P4 file to a DDL file.	
Parameter	Field	Definition
	File Variable	Name (Character) of the P4 file to convert.
	File Name	Name (Character) of file where you write DDL statements.
	File Path	Directory (Character) where the file containing the DDL statements are created.
Factor	■ Use this panel only in **SQL.ADVCONV.CREATE.**	
See also	SQL Advanced Conversion: Generate DDL Statements	

sqlddl

	Description	
Definition	The **sqlddl** command panel displays the DDL (Data Definition Language).	
Parameter	Field	Definition
	file variable	Identifies the file for which you want to generate a DDL.
	table ddl	Name of the array to which the create statement is returned.
	index ddl	Name of the array to which the create index statement is returned.
Factor	N/A	
See also	N/A	

SQLexecute

	Description
Definition	The **SQLexecute** command panel executes SQL commands in the RDBMS.

Parameter

Field	Definition
SQL statement	SQL commands to execute on the RDBMS.
Commit after execute	Set to **true** to end your transaction and make all changes performed in the transaction permanent.
Rollback after execute	Set to **true** to undo work done in the current transaction if an error occurs.
SQL generic error code	If the application returns an error exit, the generic error can be examined to determine the error that occurred. The generic error codes return the following values:

ERROR	–1
NOTFOUND	100
DUPLICATE	–1000
MANDATORY	–1001
LOCKERR	–1002
MAXIO	–1003

Factor	N/A
See also	N/A

SQLfetch

	Description	
Definition	Use the **SQLfetch** command panel to fetch the next row that a query from the RDBMS table retrieves. You can only fetch the next row, not the previous row, from the RDBMS table.	
Parameter	Field	Definition

	Table variable	This variable contains the row of data that the RDBMS table retrieves.
		To retrieve the current row of output into a variable:
		`$row = contents ($tablevar)`
		To retrieve a specific column out of a row, `$number` represents the column number in the row.
		`$a = $number in $row`
		The following retrieves the first column out of the row:
		`$a = 1 in $row`
		or
		`$a = 1 in $tabvar`

Factor	N/A	
See also	N/A	

SQLgeterr

	Description	
Definition	Use the **SQLgeterr** command panel to retrieve error codes and messages from the RDBMS.	
Parameter	Field	Definition
	Error code	Error code that the RDBMS returns.
	Error message	Error messages that the RDBMS returns.
Factor	N/A	
See also	N/A	

SQLprocedure

	Description	
Definition	Use the **SQLprocedure** command panel to execute a stored procedure on the RDBMS.	
Parameter	Field	Definition
	Procedure name	Name of the stored procedure to execute.
	Parameter names	Names of the arguments to a stored procedure. Supply the value of each parameter name when the procedure executes.
	Parameter values	Values of the arguments to a stored procedure.
	Return status	Stored procedures can return a value indicating the procedure completed successfully or indicating the reason the procedure failed. See the appropriate RDBMS SQL reference for return status values.
	Return parameter flags	For Sybase, the remote procedure call must indicate that you use the parameter as a return parameter.
Factor	N/A	
See also	N/A	

SQLselect

	Description	
Definition	Use the **SQLselect** command panel to select data from a table on the RDBMS.	
Parameter	Field	Definition
	Table variable	Contains information that represents the current SELECT statement and the current row of data that the RDBMS table retrieves.
	Select Statement	Select statement that retrieves data from the RDBMS table.
	Exit if no records selected by query	Exit the panel takes if the query returns no records.
Factor	N/A	
See also	N/A	

sqlunl

	Description	
Definition	The **sqlunl** (SQL Advanced Conversion-Unload Data) command panel unloads a P4 file.	
Parameter	Field	Definition
	P4 File Variable	ServiceCenter filename to unload (character).
	SQL File Variable	"SQLTEMP" + <*filename*> copy of P4 file (character).
	File Path	Directory where files containing unloaded data are created (character).
	Error Threshold	Number of errors allowed before cancelling unload (number).
	Total Records Processes	Number of records processed by the application (number).
	# of Records with Errors	Number of records with errors processed by the application (number).
	Volser	**Note:** No longer supported.
Factor	■ Used only in Application: **SQL.Advconv.unload**.	
See also	SQL Advanced Conversion: Unload Data for Bulk Load	

Obsolete command panels

The following command panels are obsolete:

- external
- fload
- funload
- join
- pgmcall
- pgmdelete
- pgmload
- projecte
- qbe (replaced by genquery)
- safuser

4 Display Panel Conversions

CHAPTER

You can convert a RAD application to use the **display** application. Topics in this section include:

- *Using the display application*—overview of the display application.
- *Display command panel*—description of the display command panel, its fields, and typical variables.
- *Display utilities*—description of application that adds, updates, or deletes displayoption and displayevent records; and the application that converts RAD applications to use the display application.
- *Converting an application*—procedures for converting an application.
- *Checking your conversion for accuracy*—procedures for testing your conversion for accuracy.

Using the display application

The **display** application replaces **rio** and **fdisp** RAD panels in standard, ServiceCenter applications. You run the **display.cv** utility to convert an application to issue calls to **display** rather than to **rio** or **fdisp** panels.

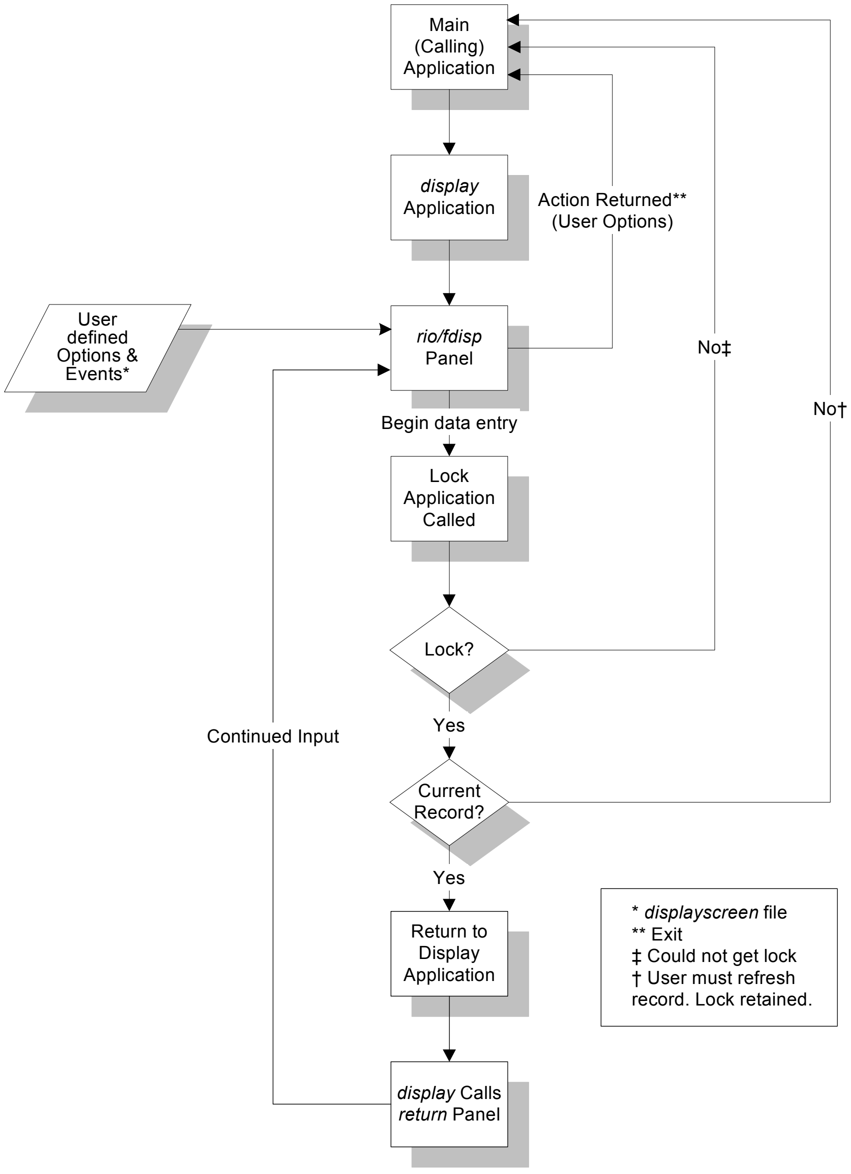
Note: Standard out-of-box ServiceCenter applications can call the **display** application without using the conversion utility. If your system contains custom applications that use **rio** or **fdisp** panels, you can convert these to take advantage of **display's** control features.

Data files within **display** allow you to create options and events that become available to the calling application at run-time. Use the Window controls to customize various window display features. All options and screen events within **display** are table-driven and not embedded in the RAD code. Consequently, you have access to these features and can edit them freely.

The **display** application uses a passed variable to cache all details about the screen and the options. This feature allows the system to redraw the screen more efficiently, without having to read the data from the database.

When the **display** application is called, the calling RAD application passes an application name and context to **display** and receives a token indicating which function to perform. For example, click **Find** to return a token called `find`. The **display** application looks up the option you selected in the `displayoptions` database and returns the token to the RAD application. **Display** can return a token, call another application (that is, start a new thread) or execute simple instructions.

The following chart illustrates the actions of the **display** application.


Main (Calling) Application
display Application
Action Returned** (User Options)
User defined Options & Events*
rio/fdisp Panel
No‡
No†
Begin data entry
Lock Application Called
Lock?
Yes
Continued Input
Current Record?
Yes
Return to Display Application
display Calls *return* Panel
* *displayscreen* file
** Exit
‡ Could not get lock
† User must refresh record. Lock retained.

Display command panel

In all ServiceCenter applications in the standard system, the **display** command panel replaces the **rio** and **fdisp** panels.

Fields

Field	Description
Application	Name of the calling application.
Label	Application panel name.
Comments	Reference comment about the panel. This is the only place where this exists.
Messages Array	Message to be returned to the calling application when the **display** panel exits. The message can provide users with information or tell the calling application to proceed with an operation.
QBE?	Logical field determining which panel (**rio** or **fdisp**) is being replaced. If the field is *true*, the **fdisp** panel is replaced. If the field is *false*, the **rio** panel is replaced.
Screen ID	Value from the displayscreen file form that is a unique name, identifying the record whose display characteristics are being read. **Note:** The Link record for the application allows this field to call the database to display the displayscreen file. This allows an integrated environment for modifying display behavior from within RAD.
File	File variable converted to $L.file within the **display** application.
Array of local variables	Each element of this array is mapped to the variable name of the same index in the **Array to reference variables.**
Array to reference variables	Each element of this array is a character string that becomes a variable containing the contents of the corresponding index in the **Array of local variables.** This is required when an event, option, or screen requires a reference to a local variable.
Return Action	String token indicating which function the application is to perform (for example, **Fill**).

Field	Description
Return Option	Option number that tells the application which option the user selected. This is NOT RECOMMENDED. When events (rather than options) are implemented that require the application to perform an action, there is no valid option; it is more meaningful to have the RAD code branch on **Fill** rather than on Option 8.
Text Bank	The bank of options to display in Text mode. The **display** application sets this value. Save it if the calling application calls **display** again. **Note:** Always use a different variable name for each call to **display** within a single application. You can use a $G variable and change it for each frequently used call to **display**.
Exit: close/menu	Exit called when the user closes the main (calling) application. The return action is set to close and the option is set by the user.
Exit: back	Exit called when the user selects the **Back** option. The return action is set to back and the option is set by the user.
Screen Cache	Not used in this version

Useful variables

Variable	Description
$displaymaster	Global variable. If the value is NULL or pointing to the displaymaster file, this variable is initialized to the record in the displaymaster file where language = $lo.language, or ENG if $lo.language is NULL.
$lo.language	Global variable selecting the language for the user's session. Set this at login if you do not want ENG.
$L.filed	Local variable referenced by the options or the screen. Use this variable if any expression or condition needs to reference a field in the file variable passed to the **display** application.

Display utilities

The purpose of these routines is to help implement display in a system with custom or old RAD (display.cv) and to help maintain displayoptions (display.options.fc). The actual RAD routine that makes display work is named display.

display.options.fc

This utility is called from the Subroutines process of the **displayscreen** Format Control record. It adds, updates, or deletes displayoption and displayevent records associated with the screen.

Fields

Field	Description
Old file	Input value of file. This variable is $file0 in Format Control.
New file	Input value of second.file. This variable is $file in Format Control.
Add/Update	Input value of prompt. This variable controls how associated records are processed. Valid values are: ■ add ■ update ■ delete

display.cv

This utility converts existing **rio** and **fdisp** panels in an application to a displayscreen record and a displayoption record. It creates two new panels in the application with the name <*originalname*>.NEW and <*originalname*>.NEW.decide.

The second panel (<*originalname*>.NEW.decide) is a **decision** panel with the same exits as the **rio** and **fdisp** panels it replaces. The condition is $L.action=<action>, where <action> is the action added to the displayoption file.

There are no parameters on this panel.

Important: Always back up any application prior to conversion. You can then revert to the previous version if you prefer to use the original **rio** or **fdisp** panels.

Converting an application

If you have a custom application in RAD and want to convert it to use the **display** panel, you must run the **display.cv** conversion utility.

Important: Always back up any application prior to conversion.

To convert a custom RAD application to use the display panel

1. Click Command in the system administrator's home menu.
2. Type `*adisplay.cv` in the command line.
3. Press **Enter**.

   A dialog box prompts you for the name of the application to convert.
4. In the **Application** field, type the name of the application you want to convert.

   Warning: If you attempt this process using this application, copy and rename the application first so you can replace the original in case a problem arises.

5. Press **Enter.**

   A QBE list displays the **rio** or **fdisp** panels in the named application.
6. Double-click the panel you want to convert.

   The requested panel opens.

   Note: You can edit the panel. Changes occur in the `displayscreen` and `displayoption` records after conversion.

7. Click **Convert**.
8. Type the name of the **displayoption Screen ID** (name of the application you are converting).
9. Click **Convert**.
   - If the Screen ID already exists, you receive the following message in the status bar: `Duplicate screen already exists. Try again.`
   - If the conversion is successful, you receive the following message in the status bar: `Convert done!`

10. Click the Message button to display a synopsis of the entire process.

The displayscreen file is created first, followed by the **displayoptions**. The two new panels are added to the original application last (reading from bottom to top on the list).

Checking your conversion for accuracy

Once you convert your **rio** or **fdisp** panel to a **display** panel, check the following for accuracy:

- Correct field values in the **display** panel.
- Correct link between the **display** panel and the appropriate displayscreen record.
- Correct field values in the **displayscreen** record.
- All necessary variables passed to **display** application.

Field validity

To validate fields in the display panel

1. Open the Application Development Encyclopedia of the converted application.
2. Click **Edit**.

   The parameter panel opens.
3. Click **Goto**.
4. In the Input field, type display.
5. Click **Panel Type**.

   The requested display panel opens.
6. Verify that the following fields in the **display** panel contain valid values:

Field	Description
qbe?	Field evaluates to true for an **fdisp** panel or false for a **rio** panel.
Screen ID	Name of the displayscreen Screen ID you created during the conversion process. To view the record, put the cursor in this field and click Find.
File	Field must contain the correct file variable.
Array of local variables	Enter any local variables required for screen processing.

Field	Description
Array to reference variables	Enter the names of desired variables into this array.
Return Action, Return Option (not required), Text Bank	Ensure variables for these fields are correct.
Screen Cache	Not used in this version.

7 Verify all exits.

Links

In order for **display** to function properly within a RAD application, a valid link must exist between the **display** panel and the **displayscreen** record defining the actions for that panel.

To check for a valid link

1 Open the Application Development Encyclopedia of the converted application.

2 Click **Edit**.

The **parameter** panel opens.

3 Click **Goto**.

4 In the input field, type `display`.

5 Click **Panel Type**.

The new display panel opens.

6 Place the cursor in the **Screen ID** field.

7 Click **Find**.

The **displayscreen** record of the selected Screen ID opens.

8 Select the **Options** tab to check your options for accuracy.

Error check

To check the link file record if the link test fails

1 From the System Navigator, click **Utilities** > **Tools** > **Links**.

A blank link record opens.

2 Type `application` in the **Name** field.

3 Click **Search**.

The application link record opens.

4 Verify that a line exists in the **application** link record linking the **text** field with the displayscreen file and the **Screen ID** field.

   Note: You can add a second link connecting the **text** field with the displayscreen file. Scroll the link record to the right and add screen.id#".new" to the **Add Query** field. This allows you to create a new displayscreen record.

5 If no displayscreen record is found (and the **application** link record is correct), create a new **displayscreen** record using the same **Screen ID** name.

   a Modify the new displayscreen record to ensure that field values are correct.

   b If the record used the **display.cv** conversion utility, verify that references to local variables are set correctly when **display** is called.

Verifying fields

Check the information in the **displayscreen** form for accuracy.

Field	Description
On option 0	Defines the **Action** to take. The choices are: do nothing, redraw screen, and return to appl.
Format	The correct form name or variable. This can be blank for **fdisp** panels.
I/O (If RIO)	If null, the screen does not display.
Options (Options tab)	If the records are converted, remove any options that the **display** application handles. ■ Option #3: Back ■ Option #6: More (Text mode) ■ Option #12: Close application (Text mode) ■ Option #999: Close application (GUI) **Note:** You can remove these options before the conversion process by blanking out the option number in the confirmation screen.

Field	Description
Local Variables (Options tab)	Verify that any local variable references in all options or events are properly expressed in the call for the **display** application.
Option Labels (Options tab)	Verify that the conversion routine matched the Options with the correct **Actions** and **Conditions**.

Passing local variables

If the scope of a variable does not allow it to be seen by a sub-application call, and it is required for conditions or expressions at run time, the variables must be passed to the **display** application.

The parameter panel of the **display** application has two fields that can do this:

- Array of local variables
- Array to reference variables

The first array has, as each element in the array, the actual local variable. The second array has as each element in the array, a string that becomes the new variable at run-time. Peregrine recommends that you prefix the new variables with `$L.` (making them *local* variables) to avoid unexpected side effects.

Examine the values in the following panel:

Note: `$L.environment` and `$FILE0` are local variables and not visible to the sub-application.

The **Array to reference variables** field is set to `{"$L.environment", "$L.file0"}`. The **display** application sets the value of the `$L.environment` variable to the passed `$L.environment`, and the `$L.file0` variable to the passed `$FILE0`.

For additional information on the use of variables in ServiceCenter, go the the **Reference** > **System language** topics in *Administering ServiceCenter* online help.

Note: Peregrine recommends that you reference as few global variables as possible. Global variables can cause side effects, and can prevent applications from being *re-entrant* (called more than once).

APPENDIX A

Command Panel List

The following table is an alphabetical listing of the RAD command panels.

attach	mb.ok	sqlddl
call	mb.yes.no	SQLexecute
compile	msg	SQLfetch
configure	next	SQLgeterr
connect	previous	SQLprocedure
count	print	SQLselect
dde	priority	sqlunl
decision	process	thread.start
detect.keyed	project	unlock
disconnect	radd	user.login
event.send	rdelete	wclose
fcreate	read	wopen
fdisp	return	write
fmt	rinit	wselect
fregen	rio	
fremove	rupdate	
freset	select	
genquery	signal	
lock	sleep	
loop	sqlcrt	

Index

Peregrine
SYSTEMS®

June 18, 2004