JAM

JAM 7.01 Update

December 1995

This software manual is documentation for JAM® 7.01. It is as accurate as possible at this time; however, both this manual and JAM itself are subject to revision.

Send suggestions and comments regarding this document to:
Technical Publications Manager
JYACC, Inc.
116 John Street
New York, NY 10038
(212) 267–7722

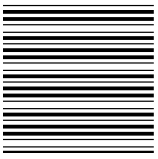


Table of Contents

About this Guide . . . . . **vii**

Chapter 1 What's New in JAM 7.01 . . . . . **1**

Graph Widget Enhancements 1
Screen Wizard Enhancements 2
Widget Type Preservation 2
Transitive Links 3
Selection Screens 4
Utility for Runtime Libraries 4
SQL Generation 4
Property API Enhancements 4
Configuration Changes 5
@NULL Variable 5
Library Functions 6
sm_jfilebox 6
sm_optmnu_id 6
sm_set_help 6
Database Driver Enhancement — SYBASE 7
Online Documentation 7
PostScript Files 7
Search Panels 9

Enhanced View Capability 9
Enhanced Print Capability 9
Icons for New and Changed Features 10

Chapter 2 Documentation Addendum 11

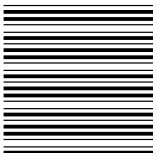
Getting Started 11
Editors Guide 12
Fonts for Graphs 12
Frequency Subproperty 12
Synchronized Scrolling Arrays 12
Application Development Guide 13
Changing Widget Types at Runtime 13
Changing Widget Properties 13
LDB Functionality 14
Transaction Manager Commands 14
Application Traversal Properties 14
Accessing Transaction Information 15
Error Processing in Database Applications 17
Stub Functions 18
Configuration Guide 18
SMLPRINT and the LP Logical Key 18
Setting Number of Decimal Places 18
Logical Keys 19
Use or Discard Error Acknowledgment Key 19
Windows-Specific Information 20
Database Driver – ODBC 21
Formatting for Colon Plus Processing and Binding 21
Autocommit 21
Database Driver – ORACLE 21
Formatting for Colon Plus Processing and Binding 21
Declaring Cursors 22
Using Stored Subprograms 22
Language Reference 22
Comments in JPL 23
Property API Supplement 23
Search Path 24
dm_getdbitext 24
sm_calc 25
sm_filebox 25

sm_fio_open 25
sm_message_box 26
sm_mncrinit 26
sm_ms_inquire 26
sm_prop_get/sm_prop_set 26
sm_slib_install 26
sm_strip_amt_ptr 27

Chapter 3 Using JAM for Macintosh 29
Setting JAM for Macintosh Defaults 30
Preferences File 30
Setting SMBASE 34
Setting Properties to Resource Names 35
Editing PICT Resources 35
Editing an Executable’s Resources 36
Editing the Resource File 37
Adding PICT Resources 37
The Keyboard Interface 38
Using the Screen Editor 39
Using the Menu Bar Editor 40
Using JPL 41
Setting JAM Properties at Runtime 41
Using JAM Library Functions 42
Using JAM Utilities 43
Using the Stealth Utility 43
Using GraphicConverter 44
Using Apple Events 45
Runtime/Editor Events 45
Utility Events 46

Appendix A Library Functions Addendum 49

Index 61



About this Guide

This document includes:

- Descriptions of what's new in JAM 7.01 including online documentation enhancements.
- Addendum to JAM 7.0 documentation.
- Information about JAM for the Macintosh.
- An appendix of new and changed JAM library functions.

Conventions

The following typographical and terminological conventions are used in this guide:

Text Conventions

`expression` Monospace (fixed-spaced) text is used to indicate:

- Code examples.
- Words you're instructed to type exactly as indicated.

	❍ Filenames, directories, library functions, and utilities. ❍ Error and status messages.
`KEYWORDS`	Uppercase, fixed-space font is used to indicate: ❍ SQL keywords. ❍ Mnemonics or constants as they appear in JAM include files.
numeric_value	Italicized helvetica is used to indicate placeholders for information you supply.
[option_list]	Items inside square brackets are optional.
`{x | y}`	One of the items listed inside curly brackets needs to be selected.
x ...	Ellipses indicate that you can specify one or more items, or that an element can be repeated.
new terms	Italicized text is used: ❍ To indicate defined terms when used for the first time in the guide. ❍ Occasionally for emphasis.

Keyboard Conventions

XMIT	JAM logical keys are indicated with uppercase characters.
Alt+A	Physical keys are indicated with initial capitalization, and keys that you press simultaneously are connected with a plus sign.

JAM Documentation

The JAM documentation set includes the following guides and reference material:

Read Me First — Consists of three sections and is available online and printed:

- *What's New in JAM* — Briefly describes what's new in JAM 7.
- *Installation Guide* — Describes how to install JAM on your specific platform and environment.
- *License Manager Installation* — Instructions for installing the License Manager (used on many UNIX and VMS platforms).

Getting Started — Contains useful information for orienting you to JAM. Includes a description of the JAM environment and features, how JAM addresses real-world application development issues, and a guided tutorial for building a mini-JAM database application. (online)

Editors Guide — Instructions about using the JAM authoring environment; learn how to use the graphical tools for creating, editing, and designing your application interface. Includes detailed descriptions of the screen editor, screen wizard, menu bar editor, and styles editor. The *Editors Guide* is also provided online on GUI platforms. It is installed with the installation of the JAM software and can be accessed by selecting help from within the screen editor. (online)

Application Development Guide — Information by topic to guide you in developing your JAM application. This includes components of the JAM development environment such as the repository, hook functions, and menu bars, as well as sections on the SQL executor, SQL generator and the transaction manager. (online)

Language Reference — Describes JPL, JAM's proprietary programming language. Also includes reference sections for JPL commands, built-in functions and JAM library functions. The man pages in the reference sections are arranged alphabetically. (online)

Database Guide — Instructions for using JDB, JYACC's prototyping database, and for the commands and variables available in the database interfaces. Includes an Database Drivers section containing instructions unique to each database driver. (online)

Configuration Guide — Instructions for configuring JAM on various platforms and to your preferences. Some options that can be set relate to messages, colors, keys and input/output. Also includes information on GUI resource and initialization files. (online)

Master Index and *Glossary* — Provides an index into the entire documentation set and a dictionary of terms used in the documentation set. This is in addition to the indexes in the individual volumes. (online postscript file).

Upgrade Guide — Online only. Information for upgrading from JAM 5.

Online Documentation

JAM's documentation set is available online and included with the JAM distribution. Postscript files are also included in the distribution. The books can be viewed through the DynaText™ browser on GUI platforms. It can be accessed by choosing Help from within JAM or by running DynaText's read-only browser from the command line or by clicking on the DynaText icon. For instructions on using DynaText, request Help while you have a browser window open.

Collateral Documentation

The following information is also provided with your JAM installation:

- Database Driver Notes — JAM 7 has database drivers for most popular relational database engines, as well as JDB, JAM's proprietary database.

Information for JDB, Sybase, Oracle, Informix and ODBC are located in the *Database Guide*; others are included separately.

- Online help — The *Editors Guide* is provided in online form through the DynaText browser on GUI platforms. It can be accessed by choosing Help from the screen editor. For instructions on using DynaText, request Help while you have a browser window open.
- Online README file.

Additional Help

JYACC provides the following product support services; contact JYACC for more information.

- Technical Support
- Consulting Services
- Educational Services

What's New in JAM 7.01

JAM 7.01 includes the following features:

- Business graph enhancements.
- Screen wizard enhancements.
- `fastlib`, a library deployment utility.
- Changes to SQL generation.
- Application-wide properties.
- Configuration variables for octal support.
- New library functions.
- `@NULL` variable.
- SYBASE database driver enhancement.
- Online documentation enhancements.

Graph Widget Enhancements

The enhancements to business graphs include:

- Under Motif, graph size is no longer limited to a 7 x 10 aspect ratio.

- Extended colors allow you access to the full GUI colors.
- Data series colors now match the 16 JAM basic colors.
- Motif requires only one back-end process (gdsp) rather than two.
- Pie charts now support unlimited data segments. JAM 7.0 supported 12 segments, with segments after the twelfth having the same attributes as the twelfth.

  After the twelfth segment, colors repeat; however, duplicate colors near the start/end juncture are avoided.
- XY plots now support twelve (12) data sets (previously supported six) like the other chart types.
- LDB entries can be used as data sources.

Screen Wizard Enhancements

The screen wizard has been enhanced to include the following features:

Widget Type Preservation

In JAM 7.0, the screen wizard converted all repository widgets, regardless of their type, to single line text widgets on its output screens. The wizard now preserves widget types for single-row layout to accommodate widgets that occupy more than one line. For example, if you have a database-derived field in a repository entry that is an option menu, the screen wizard retains the widget's type when it is used in a single-row layout.

For grid layout, if the field is not one of the grid-supported widgets (dynamic label, list box, or single line text), the screen wizard converts the widget to a single line text widget.

When a widget is converted, it retains as many relevant properties as possible. However, the values for the following properties might not be preserved or set:

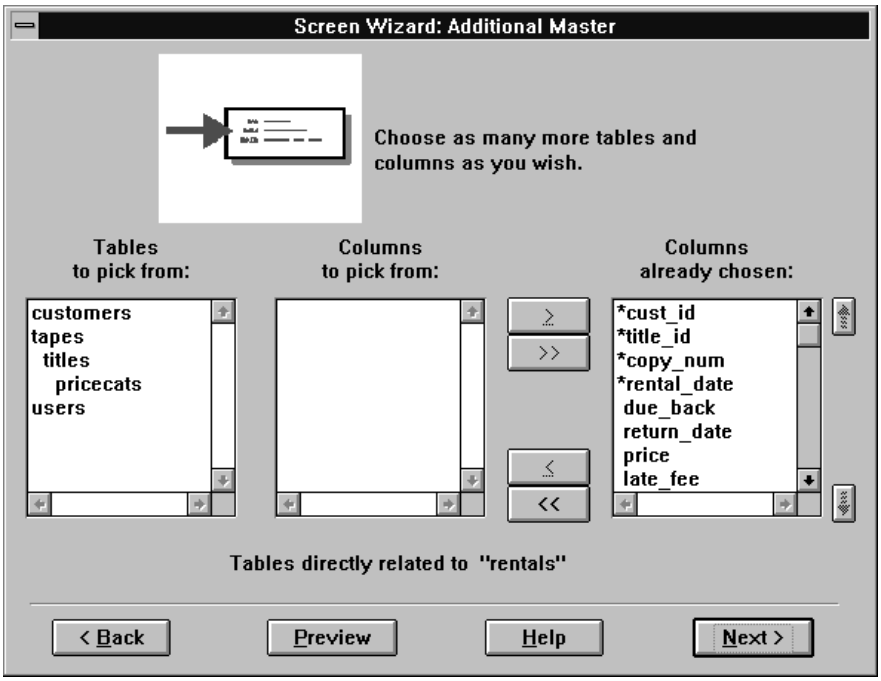
- Max Data Length (under Geometry).
- Select on Entry (under Input).
- Null Field (under Format/Display).

Transitive Links

In JAM 7.0, when you request Additional Tables, the screen wizard displays only those tables that are directly related to, that is, immediate children of the First Table. Therefore, you were not able to choose tables that are related to the First Table through a link to some other table (an ancestor).

In JAM 7.01, the screen wizard recognizes and uses these transitive links. When you choose to include additional tables in the Master, Detail, or Subdetail sections of a screen, the Additional Tables screen displays all tables that are related to the First Table, whether directly (an immediate child) or through any number of intermediary tables. You can choose any columns you wish from any of these related tables. The screen wizard makes sure that your final screen contains all the necessary table views and links.

For example, if the `rentals` table (from the `videobiz` database) is the First table, the Additional Tables dialog lists the tables from which you can choose, as indicated in the following illustration:



`customers`, `tapes`, and `users` are immediate child tables of `rentals`; `titles` is a child of the `tapes` table and `pricecats` is a child of `titles`. You can select columns from any of the tables listed. If you select `pricecats`, for example, as one of the additional tables to include on your screen, the screen wizard includes the columns you select, and automatically includes a table view for `pricecats`,

and all table views and links needed to connect `pricecats` to `tapes` (since `tapes` is the immediate child of `rentals`).

In addition, when you request that the screen wizard generate selection screens, it will include each indirectly related table on the screen as its ancestor. In this example, `pricecats` is included on the selection screen for `tapes`.

Selection Screens

All references to Item Selections screens, including the check box on the screen wizard's Style and Finish screen, now refer to Selection screens.

Utility for Runtime Libraries

JAM 7.01 is distributed with the new utility `fastlib` which converts your JAM 6 and JAM 7 screen libraries into read-only, index-table format. The result of the conversion is improved runtime access of your application's libraries, and therefore, enhanced performance. However, once converted, the contents of the libraries cannot be edited in the screen editor. Therefore, the utility can be used in the final step before deploying an application.

The format is:

`fastlib [-f] [-s]` *input_library output_library*

–f Overwrites an existing file having the same name.

–s Sets the SYSLIB flag of the *output_library*.

SQL Generation

In JAM 7.01, SQL generation for validation links and for primary key checking in JDB uses the temporary variable `@dmtmp` instead of `@null`.

Property API Enhancements

JAM's property API is extended to include the following properties:

Number of Occurrences

`PR_NUM_OCCURRENCES` is now available as a read-only property for grid frames and synchronized scrolling groups. This property was previously available for other widgets. The property returns the number of populated occurrences.

Maximum Number of Database Rows

The following application properties can help you to optimize database applications for specific platforms, networks, and some database engines:

max_fetches (PR_MAX_FETCHES)
Sets a limit on the number of database rows that JAM's SQL executor will return to arrays with an unlimited number of occurrences (Max Occurrences property is blank). The default value is 1000. You can increase or decrease this number; any integer value greater than zero is valid and will affect the number of rows returned to JAM variables by a subsequent SELECT or CONTINUE statement. You can also get and set this property with `sm_prop_get_int` and `sm_prop_set_int`, respectively.

max_rows_per_fetch (PR_MAX_ROWS_PER_FETCH)
Sets a limit on the number of rows that JAM's SQL executor will request when performing vendor-specific "array fetching." This does not change the number of rows fetched to your application, only the number of times JAM requests rows from the server. For example, if there are 1000 rows and `PR_MAX_ROWS_PER_FETCH` = 200, the SQL executor will fetch the result set in five batches of 200. Any integer greater than 0 and less than or equal to 1000 is valid and affects the batch size.

Array fetching is currently supported for Allbase and ORACLE. This property is primarily intended for ORACLE users. If your JAM application is fetching very large rows, you might need to lower the batch size.

Configuration Changes

The `OCTAL_SUPPORT` setup variable lets you define whether your application interprets numbers with leading zeros as octal or decimal (default). The variable takes either of the following values:

- `OCTAL_SUPPORT_ON` — Numbers with a leading zero are treated as octal unless they are hexadecimal (`0x`, `0X`) or binary (`0b`, `0B`).
- `OCTAL_SUPPORT_OFF` (default) — Numbers with a leading zero are treated as decimal unless they are hexadecimal (`0x`, `0X`) or binary (`0b`, `0B`).

@NULL Variable

Calls to installed C functions can supply the new variable `@NULL` for any parameter that accepts `NULL` as an argument. For example, the following call to `sm_mnitem_delete` supplies `@NULL` as the function's second argument; this specifies to use the most recently loaded menu script:

```
call sm_n_mnitem_delete \
     (MNL_SCREEN, @NULL, ”edit_menu”, ”select_item”)
```

Library Functions

There are three additional functions included with the release of JAM 7.01: `sm_jfilebox`, `sm_optmnu_id`, and `sm_set_help`.

sm_jfilebox

The function `sm_jfilebox` serves the same purpose as `sm_filebox`, but can be called from JPL. Its first parameter is the target for the returned filename. `sm_jfilebox` returns –1 if the parameter is not valid (but does not display an error message).

A sample use might be:

```
call sm_jfilebox (’field->jam_help_screen’, ’help’, \
’field*.jam’,’Pick the desired help screen’, FB_OPEN)
```

Or:

```
call sm_jfilebox (’newfilename’, ’c:\\jam’, ’*.jam’, \
’Enter the name of the new file’, FB_OPEN)
```

Refer to page 50 in these notes for details.

sm_optmnu_id

The function `sm_optmnu_id` is used to obtain the ID of an option menu that is being populated with data from an external source. Refer to page 56 of this update for details.

sm_set_help

The function `sm_set_help` allows your JAM application to switch to help mode (or context help). In help mode, mouse-clicking on any object in the application invokes the help that is associated with that object. This function is supported on the GUI platforms and in character mode. Refer to page 57 of this update for details.

Database Driver Enhancement — SYBASE

JAM for SYBASE has a new option available in `DBMS DECLARE CONNECTION` statements for connecting to the database. `CHARSET` specifies the name of the character set the client will use. You only need to specify this option when the client's character set is not compatible with the server and conversion needs to be performed. SYBASE determines the client's default character set by reading the `locales.dat` file.

Online Documentation

The following enhancements have been implemented for the current online documentation:

- The JAM 7 documentation set is available as PostScript files in the JAM 7.01 distribution.
- Search panels have been added to facilitate your search capabilities of the online JAM documentation.
- New view stylesheets were added.
- The online table of contents can be printed.
- Icons identify new and changed features of JAM 7.01 and Macintosh-specific features.

PostScript Files

The JAM 7.01 distribution (CD-ROM and tape distributions only) includes documentation directories that contain PostScript files of the entire JAM 7.0 documentation set and this document. The `ps` directory is located in the `docs`

directory and contains a directory for each book in JYACC's printed JAM 7.0 release:

Book directory	Description
`appdev`	*Application Development Guide*
`config`	*Configuration Guide*
`database`	*Database Guide* — includes information on database drivers for Informix, JDB, ODBC, ORACLE, SYBASE CT Library, and SYBASE DB Library (`chap14.ps` through `chap19.ps`)
`editors`	*Editors Guide*
`getstart`	*Getting Started* — includes the tutorial
`langref`	*Language Reference*
`mindex`	*Master Index* and *Glossary* — an index for the entire documentation set
`readme`	*Read Me First*
`update`	*JAM 7.01 Update*

Each book directory includes:

- Cover and copyright information (`cover.ps`).
- Table of contents and preface material (`toc.ps`), if applicable.
- Section pages, if applicable (`sec#ch`*X*`.ps`, where # is the section number and *X* specifies the chapter number that follows the section page).
- Module introductions for tutorial lessons only (`mod#l`*lesson#*, where # is the module number and *lesson#* indicates the lesson number that follows the module pages).
- Chapters (`chap#.ps`).
- Appendices, if applicable (`appendx`*letter*`.ps`).
- Index (`index.ps`), if applicable.

Note: *If you are installing on UNIX, the PostScript files are compressed. Use the* `uncompress` *utility to expand the files.*

You can print the entire documentation set or print single chapters for your own internal use. The tables of contents and indexes provide page specifications for the

contents of the directory, not the online documentation. If you have a PostScript viewer, such as `ghostview` under UNIX, you can display the content of the PostScript files.

Search Panels

New search panels are provided to help you find information in the online JAM documentation set. Once you invoke DynaText, the online viewer, and open the JAM book, you can select the desired search panel by choosing Search⇒Forms from the DynaText menu bar. When you select one of the new search panels, the Standard (default) search panel is replaced at the bottom of the DynaText window. The following search forms were added:

- Sample Function — To find a code example for a specific function, enter a JAM function name (full- or partial-string). (Available only in the JAM book.)
- Find Function — Allows you to search for a function by description. For example, if you enter the string `*cursor*` (including wildcards), the query searches for those JAM functions that have `cursor` either in the actual function name or in the function's short description. In the example, of the word cursor, the search will find `sm_delay_cursor` as well as `sm_at_cur` where the word cursor is not in the function's name, but is in the short description ("Displays a window at the cursor position"). (Available only in the JAM book.)
- Proximity search — Allows you to search for a word or phrase in relation (within a specified number of words) to another word or phrase.

Enhanced View Capability

Once DynaText (the online viewer) is invoked and the JAM documents are opened, you can choose to display or hide the graphics by choosing View⇒Main View and selecting the desired display option:

- fulltext (default) — Hides the graphics (screen captures, etc.) until you choose the camera icon to display individual pictures.
- inline — Displays all the graphics in full.

Enhanced Print Capability

You can print the online table of contents by choosing File⇒Print in DynaText. From the print dialog, select `toc` from the Print format option menu. This will

output the table of contents for the selected books or sections. The entire table of contents is 44 pages and lists items down to three levels. Be aware that this output provides a relative view of the documentation; there are no page specifications.

Icons for New and Changed Features

With the installation of JAM 7.01, hyperlinks and notes are automatically attached to those areas in the JAM 7.0 online documentation where additions, corrections, and features should be noted.

Link Icons

Hyperlink icons are found along the right margin of the JAM online documentation, and indicate those areas where JAM 7.01 additions should be noted, or where corrections to the documentation are addressed.

Double-click or choose Open on the icon to go to the new information.

To view a list of JAM 7.01 hyperlinks (owner/user `Jam701`) and select a specific topic, access the Annotations Manager by doing one of the following:

- Under Motif, choose Annotations⇒Annotations Manager.
- Under Windows and Macintosh, choose Book⇒Manage Annotations.

Note Icons

The `JamMac` notes (view a list via the Annotations Manager) installed with JAM 7.01 for Macintosh contain information that is specific to using the Macintosh to develop JAM applications (refer to page 29 for more information). Double-click on the apple icon to display the contents of a note.

Documentation Addendum

The following changes are included to clarify or replace existing sections of the JAM 7.0 documentation set. The information is organized by book title. These changes are also noted throughout the JAM 7.0 documentation with hyperlinks to the changed information.

Hyperlinks appear as an icon in the right margin. Double-click or choose Open on the icon to go to the related information.

Getting Started

The information that follows is specific to the JAM Tutorial in *Getting Started*. The lesson number and step number are provided.

Lesson 1, step 50 (online tutorial) — States that the Delete button clears the screen of data and JAM automatically saves the change to the database, thereby deleting the record. While the Delete button does clear the screen of data, you must choose the Save button after choosing Delete to save the change. The record is then deleted from the database.

Lesson 9, steps 5 and 6 — As soon as you choose the Sort Widgets property, The Sort Widgets dialog box opens. You *do not* choose the More button. In step 6, you

enter `last_name` in the Sort Widgets dialog box and choose OK to resume in the Properties window.

Editors Guide

Fonts for Graphs

To ensure portability of graphs across platforms, JAM provides a set of graph-specific fonts (prefixed with the word Graph). These fonts are displayed when you have a graph widget selected and expand the Font Name property drop-down list. Under Windows, these fonts are listed immediately after the system fonts.

Frequency Subproperty

The Frequency subproperty for a widget that displays a system date/time stated that the property took a value in tenths of a second. The property expects a value in seconds. Therefore, if you enter 10 as the value in the Frequency subproperty, the widget is updated with the operating system's time every 10 seconds.

Synchronized Scrolling Arrays

The information on when JAM automatically synchronizes scrolling arrays was incorrect and the properties associated with synchronized scrolling groups were omitted from the JAM 7.0 documentation.

JAM automatically synchronizes arrays only when they meet either of the following conditions:

- The widgets are grid members within a grid frame.
- They are database-derived widgets that belong to the same table view or to different table views that are joined by a server link.

You can manually synchronize arrays that do not meet the above conditions by making widgets members of a synchronized scrolling group.

Once a synchronized group is created, you can set properties that specifically control the group as a whole.

To change the size and/or behavior of a synchronized scrolling group:

1. Select the group by doing either of the following:

- Select one of the members of the group. Then choose Edit⇒Group⇒Select Groups. If the widget belongs to more than one group, the Select Group dialog box opens. Select the Sync Group from the list.
- Select the Sync Group from the Widget list.

The Properties window displays Sync Group-specific properties.

2. Under Geometry, set any of the following properties for the group:
   - Array Size — Enter the desired number of onscreen occurrences. All members of the synchronized group adjust to the specified size.
   - Max Occurrences — Enter the total number of occurrences. Set the property to blank to specify an unlimited number (the actual maximum is platform-specific). A value greater than that specified in the Array Size property allows the array to scroll.
   - Scroll Increment — Specify the number of occurrences by which all members should scroll when the user presses Page Up or Page Down. The default is one less than the number of onscreen occurrences.
   - Circular — Set the property appropriately.
     – Yes — Causes the first element to redisplay after the last element in the group is reached, and vice versa.
     – No — (default) If the user presses the arrow keys to move the cursor to the last/first element of the group, the cursor moves to the next/previous field. If the user presses Page Down/Up to scroll the occurrences, JAM displays the message "`No more data`" when the first/last element is reached.

Application Development Guide

Changing Widget Types at Runtime

The information in JAM 7.0 about changing a widget's type at runtime is incorrect. While you can determine or get a widget's type, the Widget Type property is a read-only property and cannot be changed at runtime.

Changing Widget Properties

Obsolete Functions

Several functions, made obsolete in JAM 7.0, were erroneously documented. Specifically, `sm_set_prop`, `sm_fset`, `sm_achg`, and `sm_protect` were cited as

functions to use for changing a widget's property values at runtime. Use JAM's property API to change properties for widgets as well as for screens and the application as a whole.

In addition, Table 65 was rendered obsolete by the new property API and should therefore be disregarded.

LDB Functionality

In general, you should regard LDBs as passive recipients of data. Although an LDB is created and edited as a screen, at runtime, JAM does not perform most of the processing that is otherwise associated with screens. Screen entry and exit functions are not executed; and no validation or formatting is performed on entry data. For example, no updates occur for an LDB entry that is formatted as a date/time field with `system_update` set to `PV_YES`.

Transaction Manager Commands

The following information was omitted from the descriptions of the `START` and `CHANGE` transaction manager commands.

Specifying Your Own Transactions

As part of its processing, `FINISH` closes the current transaction, which has been set with the `START` or `CHANGE` commands. In cases where you initiate a transaction by calling the `START` command, you must also call the `FINISH` command to close that transaction before closing the transaction's screen. Note that you may need to call the `CHANGE` command to make the transaction active before closing it with the `FINISH` command.

Application Traversal Properties

Traversal properties can be specified for an application as well as for table views and server views. When these properties are specified for the application using `@jam` (instead of a server view or table view name), JAM returns the value based on the root table view of the current transaction. The applicable properties are:

- `sv` — The root table view of the current transaction.
- `num_svs_below` — The number of server views in the current transaction.
- `sv_below` — The name of the server view in the current transaction which corresponds to the specified number.
- `num_tvs_below` — The number of table views in the current transaction.

- `tv_below` — The name of the table view in the current transaction which corresponds to the specified number.
- `num_fields_below` — The number of fields in the current transaction.
- `field_below` — The name of the field in the current transaction corresponding to the specified number. Note that the position of the field on the screen does not determine the numbering sequence and any modifications to the table view members can change the numbering sequence.

Accessing Transaction Information

The following information is provided as a supplement to the existing documentation and to help you customize transaction manager behavior. To customize transaction manager behavior, you might need to access various property settings or obtain information about the current transaction. Information is available in the following ways:

- Via transaction manager functions
- Using transaction manager variables
- Using the property API

Functions

Several functions are used by the transaction manager, but two of them are specifically used to obtain information about the current transaction—`sm_tm_pinquire` and `sm_tm_inquire`. Both functions have a series of parameters whose values can be requested either in C or in JPL. For example, the name of the current transaction or the root table view of the current transaction are two of the settings available. Some of these parameters can be set by using the functions `sm_tm_iset` and `sm_tm_pset`.

For a complete listing of the parameters, refer to the documentation for `sm_tm_inquire` and `sm_tm_pinquire`.

Transaction Manager Variables

There are transaction manager variables available for transaction manager hook functions written in JPL.

Note: *If these variables are referenced outside of a hook function, the results are unpredictable.*

Variable	Description	Availability
`@bi(`*field*`)[`*occurrence*`]]`	Access the before image value of the specified field and occurrence. *field* can be either the field name or the field number. The variable `@tm_occ` can be used to specify the current occurrence.	In a hook function using the `TM_DELETE_EXEC`, `TM_INSERT_EXEC`, or `TM_UPDATE_EXEC` events.
`@tm_occ`	Occurrence number being processed. Equivalent to `sm_tm_inquire(TM_OCC)`. A negative value indicates a deleted occurrence.	In any hook function
`@tm_occ_type`	Code reflecting the change, if any, from its before image. Equivalent to `sm_tm_inquire(TM_OCC_TYPE)`. Refer to page 366 in the *Application Development Guide* for the code values.	In any hook function
`@tm_pocc`	Parent occurrence number. Equivalent to `sm_tm_inquire(TM_PARENT_OCC)`.	In any hook function
`@tm_save_cursor`	Name of the cursor used for non-`SELECT` statements. Equivalent to `sm_tm_pinquire(TM_SAVE_CURSOR)`.	In any hook function
`@tm_sel_cursor`	Name of the cursor used for `SELECT` statements. Equivalent to `sm_tm_pinquire (TM_SV_SELECT_CURSOR)`.	In any hook function

Example

The following hook function performs a “logical” delete on a database row. Instead of physically removing the row from the database, the two slices from the `TM_DELETE` event, `TM_DELETE_DECLARE` and `TM_DELETE_EXEC`, mark the row as deleted so that it can be excluded from selection.

For this example, the `customers` tables includes a column named `deleted`. The column is used to flag the record so that it is excluded from selection.

Note the use of`@tm_save_cursor` to supply the cursor name, `@bi` to obtain the before image value for `cust_id`, and `@tm_occ` to supply the occurrence number.

```
proc logical_delete_hook (event)
{
    if (event == TM_DELETE_DECLARE)
    {
        if (@tm_occ < 0)
        {
            DBMS DECLARE :@tm_save_cursor CURSOR FOR \
                UPDATE customers \
                SET deleted = 1 \
                WHERE cust_id = ::p1
            return TM_CHECK
        }
    }
    else if (event == TM_DELETE_EXEC)
    {
        if (@tm_occ < 0)
        {
            DBMS WITH CURSOR :@tm_save_cursor EXECUTE USING \
                @bi(cust_id)[:@tm_occ]
            return TM_CHECK_ONE_ROW
        }
    }
    return TM_PROCEED
}
```

Property API

In addition to the properties displayed in the Properties window for each object, there are additional properties in the transaction manager containing information about the current traversal tree. These properties, known as *traversal properties*, return information about table views and server views in the tree, like the number of fields in a server view or the table view for a particular field.

Note: *To determine the root table view of the current transaction, call* `sm_tm_pinquire(TM_ROOT_NAME)` *or use* `@jam->sv`. *The* `root` *property is a screen property and does not necessarily describe an active transaction.*

For a description of each of the traversal properties, refer to page 380 in the *Application Development Guide*. All of the properties are documented in the Properties tables. To view the tables, refer to page 519 in the *Language Reference*.

Error Processing in Database Applications

The following example is provided to clarify how a C hook function is installed for handling database error functions.

A C hook function is installed as follows:

`DBMS { ONENTRY | ONEXIT | ONERROR } CALL` *function*

where *function* is a prototyped function that takes three arguments: two strings and an integer. For example, the following entry in the `pfuncs` structure installs `myfunc` as a prototyped function that returns an integer:

```
static struct fnc_data pfuncs[] =
{
       ...
       SM_INTFNC(”myfunc(s,s,i)”, myfunc),
       ...

};
```

For more information on installing prototype functions, refer to page 121.

Stub Functions

Stub functions are no longer supported in JAM 7, therefore, the section and table in the *Application Development Guide* should have been omitted.

Configuration Guide

SMLPRINT and the LP Logical Key

The information about the `SMLPRINT` variable and the local print (LP) key was misleading. The variable specifies the operating system command that is invoked through the LP key. The output depicts the screen in character-mode, regardless of the platform in which the command is issued.

Setting Number of Decimal Places

The definition of the setup variable, `DECIMAL_PLACES` was documented incorrectly. The following description clarifies the variable’s possible settings and how it is used by JAM.

`DECIMAL_PLACES` is used to set the default decimal places for real number display. There are two possible settings:

- `PLACES_VARIABLE` — (default) Sets the default number of decimal places used by `sm_dtofield` to equal the number of significant digits in the number, to a maximum of 20. JAM uses `sm_dtofield` to display real numbers.

- *number* — Sets the default number of decimal places used by `sm_dtofield` to *number.*

Logical Keys

The following logical keys were new to JAM 7.0, but were not documented on-line:

Table 1. Additional JAM logical keys

Mnemonic	Hex	Long Name	Description
ADDM	`0x135`	Add Mode	toggle extended selection capability in list box
BOFD	`0x133`	Beginning of Field	go to first position in entry field
BOLN	`0x12e`	Beginning of Line	go to beginning of line (widget)
EOFD	`0x134`	End of Field	go to last position in entry field
EOLN	`0x12f`	End of Line	go to end of line (widget)
EXT	`0x136`	Extend Selection	extend selection to include contiguous items in list box
EXTD	`0x137`	Extend Selection Down	extend selection using down arrow key in list box
EXTU	`0x138`	Extend Selection Up	extend selection using up arrow key in list box
LWRD	`0x11c`	Left Word	go to previous (left) word
RWRD	`0x11d`	Right Word	go to next (right) word

Use or Discard Error Acknowledgment Key

The definition of the setup variable `ER_KEYUSE` was documented incorrectly. The following description clarifies the variable's possible settings.

`ER_KEYUSE` is used to control how JAM accepts or discards keyboard input when an error message is displayed. There are two possible settings:

- `ER_NO_USE` — (default) Forces acknowledgment of all window-displayed error messages by the key defined as `ER_ACK_KEY`, spacebar, Enter, or by choosing OK; error messages that are displayed to the status line must be acknowledged with either `ER_ACK_KEY` or spacebar. The valid keypress is immediately discarded.

All invalid responses to a window-displayed error message cause the terminal to beep (through calls to `sm_bel`). All invalid keypresses to a status line message cause JAM to display an error window or beep, depending on how `ER_SP_WIND` is set.

- `ER_USE` — Any keypress acknowledges a status line error message; this setting has no effect on window-displayed error messages. The type-ahead buffer is flushed when the message is displayed, and the acknowledging keypress is saved for data-entry. If you set this as the default, you can force users to acknowledge a message by putting `%Md` at the beginning of the message text. Refer to Chapter 5 in the *Configuration Guide* for more information.

Windows-Specific Information

Specifying an Initialization File

The ability to specify an application's initialization under Windows was implemented in JAM 6.1, but was omitted from the JAM 7 documentation. It is included here for clarification.

At startup, JAM looks for the initialization file in the following places, listed in order of precedence. You can use any of these methods:

- The name of the `.ini` file can be compiled into the application executable — Supply the name of the initialization as an argument to `sm_pi_mw_setup` in the file `piinit.c` for a JAM application executable, or to `sm_pi_mw_jxsetup` in the file `pijxinit.c` for a JAM development executable. Edit the appropriate source file, which is in the `link` directory, then recompile.

  In the default distribution, these functions are supplied an empty string argument.

- The name of the `.ini` file can use the executable name — This is the most convenient and flexible method. Use an initialization file whose base name is the same as the JAM executable, for example, `LEADS.INI` for the executable `LEADS.EXE`. The initialization file must be in the Windows directory.

- The default initialization file — This is `JAM7.INI` and supplied with JAM and installed in the Windows directory.

JAM Help Entry

The `WinHelp` option was removed from the JAM initialization file (`[JAM Help]` section) in JAM 6.03. This option is no longer supported.

Database Driver – ODBC

The following information is provided specifically for the ODBC database driver:

Formatting for Colon Plus Processing and Binding

Date/Time Formats

JAM uses the ODBC standard formats for converting JAM date values to valid ODBC values. If a JAM widget has date and time property settings, it is formatted as a timestamp; if the widget has only time-specific property values, it is formatted as a time; and if the widget has only date-specific property values, it is formatted as a date. The formats are:

Column Type	Format
Date	`{d 'YYYY:MM:DD'}`
Time	`{t 'HH:MM:SS'}`
Timestamp	`{ts 'YYYY:MM:DD HH:MM:SS'}`

where `YYYY`, `MM`, `DD`, `HH`, `MM`, and `SS` are specified as integers.

Autocommit

JAM 7.01 for ODBC sets the transaction processing to `AUTOCOMMIT OFF`. As a result, when a recoverable statement (`INSERT`, `UPDATE`, and `DELETE`) is executed, it is not automatically committed. The effects of the statement are not visible until the transaction is terminated. If the transaction is terminated by `DBMS COMMIT`, the updates are committed and visible to other users. If the transaction is terminated by `DBMS ROLLBACK`, the updates are not committed, and the database is restored to its state prior to the start of the transaction.

Database Driver – ORACLE

The following information is provided specifically for the ORACLE database driver:

Formatting for Colon Plus Processing and Binding

Specifying Optimization Hints

In ORACLE, you can specify the optimization of a SQL statement by including hints in the statement itself. This is done by putting the hint inside a comment block in the Oracle SQL statement. Since the syntax for hints matches JAM's syntax forcomments, you must escape (use a backslash) the comment block (the first slash) to ensure that the hint is interpreted correctly, and not as a JAM comment.

For example, to include the hint `/*+ ALL ROWS */` in the SQL statement, the statement would be written as:

```
DBMS SQL SELECT \/*+ ALL ROWS */ empno, ename, job FROM emp
```

Refer to your ORACLE documentation for more information on using hints.

Declaring Cursors

For OCI applications, JAM does not put any limit on the number of cursors an application can declare to an ORACLE engine. For Pro*C applications, JAM defines 10 cursors for an application accessing ORACLE. It reserves one for itself (i.e., the “default” cursor); the other nine are available for the application’s use. If the application attempts to declare a tenth cursor, JAM returns the `DM_MANY_CURSORS` error. In this case, the application must close a cursor using `DBMS CLOSE CURSOR` before it can declare a new one. If nine cursors are not enough for your application, you must modify the distributed source file `oraemb.pc`.

Using Stored Subprograms

Executing Stored Procedures

For output parameters, if the destination widget is an array with the Word Wrap property set to Yes, you must also specify a maximum number of occurrences for that widget. The amount of data requested from ORACLE is determined by the size of the destination variable.

Language Reference

The following information and functions required clarification or were not included in the JAM 7.0 documentation:

- Comments in JPL.
- Description of how JAM searches for and opens a screen named in a control string or specified in library functions.
- Property API information.
- `dm_getdbitext`— Revised description (page 24).
- `sm_calc` — Clarification of arguments (page 25).
- `sm_filebox`— Correction to example (page 25).

- `sm_fio_open` — Corrected return value information (page 25).
- `sm_message_box` —Rewritten (page 52).
- `sm_mncrinit`— Incorrect function name (page 26).
- `sm_ms_inquire` — Corrected return value information (page 26).
- `sm_prop_get` and `sm_prop_set` — Correction to an argument example (page 26).
- `sm_slib_install` — Correction and clarification of the function (page 58).
- `sm_strip_amt_ptr` — Corrected parameter definition and clarification of return values (page 27).

Comments in JPL

While you can enclose a commented string with the block specifiers `/*` and `*/`, you cannot embed comments within a line of code; all text on a line that follows a comment character, including `*/`, is ignored at runtime.

Property API Supplement

The following information was omitted or incorrect in the JAM properties tables that list the JPL mnemonics and values for JAM properties.

class
Widget Transaction property takes a string as a value. The `str` is the name of a class, `–none–`, `–default–`, or an empty string. An empty string indicates that the property is set to `–default–`. Refer to the *Editors Guide*, page 285 for more information.

drop_down_data
Widget Identity property has no constraints other than the `widget_type = PV_OPTION_MENU` or `PV_COMBO_BOX`.

drop_down_screen
Widget Identity property has a constraint of `drop_down_source = PV_EXTERNAL_SCREEN`.

drop_down_source
Widget Identity property has two possible values: `PV_CONSTANT_DATA` and `PV_EXTERNAL_SCREEN`.

grid_current_occ (PR_GRID_CURRENT_OCC)
Widget Geometry property was omitted from the JAM 7.0 online documentation and is a runtime-only property. Use this property to determine the current row in a grid frame, that is, the occurrence that is highlighted or striped. It returns an integer value.

placement
Widget Format/Display property has the following constraint: `numeric_type = PV_CUSTOM`.

root
Screen Transaction property returns a string value. The `str` can contain a table view name, `–none–`, or an empty string. An empty string indicates that the property is set to `–default–`.

To get the root table view of the current transaction, use `@jam->sv` or call `sm_tm_pinquire(TM_ROOT_NAME)`.

Search Path

The search path that JAM looks for a screen (specified in a control string or specified via the functions `sm_form` and `sm_window`) did not describe how JAM handles the search when the screen specification does not include the file's extension.

When you use `sm_r_form`, `sm_r_window`, or specify a screen to open in a control string, JAM looks for the named screen in the following places in this order:

- The memory-resident screen list.
- The open libraries.
- On disk in the current directory.
- Along the path supplied to `sm_initcrt`.
- Along all the paths in the setup variable `SMPATH`. If any path exceeds 80 characters, it is skipped.

If the search fails and the supplied file name has no extension, JAM appends the `SMFEXTENSION`-specified extension to the filename and repeats the search. If all searches fail, JAM displays an error message and returns (using the appropriate function, if applicable, to display the message).

dm_getdbitext

Description

`dm_getdbitext` lets you get the full text of the last-executed `DBMS` command. This includes all commands executed from JPL with `DBMS` or executed from C with `dm_dbms` or `dm_dbms_noexp`.

You must call this function from within an installed entry, error, or exit handler. Since this function uses a buffer shared with other functions, you must either process the returned string immediately or copy it to another variable for additional processing.

This is the same string that is passed to the first argument of an installed entry, error, or exit handler, except that the error or exit handler is limited to 255 characters.

sm_calc

The arguments documented in JAM 7.0 were described incorrectly. They are:

field_number
The field to use for relative field references, for backward compatibility only. If `expression` references fields according to current conventions, supply 0.

occurrence
The occurrence in `field_number` to use for relative field references, for backward compatibility only. If `expression` references fields according to current conventions, supply 0.

expression
A math expression. Refer to page 316 in the *Editors Guide* for information on creating math expressions.

sm_filebox

The example used for `sm_filebox` has a missing backslash and should read as:

```
sm_filebox(buf, LEN, ”c:\\videobiz”, ”*.tbl”, ””, FB_OPEN);
```

sm_fio_open

A return value of ≥ 0 (not ≥1) is a handle to the opened file.

In addition, the description of `sm_fio_open` should indicate that `sm_fio_open` opens a file in the specified mode and returns an integer handle to a file stream that is accessible to other JAM library file I/O functions. Supply this handle to these functions for all subsequent I/O operations on the file stream.

Note: *The file stream that is opened by* `sm_fio_open` *is not accessible to standard C library functions.*

sm_message_box

The information on `sm_message_box` was edited extensively. The entire function description was rewritten and is included in this document (refer to page 52).

sm_mncrinit

The function name was changed to `sm_mncrinit6`, making `sm_mncrinit` obsolete. The function description is the same.

sm_ms_inquire

The order of the information for right and middle mouse buttons was reversed, both in the text and in the illustration. The correct order, from low order bits to high order bits is left, right, and middle.

For example, the bit settings returned for a just-initiated point and click operation—left button is down and just pressed—can be represented as follows:

Middle Button			Right Button			Left Button		
0	0	0	0	0	0	0	1	1

A click and drag operation that is in progress—right button is down—can be represented like this:

Middle Button			Right Button			Left Button		
0	0	0	0	0	1	0	0	0

sm_prop_get/sm_prop_set

In the description of the argument `prop_item`, the reference to a `prop` value of `SM_PR_CONTROL_STRING`, should be `PR_CONTROL_STRING`.

sm_slib_install

The function `sm_slib_install` (installs a function from a DLL into a JAM application) was documented incorrectly. The rewritten description is included in this document (refer to page 58).

sm_strip_amt_ptr

The `inbuf` argument and the returns for the function `sm_strip_amt_ptr` were documented incorrectly. The corrections are as follows:

inbuf

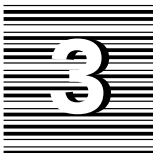
Contains the string to strip. For example, this JPL statement strips the supplied string of its currency symbol and comma, and puts 123489.12 into `amt`:

```
amt = sm_strip_amt_ptr( @NULL, ”$123,489.12” )
```

To use the data in the `field_name`/`field_number` argument, supply `NULL`.

Possible return values for `sm_strip_amt_ptr` are:

- A pointer to a buffer containing the stripped text.
- 0 if `inbuf` is set to `NULL` and the field number is invalid.



Using JAM for Macintosh

The JAM online documentation includes, in addition to JAM 7.01 hyperlinks, notes with information on using JAM for the Macintosh. An apple icon in the right margin is the placeholder for the notes throughout the online documentation. The notes provide information on how JAM supports the Macintosh. They have topic-specific titles that reflect their content.

To view the list of annotations in DynaText:

1. Choose Book⇒Manage Annotations from the DynaText menu.

   The Annotation Manager dialog box displays an alphabetized list of the annotation titles. Notes preceded with an apple icon and having a owner/user `JamMac` indicate the JYACC-distributed annotations.

2. Double-click on an note title or choose GoTo to take you to the location of a note in the online documentation.

3. Double-click on the apple icon to display the contents of the note. Click on the close box to close the note.

Setting JAM for Macintosh Defaults

JAM applications that run under Macintosh use a preferences file to set defaults for the Graphical User Interface (GUI). The preferences file is a ResEdit document whose resources control the appearance and behavior of JAM applications running on the Macintosh. You can access these resources to set up the initial state, and your users can change these settings to suit their preferences.

Preferences file settings override any duplicate settings in the file pointed to by the configuration variable `SMVARS`.

Preferences File

The preferences file contains two resources:

- Strings table (`STR`), which is edited through the Control Panels option on the Apple menu in the `JamConfig` file, contains JAM variables and their settings. These variables let you set default application appearance and general application behavior.
- Color look-up table (`clut`), which is edited using ResEdit, defining the RGB values of JAM colors. Refer to page 33 for instructions for editing this table.

filename

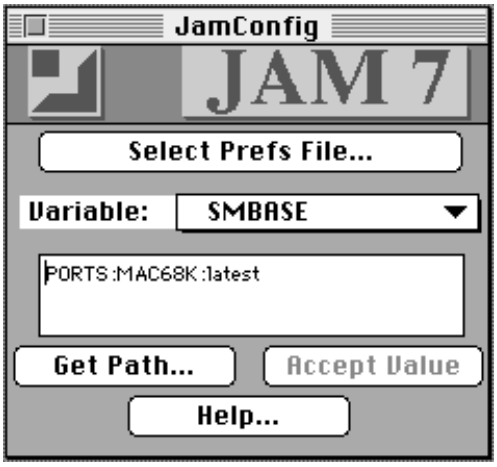
The installation program places a `Jam7` file in the Preferences folder and sets `JAM7` as the default preferences filename. The name of this file is set by the argument to JAM's GUI initialization function. To change the preferences filename, edit the argument to the function `sm_pi_mc_setup` in the file `piinit.c` for a JAM application executable, or the function `sm_pi_mc_jxsetup` in the file `pijxnit.c` for the JAM development executable. These source files are in the `Linking` folder.

At initialization, the main routine of your application (usually either `jmain.c` or `jxmain.c`) calls either the function `sm_pi_init` or `sm_pi_jxinit` to initialize the GUI. These routines in turn call `sm_pi_mc_setup` or `sm_pi_mc_jxsetup`, which set the name of the preferences file.

Accessing the Control Panel

To edit the variable settings:

1. Choose the Control Panels option on the Apple menu.
2. Open the `JamConfig` file from the list of control panels.
3. On the JamConfig window, click on Select Prefs File. The list of preferences files are displayed.
4. Select the `Jam7` file, located in the Preferences folder.



5. Expand the Variable option menu.
6. Select the variable whose value you want to edit.
7. After editing the variable's value, choose Accept Value.
8. If you click on Get Path, you can select a file from a dialog box and display the path for the selected file. The path is displayed in the JamConfig window.

Setting Variables

The variables control default application appearance and behavior. You can access these variables, which are stored in the `STR` resource, through the JAM Control Panel option on the Apple menu. Many of the variables are standard JAM configuration variables and are documented in Chapters 3 and 4 of the *Configuration Guide*. The variables that have Macintosh-specific settings are documented later in this section.

file path syntax

Strings that specify file paths—for example, `SMBASE` or `EditorHelpPath`—must use colons (`:`) as path delimiters. For example, the following string sets the path to `EditorHelpPath`:

```
$SMBASE:docs:books
```

Use `$SMBASE` for relative paths; in general, this method is best to facilitate portability across different platforms. Absolute paths must begin with the name of the hard disk. For example:

```
hd:System Folder:Preferences:Jam7
```

Macintosh-Specific Settings

The following section documents those variables that have Macintosh-specific settings:

`aboutPixmap`
Enter the name of a PICT resource for your application's resource. The `aboutPixmap` is displayed by choosing the About Box from the Apple menu. If

`aboutPixmap` is not set, the `introPixmap` (splash screen specification) setting is used.

To change the text of the About Box menu command:

1. From the application's resource file, choose the `MENU` resource.
2. Choose `MENU` resource #1 (displays the Apple icon).
3. Choose `About JAM 7`.
4. Replace the words "JAM 7" with the desired text for your application.

To include version data in the About Box:

1. From the application's resource file, choose the `vers` resource.
2. Enter the desired version data.

`Background`
Set to the name of a color in the preferences file `clut` resource. This color becomes the background color of a screen, if the screen's Color property or form background in the configuration map (`cmap`) file is not specified. The search order for determining screen background color is as follows: screen Color property, `cmap` default, and preferences file resource setting.

`CommandLine`
Executes any string entered here when JAM is launched. For example, you can invoke JAM in application mode as opposed to edit (screen editor) mode by entering the appropriate arguments. For information about startup options (refer page 4 in the *Editors Guide*).

`EditorHelpPath`
Sets the path to the folder containing screen editor online help. The default installation places the help files in `$SMBASE:docs:books`.

`font`
Obsolete for JAM 7.01. Font names are specified and defined in the `cmap` file located in the `Config` folder.

`Foreground`
Set to the name of a color in the preferences file `clut` resource. The foreground color defines the color of the dots of the screen's positioning grid in edit mode. This specification is the only resource for defining a screen's foreground color

since there is no screen Foreground Color property or a `cmap` entry for a screen's foreground color.

`HideHiddenWindows`
Set to Yes or No. Yes hides screen editor windows during Test mode. The default is set to Yes.

`JamFonts`
Set to a font name and size that serve as the application default font, for example, `geneva–12`. In the absence of any other font specification (at the screen- or widget-level) or in the `cmap` file (`default_font` specification), all application components—screen titles, labels, menu items, and so on, use this font and font size.

The search order for determining the application default font is as follows: `cmap` file, `JamFonts`, system font (set through the Views control panel).

`MoveThreshold`
Set to the number of pixels that the cursor must be moved in order to be considered a drag instead of a click. This applies when dragging the rubberband, or creating, moving, or resizing an object. The default is set to 10 pixels.

`StackedWindowsAreDialogs`
Set to T/True or F/False. A True setting forces all stacked windows in a JAM application to behave as dialog boxes. As dialog boxes, they are modal and have a dialog style border. If you set this option to True, you should avoid changing the window stack order or sibling relationships between windows. The default is set to True.

`TrueType`
Set to True if TrueType fonts are used, and False if bitmapped fonts are used. The default is set to True, and the default font displays as TrueType.

Redefining JAM Colors

JAM provides sixteen basic colors—eight highlighted and eight unhighlighted. You can map these colors to any of the colors supported by the GUI. The mapping between JAM colors and GUI colors defines your color palette.

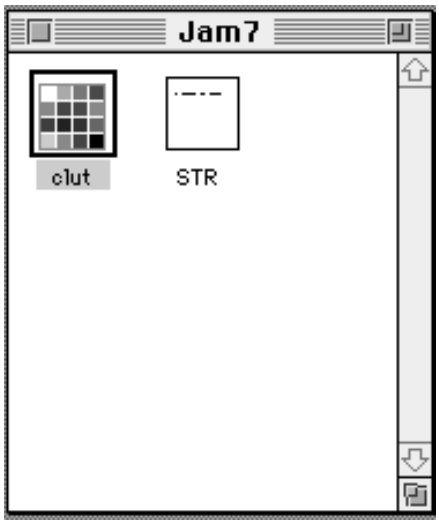
To create custom colors beyond the provided sixteen basic colors, use the configuration map file, described in the *Configuration Guide* (refer to Chapter 8).

Most color monitors support at least sixteen primary colors. These sixteen primary colors are mapped to palette colors in the `clut` resource:

`black`	`red`	`hi_black`	`hi_red`
`blue`	`magenta`	`hi_blue`	`hi_magenta`
`green`	`yellow`	`hi_green`	`hi_yellow`
`cyan`	`white`	`hi_cyan`	`hi_white`

You can use the `clut` resource to access JAM color definitions and modify them:

1. Launch ResEdit and open (File⇒Open) the `Jam7` preferences file.



2. Select the `clut` resource icon and open it for editing (Resource⇒Open clut Picker). JAM primary colors are listed in tabular format, along with their resource IDs and attribute values.

3. Select a color entry and edit its color definition by choosing Resource⇒Open Resource Editor. A dialog box displays that shows the color's current RGB values. You can edit the color definition in two ways:

   - Type in the desired RGB values.
   - Click on the color square and choose Resource⇒Open Color Picker. The color wheel dialog box displays. Change the color definition by manipulating the color wheel. Choose OK to accept your changes.

4. Accept the color changes by closing the color definition dialog box. When you exit the preferences file, the resource editor prompts you to save your changes.

Setting SMBASE

To run JAM, the `SMBASE` variable must be set to the path of the JAM folder. Generally, this variable is set during your installation program. However, if you need to change the variable or you need to move the JAM folder, the `SMBASE` variable must be redefined. `SMBASE` is set in the Preferences file's `STR` resource and has the following syntax:

hard-drive-name:*path-to-JAM-folder*

For example, for a given installation of JAM in the folder `Jam7` on the hard drive `mike`, `SMBASE` should be set as follows:

```
mike:Jam7
```

Setting Properties to Resource Names

Any property that is set to the name of a graphical resource—for example, a label's Active Pixmap property or the screen's Pointer property—must reference a resource that is stored under that name either in the resource fork of the application or in a file accessible by the application. Other application resources, such as message icons and the application's logo, are stored in the executable. You can examine and edit these resources through a resource editor such as ResEdit. The next section shows how to edit an application's PICT resources.

Graphical resources must be in a Macintosh PICT format. If they are stored in a file, the name of the file must exactly match the name specified in the property, without any file extension. JAM looks for the file in the following location order:

- Resource fork of the application.
- The path specified in the JAM property.
- Current folder in use by the application.
- Folders specified by `SMBASE`.
- Folders specified by `SMPATH`.

Note: *Macintosh resources are all stored under integer identifiers; you can refer to resources by their IDs; however, for cross-platform portability as well as easier reference, all resources must be assigned names.*

Editing PICT Resources

A JAM for Macintosh application that uses pictures—for example, as a label's Active Pixmap property or message icon—must have those pictures stored as PICT resources within the executable or as PICT files. Any picture saved in PICT format can be stored in a JAM for Macintosh application as a PICT resource.

A JAM application contains a number of PICT resources that JAM itself uses. These are available to all JAM applications. Because JAM requires these resources, they should be left unchanged. However, you can supplement them by adding new PICT resources to the application. Because editing an executable's resources requires no recompilation, any changes that you make are immediately accessible. This can be especially useful during the development process.

You can also store an application's pictures in its own resource file and link this into the executable. These PICT resources supplement the pictures already in the JAM executable `jamdev`. The default distribution contains a resource file, `jam.pi.rsrc`, that you can edit and link into an application.

In general, you should edit the executable's resources only to achieve short-term goals—for example, to test a label's pixmap properties during development. To make permanent changes, edit the project's resource file and relink it into the executable.

Editing PICT resources requires these tools:

- A Macintosh resource editor such as ResEdit. Because JYACC provides ResEdit, usage of this utility is assumed in the sections that follow.
- A graphics application that can read the pictures that you want added to the JAM application.

Relinking PICT resources into your JAM application requires a compiler/linker such as Symantec's C++ THINK Project Manager or Metrowerk's CodeWarrier IDE.

Editing an Executable's Resources

To modify an executable's PICT resources, use this procedure:

1. In ResEdit, choose Open⇒File and open the JAM application whose PICT resources you wish to edit. If you are launching ResEdit, it displays a dialog box that prompts you to open the desired file.
2. Open the PICT resources: select the PICT icon and choose Resource⇒Open PICT Picker. All PICT resources in the application display as icons.

You can now replace the contents of current resources, or add and remove resources. For information on adding resources, refer to page 37.

Only replace or remove the contents of PICT resources that you have added; replacing or removing the contents of PICT resources used by JAM can yield unpredictable results.

Note: *This method of making resource changes affects only the current executable; to propagate changes to subsequent versions of an application, you should edit the resource file* `jamdev.pi.rsrc` *itself. Methods for doing so are described in the next section.*

Editing the Resource File

To make resource changes that are available to all versions of a JAM application, you must modify the resource file for that application's project and relink it into the executable. There are two methods for doing this:

- Edit the executable's resource file directly. With ResEdit or another resource editor, you can make the desired changes to the resource file—change the contents of existing PICT resources, remove resources, or add new ones.
- Create a separate resource file for PICT resources and make the desired changes there. Then add this resource file to the project.

Although it is more straightforward to edit the application's resource file directly, the second method ensures that your PICT resources remain intact in the event that the resource file is corrupted or lost. In either case, you should always make a copy of your resource file before editing it or relinking it into the application.

The following procedure shows how to maintain PICT resources in a separate resource file:

1. Create a resource file for the application's pictures:
   - Launch ResEdit and choose New.
   - If ResEdit is already running, choose File⇒New.
2. Add the desired PICT resources to the new resource file. Refer to page 37 for information on how to do this.
3. Save the new resource file. The file name must have the `.rsrc` extension.
4. Compile and link with either Symantec's or Metrowerks's compiler/linker. The PICT resource file is added to the project.

Adding PICT Resources

To add a PICT resource:

1. Open the PICT resource.

2. Choose Resource⇒Create New Resource.

   The Select New Type dialog box prompts you to specify the resource type.

3. Select `PICT` from the list box or type it in directly.

   ResEdit creates and opens a PICT resource; this resource gets the next available resource ID. If this is the resource file's first PICT resource, ResEdit also creates a top-level PICT icon and its corresponding window.

4. Paste the clipboard contents into the new PICT resource.

5. Choose Resource⇒Get Resource Info. from ResEdit's Resource menu.

   ResEdit displays the picture's Info dialog box.

6. In the Name field, assign a name to the PICT resource. JAM applications use this name to access the picture.

7. Mark the Purgeable check box. This allows the Memory Manager to remove this picture from memory if necessary.

8. Save your changes by choosing File⇒Save.

The Keyboard Interface

When the JAM documentation refers to the following commands or keystrokes, use the Macintosh equivalent:

Keystroke specification	Macintosh equivalent
Ctrl+click	Command+click
Ctrl+drag	Command+drag
Alt key	Command key or Option key
Exit	Quit

- The default key file for the Macintosh defers to Macintosh convention and maps the F2 key to Cut. If you want F2 to invoke Test mode from the screen editor, set the `SMKEY` variable to `macjkeys.bin`.
- In the key file, Alt key accelerators map to command key equivalents. For example, Alt A means Command A.

Using the Screen Editor

The following items describe restrictions and behavior that is unique to running JAM and developing JAM applications on the Macintosh:

Pop-up Menus

Pop-up menus for menu bar shortcuts are not available with this release.

Dialog Boxes

JAM dialog boxes only allow focus on non-button widgets. This behavior is consistent with Macintosh dialog conventions.

Business Graphs

Business graphs are not available for the Macintosh in the current release. Although you can set all graph widget properties, a graph is visible onscreen only as a box. However, the graph will be displayed when the screen is opened under Windows or Motif. Because you can set graph widget properties, the Create⇒Graph option and Graph tool are active.

Screen Properties

The following screen properties can be set (for non-Macintosh platforms), but have no effect on the Macintosh. This allows JAM applications running under Macintosh to be consistent with native conventions:

- Icon property — Screens under Macintosh cannot be minimized, and therefore require no icon for minimized screens.
- Border and all its subproperties — Screens always have borders on the Macintosh; their display properties are handled by the operating system.
- Title Bar — Screens always have title bars on the Macintosh.
- System Menu — Screens never have a system menu on the Macintosh. However, the Close Item subproperty is supported. Thus, you can set the Close Item subproperty to No to remove the close box from JAM screens on the Macintosh.

Grid Frame Widgets

Grid widgets on the Macintosh were implemented with the following features:

- Grid widgets on the Macintosh do not support the Frozen Columns property in the Format/Display category.

- Clicking on either a grid column title or a grid cell allows you to select and drag columns.
- Each grid column can have its own font assignment. Therefore, the height of a row is equal to the height of the tallest font.
- The background color of the stripe row is defined by the Color Control Panel and corresponds to the “highlight color.”

Line/Box Properties

Lines and boxes support these values for the Line/Box Style property:

```
Dash
Dot
Dashdot
Dashdotdot
Double Dash
```

Fonts

The default font displays as TrueType. The `TrueType` option in the JAM7 preference file lets you choose between TrueType fonts and bitmapped fonts in the screen editor.

If the Set to True if TrueType fonts are used, and False if bitmapped fonts are used.

Font Point Sizes

The default font size is 12 points on the Macintosh.

The 8 point font size is replaced with a 9 point font size. However, you can add the 8 point font size back into the configuration map file if required.

File menu option in application/test mode

The JAM application/test mode menu has a File option whose drop-down menu contains a single item, Quit. Choosing this item returns you to the calling environment—for example, from test mode back to the screen editor.

Using the Menu Bar Editor

The following restrictions apply to the menu bar editor:

- Menu item mnemonics are not accessible on the Macintosh.

- Menu items that are defined as Windows Operations types are ignored.
- Top-level menu items cannot be set to action types; action types can only be submenu types.
- The toolbar on the Macintosh requires toolbar pixmaps to be 16 pixels wide by 15 pixels high or smaller in order for them to not be cut off.

Note: *The above features are restrictions on the Macintosh. However, if they are created using menu script on the Macintosh, they will display when the screen is opened under other platforms.*

Using JPL

The following items describe restrictions and behavior that is unique to running JAM on the Macintosh:

JPL Variables

JPL variables are case-sensitive, so you need to be aware of case when naming and specifying JAM variables, widget names and screen names. For example, the following syntax that specifies a field on a screen is case-sensitive:

```
screen!field
```

Filenames

If a filename consists of more than one word or contains spaces on the Macintosh, you must include the filename in double quotation marks. Filenames are often used as arguments to the JPL commands `public` and `unload` or in control string specifications. If a pathname is specified, and any item in the path contains a space, the entire pathname must be enclosed in double quotes:

```
"hd:Desktop Folder:myfile"
```

JPL Editor

To write or edit your JPL procedures in the screen editor, you can either edit text directly in the JPL Program Text window or choose the Editor button. If you choose Editor, JAM automatically launches SimpleText. To exit SimpleText, select File⇒Quit in order to return to the JPL Program Text window. If you close SimpleText by using its close box, SimpleText remains active and the JPL window is *not* updated.

Setting JAM Properties at Runtime

In order to change JAM properties at runtime, you can use JAM's property API or you can write a JPL procedure which can then be executed as an Apple event using

AppleScript. As an example, set the Font Name property for the Code widget to Helvetica. A JPL procedure called `prop_chg` expects three arguments: the widget name, the property, and the property value.

```
proc prop_chg(name, prop, value)
   @widget(name)->"@property"(prop) = value
   return
```

The AppleScript code to execute the `prop_chg` procedure would be:

```
execute "^prop_chg(\"Code\", \"font_name\", \"helvetica\")"
```

Using JAM Library Functions

sm_shell

A call to the library function `sm_shell` must format its first argument as follows:

"app-name[file-name]"

If you specify a file to open, its name must be preceded by a space. If the program or filenames contain spaces, enclose the names with double quotation marks preceded by backslashes (\"). Both *app-name* and *file-name* must include their respective paths, either from the hard disk or relative to `$SMBASE`. For example, the following statement specifies to open the MS Word file `Summary95`:

```
sm_shell("\"hd:Apps:MS Word\" hd:Status:Summary95",0);
```

sm_message_box

The library function `sm_message_box` displays a message in a dialog box. Refer to page 52 for information about the function and dialog box modality on the Macintosh.

The following JAM dynamic library functions are available on the Macintosh power PC system, and work as described for Windows:

- `sm_slib_error`
- `sm_slib_install` (refer to page 58)
- `sm_slib_load`

Note: *The dynamic library functions are intended for use only in a power PC system. Currently, JAM for the Macintosh does not support dynamic libraries for 68K systems.*

Using JAM Utilities

The Utilities folder contains a number of JAM utilities, such as `f2asc` and `formlib`. To use a utility, double-click on its icon. This invokes a dialog box in which you can specify input and output options and enter the arguments that are required by the utility. For information about these utilities and their respective arguments, refer to the JAM documentation for information about specific utilities.

Alternatively, utilities can be used via scripted AppleEvents. Refer to page 45 for information on using AppleEvents.

If the utility arguments accept wildcards, you can also use these wildcard characters on the Macintosh:

- * — Match any sequence of characters.
- ? — Match any single character.

Using the Stealth Utility

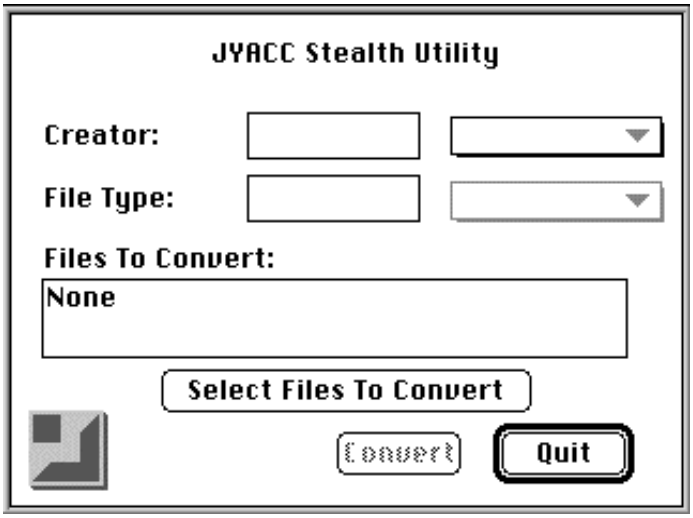
The Stealth utility allows you to specify a new creator ID and file type for files in the directory from which Stealth is launched. This utility converts all non-application files that are in the same folder from which Stealth is launched. Thus, when files are imported from other platforms to the Macintosh, they must be placed in the same folder as the Stealth utility. In addition, the utility allows you to batch-process multiple files in a single directory. The Stealth utility facilitates cross-platform portability.

Specifying a New Creator ID and File Type

To specify a new creator ID and file type:

1. Double-click on the Stealth utility icon. Ensure that the utility is launched from the same folder as the file/files that you want to convert.

   The JYACC Stealth utility dialog box appears.



2. Choose a utility from the Creator pop-up list or type the code in the text box. The creator code appears in the text box.
3. Choose a JAM file type from the File Type pop-up list or type the code in the text box. The file type code appears in the text box.
4. Choose the Select Files to Convert button. A dialog box appears that allows you to select files or directories that need to be converted. If you select a directory, then all files in the directory will be converted.
5. Select the file/files for which you want to specify new creator IDs and file types.
6. Choose Convert. The creator and file type codes of the selected file/files will change to match what is specified in the text boxes.
7. Choose Quit to exit the Stealth utility.

Note: *Creator and Type codes must be four characters long.*

Using GraphicConverter

JYACC provides the GraphicConverter utility with JAM 7.01. This utility lets you convert BMP, JPEG, and GIF images to PICT images before using them on the Macintosh. In supporting these file formats, the GraphicConverter utility allows for cross-platform portability, since Windows and Motif also support these file formats. In addition to converting these file formats to PICTs, the GraphicConverter can also batch-process files.

Using Apple Events

JAM supports the following Apple events:

- Executing a JAM control string.
- Entering test mode.
- Executing a JAM utility.
- Redirecting output/errors.
- Changing the current working directory.

Runtime/Editor Events

The AppleScript for executing JAM control strings and entering test mode are runtime/editor events.

Executing Control Strings

The AppleScript syntax for executing a JAM control string is:

`execute` *"control_string"*

You must precede any double quotation mark within the control string with a backslash (\), to differentiate it from the quotation marks that enclose the control string.

Note: *The JPLExecute Apple event is now superseded by the Apple event that executes a control string.*

Entering Test mode

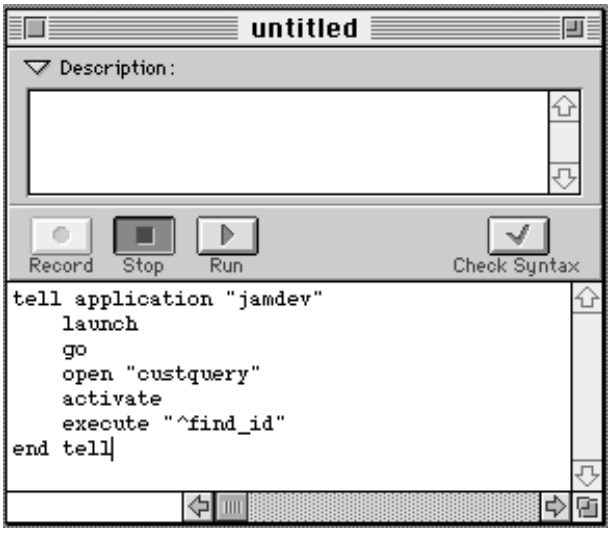
The AppleScript syntax for entering Test mode is:

`go`

Example

The following AppleScript is an example of the Apple events for entering test mode and executing a JAM control string:

```
tell application ″jamdev″
   launch
   go
   open ″custquery″
   activate
   execute ″^find_id″
end tell
```



The following steps are executed:

1. Starts up the JAM development program.
2. Goes into Test mode.
3. Opens the JAM screen named `custquery`.
4. Brings that screen to the foreground.
5. Executes a JPL procedure named `find_id`.

Utility Events

The AppleScript for executing JAM utilities, redirecting output/errors, and changing the current directory are referred to as utility events.

Executing JAM utilities

The AppleScript syntax for executing a JAM utility is:

`ExecuteCmdLine` *"input_string"*

Redirecting output/errors

The AppleScript syntax for redirecting output/errors to a file is:

`SEND input | errors TO` *filename* `[APPENDING true|false]`

Changing the current directory

The AppleScript syntax for changing the current directory is:

`GoTOFolder` *folder_specification*

Example

The following AppleScript shows Apple events for executing JAM utilities, redirecting output/errors, and changing the current directory:

```
tell application "formlib"
    launch
    SEND output to "output.out"
    SEND errors to "errors.out"
    GoTOFolder SMBASE & "Config"
    ExecuteCmdLine "-t jamdev7.lib"
    ExecuteCmdLine "-xf jamdev7.lib smaise"
    ExecuteCmdLine "-xf jamdev7.lib smddtoc"
    ExecuteCmdLine "-cf test.lib"
    ExecuteCmdLine "-r test.lib smaise smddtoc"
    ExecuteCmdLine "-t test.lib"
    quit
end tell
```

The following steps are executed:

1. Starts up the JAM `formlib` utility.
2. Redirects output to a file named `output.out`.
3. Redirects errors to a file named `errors.out`.
4. Goes to the `Config` folder in `SMBASE`.
5. Lists the table of contents (`-t`) of the `jamdev7` form library.
6. Extracts (`-xf`) the screen named `smaise` from the `jamdev7` form library.
7. Extracts the screen named `smddtoc` from the `jamdev7` form library.
8. Creates (`-cf`) an empty form library named `test.lib`.
9. Adds (`-r`) the screens `smaise` and `smddtoc` to the form library `test.lib`.
10. Lists (`-t`) the table of contents for the form library `test.lib`.

Note: *Sample AppleScripts are available with the distribution of JAM 7.01. For example, the SMBASE:Utilities:Utility Runner script is present in the distribution.*

Library Functions Addendum

This appendix includes new or changed descriptions of JAM library functions. Those functions that were rewritten are included here because the changes were extensive; therefore, these descriptions replace those in the JAM 7 documentation.

sm_jfilebox

Opens a file selection dialog box

```
int sm_jfilebox (char *selection, char *path, char *file_mask, char *title,
   int open_save);
```

`selection`	A local or global JPL variable, widget, or property to get the selected file's full pathname.
`path`	The initial path for the directory tree. If you supply an empty string, the dialog box initially shows the directory in which the JAM application was launched.
`file_mask`	A filter to narrow down the display of files in `path`. Use at least one wildcard character. For example, to narrow down the display to all files that have the extension doc, supply `"*.doc"` as the argument. To show all files, supply an empty string.
`title`	The text of the dialog box's title. Supply an empty string to suppress title display.
`open_save`	Valid only for Macintosh and Windows, determines the title of the file type option menu; ignored by other platforms. The title is platform-specific; for example, in Windows, `FB_OPEN` sets the title to List Files of Type.

Returns

1	Success: the user chose OK and JAM copied the filename to `buffer`.
0	The user chose Cancel. No text is copied to `selection`.
-1	Failure: A malloc error occurred or the buffer was too small.

Description

`sm_jfilebox` invokes a file selection box that lets users choose a file to open or save a file. On GUI platforms, JAM uses the GUI's standard file selection dialog. The dialog box initially displays the contents of the `path`-specified directory, and lists files that match the wildcard specification in `file_mask`. Users can browse through the directory tree. When the user chooses OK, JAM copies to `selection` the full pathname of the file to open or save.

If you are running an application on Macintosh or Windows, JAM uses the value of `open_save` to change the title of the file type option menu. You specify the option menu's contents through `sm_filetypes`.

Example

```
proc open_save()
vars filename

if @widget(”@current”)–>name == ”save_button”
{
  call sm_jfilebox \
  (”filename”, ”c:\\videobiz”, ””, ”Save File”, FB_SAVE)
  call save_proc(filename)
}
else if @widget(”@current”)–>name == ”new_button”
{
  call sm_jfilebox \
    (”filename”, ”c:\\videobiz”, ”*.doc” ”New File”, FB_OPEN)
  call open_proc(filename)
}
```

See Also

`sm_filebox`, `sm_filetypes`

sm_message_box

Displays a message in a dialog box

```
int sm_message_box(char *text, char *title, unsigned int options, char *icon);
```

`text`	The text of the message. The text can contain format options shown in "Description."
`title`	The title of the dialog box. A null pointer or empty string specifies no title.
`options`	A bit mask that specifies message box display and behavior. Arguments that set different bits can be OR'd together. Table 2 shows the flags that you can set on this mask.
`icon`	Specifies the icon to use in the dialog box. The icon specified here overrides any icon set through `options`. This argument is ignored in character-mode.

Returns

An integer that indicates which button was pushed:

1 `SM_IDOK`: OK
2 `SM_IDCANCEL`: Cancel
3 `SM_IDABORT`: Abort
4 `SM_IDRETRY`: Retry
5 `SM_IDIGNORE`: Ignore
6 `SM_IDYES`: Yes
7 `SM_IDNO`: No
8 `SM_IDHELP`: Help
9 `SM_IDYESALL`: Yes to All
10 `SM_IDOKALL`: OK to All
11 `SM_IDNOALL` No to All

Description

`sm_message_box` creates a dialog box that displays a message and requests the user to select a button. JAM prevents further interaction with the application until the function returns with the user's selection.

The message text is a single string that wraps within the window. The text can contain these % format options:

%K*keyname*

Displays the specified key, where *keyname* is a logical key mnemonic. When JAM displays the message, it replaces *keyname* with the key label string defined for that key in the key translation file. If there is no label, the %K is stripped out and the mnemonic remains. Key mnemonics are defined in `smkeys.h`

%B

Beeps the terminal with `sm_bel` before the message displays. This escape character must precede the message text.

%N

Creates a new line.

You control message box display and behavior by setting one or more flags in Table 2. You can set one flag from each group. Flag settings from different groups can be OR'd together.

Table 2. Message box settings

Flag settings (by group)	Display/Action
Button Combinations	
`SM_MB_OK`	OK
`SM_MB_OKCANCEL`	OK, Cancel
`SM_MB_ABORTRETRYIGNORE`	Abort, Retry, Ignore
`SM_MB_YESNOCANCEL`	Yes, No, Cancel
`SM_MB_YESNO`	Yes, No
`SM_MB_RETRYCANCEL`	Retry, Cancel
`SM_MB_YESALLNOCANCEL`	Yes, Yes to All, No, Cancel
`SM_MB_OKALL`	OK, OK to All
`SM_MB_OKHELP`	OK, Help
`SM_MB_OKCANCELHELP`	OK, Cancel, Help
`SM_MB_ABORTRETRYIGNOREHELP`	Abort, Retry, Ignore, Help
`SM_MB_YESNOCANCELHELP`	Yes, No, Cancel, Help
`SM_MB_YESNOHELP`	Yes, No, Help
`SM_MB_RETRYCANCELHELP`	Retry, Cancel, Help

Flag settings (by group)	Display/Action
SM_MB_YESALLNOALLCANCEL	Yes, Yes to all, No, No to all, Cancel
System Icon Display	
SM_MB_ICONNONE	No icon
SM_MB_ICONSTOP	Stop
SM_MB_ICONQUESTION	Question
SM_MB_ICONWARNING	Warning
SM_MB_ICONINFORMATION	Information
Default Button	
SM_MB_DEFBUTTON*n*	Sets the button in the *n*th position as the default button.
Modality	
SM_MB_APPLMODAL	Confines user interaction to JAM application message box until message is acknowledged, however, user can interact freely with other applications.
SM_MB_SYSTEMMODAL	Confines user interaction with JAM application message box until message is acknowledged.

The following sections describe these settings in more detail.

Button Combinations

User options are controlled through the message box buttons. Table 2 shows the permissible combinations and the constants that set them.

Your message file defines the labels of message box buttons. You can edit this file and modify the label text. For more information on button label text, refer to page 70 in the *Configuration Guide*.

System Icon

You can use the `options` parameter to set a flag for the system icon you want to display in the message window, if any. The actual icon that appears is platform-specific. In character mode, JAM searches in the message file for the tag that corresponds to the specified icon and its associated text; this text appears in front of the title text. For information on modifying message file tags, refer to page 70 in the *Configuration Guide*.

Default Buttons

The `options` parameter can set the default button. The default button is specified by position—for example, you can set the third button as the default. You cannot set the Help button as the default button.

Modality

JAM requires the user to respond to the message before continuing interaction with the application. You can extend this restriction to the entire system, and thereby prevent interaction with other applications, by setting `SM_MB_SYSTEMMODAL` on the `options` parameter. The default modality setting is `SM_MB_APPLMODAL`, which constrains user interaction only within the JAM application.

Note: *Macintosh dialogs are always system-modal—that is, users cannot switch from a dialog to another application. JAM applications running on Macintosh treat both SM_MB_SYSTEMMODAL and SM_MB_APPLMODAL as requests for system modality.*

Example

```
proc clean_exit()
{
   vars btnPush
   btnPush = sm_message_box(”Save changes before exiting?”,\
             ””,SM_MB_YESNOCANCEL|SM_MB_ICONQUESTION,””)

   if (btnPush == SM_IDCANCEL)
   {
       return
   }
   if (btnPush == SM_IDYES)
   {
      call save_changes()
   }
   if (btnPush == SM_IDNO)
   {
      call sm_jclose()
   }

}
```

sm_optmnu_id

Gets the ID of an option menu or combo box

```
int sm_optmnu_id( void );
```

Returns

- An integer handle that uniquely identifies an option menu.
- PR_NULL_OBJID: Unable to identify an option menu.

Description

sm_optmnu_id gets the object ID property of an option menu/combo box that is initialized on pop-up from an external screen (initialization = PV_FILL_AT_POPUP); this function can only be called by the external screen's entry function; otherwise, it returns PR_NULL_OBJID. (For more information about initializing option menu data, refer to page 178 in the *Editors Guide*.)

For example, you might have two option menus that are initialized from the same external screen but require different sets of data. The external screen's entry function can call sm_optmnu_id to get the ID of its caller and thereby determine which database query fetches the required data:

```
/* get the option menu's ID */
vars opt_id
opt_id = sm_optmnu_id()

dbms declare cursor c1
dbms with cursor c1 alias array1

/* query the database according to option menu name */
if @id(opt_id)->name == "ratings_opt"
{
  dbms declare cursor c1 \
       select rating_code from titles \
       group by rating_code order by 1
}
else if @id(opt_id)->name == "genre_opt"
{
  dbms declare cursor c1 \
       select dscr from codes \
       where code_type = 'genre_code' order by 1
}

dbms with cursor c1 execute
dbms close cursor c1
return
```

sm_set_help

Puts a JAM application into help mode

```
void sm_set_help( void );
```

Description

`sm_set_help` puts JAM into help mode. When JAM is in help mode, mouse-clicking on any object in the JAM application invokes the help that is associated with that object. On GUI platforms, help mode changes the mouse pointer shape to the symbol associated with help—for example, on Windows and Motif, a question mark with an arrow. In character mode, the status line displays `Help Mode`.

While JAM is in help mode, the user can click on any application object that can have help associated with it—a screen, toolbar or menu item, or widget. Help on an object is accessible whether or not the object is inactive or focus-protected. Clicking on the screen is equivalent to using the logical key FHLP; clicking on a widget is equivalent to HELP (or field-level help).

JAM exits help mode and restores the mouse pointer shape after a mouse click occurs.

sm_slib_install

Installs a function from a DLL into a JAM application

```
int sm_slib_install(char *fnc_spec, int language, int return_type);
```

`fnc_spec`

A string that includes the name of the function to install and a comma-delimited list of its argument types enclosed in parentheses:

"func-name(*param-list*)*"*

JAM supports string and integer arguments, specified by `s` and `i`, respectively. Specify any combination of strings and integers from zero to five arguments. JAM also supports functions with six integer arguments.

For example, this statement installs the Windows library function FindWindow, which expects two string arguments:

```
err = sm_slib_install
   ( "FindWindow(s,s)", SLIB_PASCAL, SLIB_INTFNC );
```

`language`

Specifies which language calling convention to use when pushing this function's arguments onto the code stack. The convention that you specify must conform to the order in which the function expects to find its arguments stacked. Supply one of these identifiers:

`SLIB_C`

Arguments are pushed onto the stack in left-to-right order.

`SLIB_PASCAL`

Arguments are pushed onto the stack in right-to-left order. Most Windows 3.1/3.11 functions use this convention.

`return_type`

The installed function's return type, specified by one of these arguments:

```
SLIB_INTFNC
SLIB_STRFNC
SLIB_DBLFNC
SLIB_ZROFNC
```

`SLIB_ZROFNC` specifies to ignore the installed function's return value and always to return 0.

Environment

Windows

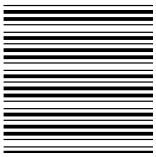
Returns

0	Success.
–1	Cannot find function in the loaded libraries.
–2	Invalid argument.

Description

`sm_slib_install` installs the specified function from a shared library previously installed by `sm_slib_load`. JAM searches for the function in all libraries loaded by `sm_slib_load`, starting with the one most recently loaded. This function is installed as a prototyped function and can be called directly from JPL modules.

Note: *In the Windows distribution, JAM automatically loads the DLLs* KEYBOARD, KERNEL, *and* USER, *in that order. All functions in these libraries are available for installation.*



Index

Symbols

@dmtmp variable, 4

@NULL variable, 5

A

About box, 31
  including version data in, 32
  setting text for, 32

aboutPixmap, 31

Additional tables, 3

ADDM key (add mode), hex value, *19*

Annotations
  hyperlinks, 10
  notes, 10
  on Macintosh, 29

Annotations Manager, 10

appdev directory, *8*

Apple event, 41, 45–47

AppleScripts, samples, 47

Application, traversal properties, 14–15

Application behavior, changing default, in Macintosh, 31

Array Size property, 13

AUTOCOMMIT, for ODBC, 21

B

Binary support, 5

BOFD key (beginning of field), hex value, *19*

BOLN key (beginning of line), hex value, *19*

Border property, under Macintosh, 39

C

CHANGE, transaction manager command, 14

CHARSET, 7

Circular property, 13

class property, 23

clut resource
  color specifications, 32, 33–34
  defined, 30

Color property, for pie chart segments, 2

Colors, defining for Macintosh, 33–34

Combo box widget
  get ID for, 6
  runtime properties, 23

CommandLine, 32

Comments
  and ORACLE optimization hints, 21
  in JPL, 23

Compilers, for Macintosh, 36

config directory, *8*

Context help, 6

Control strings, executed by Apple event, 45

Cursor (database), and ORACLE, 22

D

Data/time format
  and LDB, 14
  for ODBC, 21
  system, 12

Database drivers
  ODBC, 21
  ORACLE, 21
  SYBASE, 7

Decimal places, setting number of, 18

DECIMAL_PLACES, 18

Dialog box
  default behavior on Macintosh, 39
  setting behavior on Macintosh, 33

Directory, changing with Apple event, 47

DLL, installing function from, 58

dm_getdbitext, description, 24

docs directory, 7

drop_down_data property, 23

drop_down_screen property, 23

drop_down_source property, 23

DynaText
  display graphics, 9
  link icons, 10
  search panels, 9

E

editors directory, *8*

EOFD key (end of field), hex value, *19*

EOLN key (end of line), hex value, *19*

ER_ACK_KEY, 19

ER_KEYUSE, 19–20

Error acknowledgement key, 19

Error message, displaying in dialog box, 52

Error processing, 17

Errors, redirecting with Apple event, 46

EXT key (extend selection), hex value, *19*

EXTD key (extend selection down), hex value, *19*

External screen, getting ID of calling option menu, 56

EXTU key (extend selection up), hex value, *19*

F

fastlib, 4

File path syntax, 42
  on Macintosh, 31

File selection dialog box, 6, 50
  opening, 50

Filename
  extension, 24
  search path, 24
  syntax under Macintosh, 41

Find Function search panel, 9

FINISH, transaction manager command, 14

Font, 40
  application default on Macintosh, 33
  for graph widgets, 12
  names on Macintosh, 32
  point size, 40

Font Name property, 12

Frequency property, 12

Frozen Columns property, under Macintosh, 39

Function
  obsolete , 13
  stub, 18

G

getstart directory, *8*

Graph widget, 1–2
  fonts, 12
  on Macintosh, 39

Graphics, displaying in DynaText, 9

Graphics file support
  converting for Macintosh, 44
  on Macintosh, 35

Grid display layout, 2

Grid frame widget
  determining current row, 24
  under Macintosh, 39

grid_current_occ property, 24

H

Help mode, 6, 57

Hints, optimization with ORACLE, 21

Hyperlinks, 10

I

Icon property, under Macintosh, 39

Icons, in DynaText, 10

index directory, *8*

Initialization file, 20

inline view, in DynaText, 9

introPixmap, 32

J

JamConfig file, 30

JamFonts, 33

jmain.c, 30

JPL
  editing, 41
  syntax, 23
  variables on Macintosh, 41

jxmain.c, 30

K

Keyboard, Macintosh equivalents, 38

L

langref directory, 8

LDB
  and date/time fields, 14
  as graph data source, 2

Library, deploying, 4

Line styles, on Macintosh, 40

List box widget, extended selection in, *19*

Logical key
  mapping on Macintosh, 38
  mnemonic and hex values, 19

LP key (local print), 18

LWRD key (left word), hex value, *19*

M

Macintosh
  notes in DynaText, 10
  preferences file, 30–34
  setting defaults, 30

macjkeys, 38

Master (only) format, additional tables, 3

Master–Detail format, additional tables, 3

Master–Detail–Subdetail format, additional tables, 3

Max Occurrences property, 13

max_fetches property, 5

max_rows_per_fetch property, 5

Menu bar editor, using on Macintosh, 40

Menu item
  definitions on Macintosh, 41
  mnemonics on Macintosh, 40

Message dialog box, 52
  button combinations, 53, 54
  default button, 54, 55
  modality setting, 54, 55
  system icon, 54
  text format options, 52

Mouse buttons, 26

N

Note icons, 10

Number of occurrences, runtime property, 4

Number of rows, application property, 5

O

Octal support, 5

ODBC
  Autocommit, 21
  date/time formats, 21

Option menu widget
  get ID of, 6, 56
  identifying to external screen, 56
  runtime properties, 23

ORACLE, array fetching, 5

P

PICT resources
  adding, 37
  editing, 35

Pie chart, 2

placement property, 24

Point size, 40

Pop–up menus, on Macintosh, 39

Portability
  of graphics files on Macintosh, 44
  on Macintosh, 43–44

PostScript files
  in 7.01 distribution, 7
  table of contents, 8
  uncompress, 8

Primary keys, checking in JDB, 4

Properties, setting at runtime, using Apple event, 41

Property values
  via functions, 15
  via property API, 17
  via transaction manager variables, 16

Proximity search panel, 9

ps directory, 7

R

readme directory, 8

Redirecting output, with Apple event, 46

ResEdit, 36, 38

root property, 24

RWRD key (right word), hex value, *19*

S

Sample Function search panel, 9

Screen, background color, on Macintosh, 32

Screen wizard
  and widget types, 2
  Delete button, 11

Scroll Increment property, 13

Search panels, 9

Search path, screen, 24

Selection screen, 4

Server link type, synchronized scrolling, 12

SimpleText, 41

Single–row layout, 2

sm_calc, arguments, 25

sm_filebox, example, 25

sm_fio_open, description/return values, 25

sm_jfilebox, 50–51

sm_message_box, 52–55

sm_mncrinit, 26

sm_ms_inquire, 26

sm_optmnu_id, 56

sm_prop_get, 26

sm_prop_set, 26

sm_set_help, 57

sm_shell, syntax under Macintosh, 42

sm_slib_install, 58–59

sm_strip_amt_ptr, argument/return values, 27

SMBASE, setting on Macintosh, 34

SMLPRINT, 18

Sort Widgets property, 11

Splash screen, on Macintosh, 32

START, transaction manager command, 14

Stealth utility, 43–44

Stored procedures, and ORACLE, 22

STR resource, 31
  defined, 30

Stub functions, 18

SYBASE, 7

Sync Group properties, 13

Synchronized arrays, 12–13

System date/time, 12

System Menu property, under Macintosh, 39

T

Table of contents
  PostScript files, 8
  printing in DynaText, 9

Table views
  determining root, 17
  synchronized scrolling, 12

Test mode, accessing with Apple event, 45

Title Bar property, under Macintosh, 39

toc print option, 9

Transaction manager, variables, 16

Transitive links, 3–4

Traversal properties, application–level, 14–15

TrueType, on Macintosh, 33

U

uncompress utility, 8

update directory, *8*

Utilities
  executing via Apple event, 46
  GraphicConverter, 44
  ResEdit, 36
  runtime libraries, 4
  Stealth (on Macintosh), 43–44
  using on Macintosh, 43

V

Validation link, SQL generation for, 4

Version resource, on Macintosh, 32

W

Widget runtime properties, 23

Widget type, used in screen wizard output, 2

Widget Type property, changing at runtime, 13

Windows initialization file, specifying, 20
WinHelp option, 20

X

XY plot, 2