Environment Variables and System Options

Version 9.3

Printed: December 2009

DCN EV-01-9301

Synergex
2330 Gold Meadow Way
Gold River, CA 95670 USA

http://www.synergex.com
phone 916.635.7300
fax 916.635.6549

Contents

Preface

About this manual

This manual lists all of the environment variables and system options available in Synergy/DE™.

Manual conventions

Throughout this manual, we use the following conventions:

- In code syntax, text that you type is in `Courier` typeface. Variables that either represent or should be replaced with specific data are in *italic* type.
- Optional arguments are enclosed in *[*italic square brackets*]*. If an argument is omitted and the comma is outside the brackets, a comma must be used as a placeholder, unless the omitted argument is the last argument in a subroutine. If the comma is inside the brackets and an argument is omitted, the comma may also be omitted.
- Arguments that can be repeated one or more times are followed by an ellipsis…
- A vertical bar (|) in syntax means to choose between the arguments on each side of the bar.
- Data types are **boldface**. The data type in parentheses at the end of an argument description (for example, (**n**)) documents how the argument will be treated within the routine. An **a** represents alpha, a **d** represents decimal or implied-decimal, an **i** represents integer, and an **n** represents numeric (which means the type can be **d** or **i**).

WIN

Items or discussions that pertain only to a specific operating system or environment are called out with the name of the operating system.

Other useful publications

- Synergy Language release notes (**REL_DBL.TXT**)
- *Synergy Language Reference Manual*
- *Getting Started with Synergy/DE*
- *Professional Series Portability Guide*

Product support information

If you cannot find the information you need in this manual or in the publications listed above, you can call the Synergy/DE Developer Support department at (800) 366-3472 (in North America) or (916) 635-7300. To purchase Synergy/DE Developer Support services, contact your Synergy/DE account manager at the above phone numbers.

Before you contact us, make sure you have the following information:

- The version of the Synergy/DE product(s) you are running.
- The name and version of the operating system you are running.
- The hardware platform you are using.
- The error mnemonic and any associated error text (if you need help with a Synergy/DE error).
- The statement at which the error occurred.
- The exact steps that preceded the problem.
- What changed (for example, code, data, hardware) before this problem occurred.
- Whether the problem happens every time, and whether it is reproducible in a small test program.
- Whether your program terminates with a traceback, or whether you are trapping and interpreting the error.

Synergex Professional Services Group

If you would like assistance implementing new technology or would like to bring in additional experienced resources to complete a project or customize a solution, Synergex™ Professional Services Group (PSG) can help. PSG provides comprehensive technical training and consulting services to help you take advantage of Synergex's current and emerging technologies. For information and pricing, contact your Synergy/DE account manager at (800) 366-3472 (in North America) or (916) 635-7300.

Comments and suggestions

We welcome your comments and suggestions for improving this manual. Send your comments, suggestions, and queries, as well as any errors or omissions you've discovered, to **doc@synergex.com**.

1

Environment Variables

Setting Environment Variables and Initialization Settings

To more easily migrate between Synergy Language environments and to reduce the need for source-code modification, you can use environment variables and initialization settings to externally control program functionality.

You can set environment variables by

- entering the appropriate commands at the operating system prompt.
- placing the commands in your log-in file.
- using the SETLOG routine. (See SETLOG in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual* for more information.)
- setting them as initialization settings in an initialization file on Windows. (See "Synergy initialization files" on page 1-11.)

You can obtain the value of most environment variables using the GETLOG routine. (See GETLOG in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual* for information.) For a list of exceptions on Windows, see page 1-12.

We recommend that you place the environment variable commands in your log-in file, so that your development environment is set up automatically each time you log in. The log-in file you use and the commands you enter depend on what operating system you are running.

Follow the instructions below to set up the environment variables you need for your system.

Settings on Windows

You can set environment variables in one of three ways:

- At the command prompt using the **set** command. Environment variables set this way only affect executables run from the command prompt window that processed the **set** command.
- From the System Properties dialog, which can be accessed using the System icon from Control Panel (Control Panel > System > Advanced > Environment Variables button) or through My Computer (right-click My Computer > Properties). Environment variables set this way affect executables run from the menu or any command prompt started after the change. See your Windows documentation for more information. (This method sets the environment variable in the global environment.)
- Include them in a Synergy™ initialization file, as described in "Synergy initialization files" below. Environment variables set in Synergy initialization files affect the Synergy runtime, Synergy/DE development tools, and Synergy applications.

Any environment variables set for use by *xf*Server or *xf*ServerPlus must be set in the Windows registry. You can set environment variables in the registry using the Synergy Configuration Program.

Synergy initialization files

Synergy.ini and **synuser.ini** are the initialization files that contain environment variables affecting the Synergy runtime, Synergy/DE development tools, and Synergy applications on Windows. The **synergy.ini** file contains system- or application-specific settings, for multiple users. The **synuser.ini** file contains user-specific settings (for example, personal preferences such as colors, fonts, the state of the application window, the position and size of the print preview window, and any overrides to system-specific settings).

When a system administrator or application provider changes a setting in the system initialization file (**synergy.ini**), all users get the new setting unless they have overridden the setting in their user file (**synuser.ini**). When a user modifies a setting in his or her **synuser.ini** file, the change only applies to that user. Changes to the **synuser.ini** file do not affect the general behavior of the application or the settings of any other user.

Variables in **synergy.ini** and **synuser.ini** can be set for individual Synergy/DE tools (**dbl**, **dbr**, **dblink**, **dblibr**) and can be changed at any time. With the exception of a few special startup parameters, the settings are in effect for the duration of the executable that loaded them, but after the executable has finished running, the original environment variable settings once again take effect. Variables in these files can also be set for individual Synergy Language programs (**.dbr**). However, they are only interpreted for the first **.dbr** program executed by **dbr.exe**, and then they stay in effect for all subsequent programs that are chained to. The initialization files are not reread for the chained-to programs.

The service runtimes, **dbs** and **dbssvc**, only look in the environment for environment variables, with one exception. If SFWINIPATH is set in the environment at startup, both **dbs.exe** and **dbssvc.exe** will read the **synergy.ini** file.

Most environment variables listed in this chapter can be set in a Synergy initialization file. Refer to the documentation for each environment variable to determine whether or not it can be set there. Note that on the individual environment variable pages, we only list **synergy.ini** (and not **synuser.ini**) as a setting location. This is because we do not recommend that you manually edit the **synuser.ini** file, but instead let it be created or updated by the Synergy/DE tools or programmatically by your Synergy application. However, any environment variable that can be set in **synergy.ini** can also be set in **synuser.ini**.

On Windows, all settings in the [synergy], [dbr], and [*myprog*] sections of **synergy.ini** and **synuser.ini** are put into the environment, with the exception of APP_HEIGHT, APP_WIDTH, APP_STATE, APP_X, APP_Y, CWD, DBG_HEIGHT, DBG_WIDTH, DBG_X, and DBG_Y. Settings in the [colors] or [fonts] section, or in any section that you add (with the exception of [*myprog*]), are *not* set in the environment.

Although you can set PATH in the **synergy.ini** file, this file is only recognized after the executables have started. PATH should be set with the System icon from Control Panel.

Creating synergy.ini and synuser.ini

The installation program places a default **synergy.ini** file in the synergyde\dbl directory when you install Synergy/DE. If the environment variable SFWINIPATH is not set, or if the **synergy.ini** file referenced by SFWINIPATH cannot be accessed, the **synergy.ini** file in synergyde\dbl will be used. However, since this is just a default file that will be removed during an upgrade or uninstallation, we recommend that you copy this file elsewhere and specify the path to the copied file with SFWINIPATH. See SFWINIPATH on page 1-190 for more information.

The **synuser.ini** file is created the first time an entry is written to it. Synergy/DE tools write to **synuser.ini** at runtime when they save your user settings. The **synuser.ini** file is created in the Synergex subdirectory of your local application data directory (Documents and Settings\username\Local Settings\Application Data).

To determine the exact location of the **synergy.ini** and **synuser.ini** files being used by Synergy, you can run the **synckini** utility in the dbl\bin directory. See "The synckini Utility" in the "General Utilities" chapter of *Synergy Language Tools*.

Organization of synergy.ini and synuser.ini

Initialization files are organized into one or more sections, with section headings enclosed in square brackets and beginning in the left-most column of the file. For example:

```
[synergy]
```

Settings in initialization files use the following format:

ini_variable=*value*

For example, a **synergy.ini** file containing the DBLDIR, ISAMC_REV, and SHELL initialization settings might look like this:

```
[synergy]
DBLDIR=c:\synwin\dbl
ISAMC_REV=3
SHELL=c:\dos\command.com
```

Different initialization file sections affect different things:

Initialization settings in	Affect
`[synergy]`	All Synergy/DE development tools and the runtime
`[dbr]`	Only the runtime
[*myprog*] (*myprog* is the name of any **.dbr** file)	Only the runtime when ***myprog*.dbr** is run
`[dbl]`	Only the compiler
`[dblink]`	Only the linker
`[dblibr]`	Only the librarian
`[listdbo]`	Only the **listdbo** utility
`[listelb]`	Only the **listelb** utility
`[listdbr]`	Only the **listdbr** utility
`[fonts]`	Fonts in Synergy/DE executables
`[colors]`	Colors in Synergy/DE executables
`[fconvert]`	Only the **fconvert** utility
`[isutl]`	Only the **isutl** utility

Priority of synergy.ini and synuser.ini settings

On all Windows systems, initialization settings included in the Synergy initialization files override any previous settings (such as those set in the system environment at startup). For example, if you set DBLDIR and ISAMC_REV in the environment, and your **synergy.ini** file contains a setting for DBLDIR but has no setting for ISAMC_REV, Synergy/DE products will use the DBLDIR setting in **synergy.ini** and the ISAMC_REV setting specified in the environment.

The two initialization files themselves are read according to the following hierarchy:

1. [synergy] section of **synergy.ini**
2. [synergy] section of **synuser.ini**
3. [dbr] section of **synergy.ini** (also [dbl], [dblink], and [dblibr])
4. [dbr] section of **synuser.ini** (also [dbl], [dblink], and [dblibr])
5. [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file)
6. [*myprog*] section of **synuser.ini** (where *myprog* is the same **.dbr** file)

7. Other sections of the initialization files, such as [fonts], [colors], etc., following the same precedence (**synergy.ini** is read first, followed by **synuser.ini**)

The last specification overrides the first in any conflict. For example, if the same font or color is defined in both files, the specification in the **synuser.ini** file prevails. If a font or color is defined in only one of the files, the definitions are additive.

The following table illustrates the hierarchy of interpreting environment variables on Windows.

Global environment[a]	synergy.ini	synuser.ini	Final result
DBLDIR=c:\synergyde\dbl	DBLDIR=c:\mysynergy		DBLDIR=c:\mysynergy
	[fonts] DEFAULT=Fixedsys;9;A	[fonts] DEFAULT=Arial;10;A	[fonts] DEFAULT=Arial;10;A
ISAMC_REV=3			ISAMC_REV=3

a. Setting environment variables in the global environment means setting them from the System icon in Control Panel.

Modifying the synergy.ini file

You can add additional initialization settings or change existing settings in your **synergy.ini** file by editing the file using any text editor.

If you use a word processing program or any other program that formats text, the formatting characters may make the **synergy.ini** file unreadable to Synergy/DE. Be sure to save the changed file as a text file, without formatting information.

Alternately, you can use the **synckini** utility which prompts you to edit the **synergy.ini** file in the registered editor for **.ini** files (Notepad by default). See "The synckini Utility" in the "General Utilities" chapter of *Synergy Language Tools*.

Modifying the synuser.ini file

Although you can locate and manually edit your **synuser.ini** file, we do not recommend it. The **synuser.ini** file is updated programmatically by a few Synergy/DE tools and can be updated programmatically by your Synergy application. For example, Composer updates the [fonts] and [colors] sections of **synuser.ini**. The Synergy Windows printing API updates the **synuser.ini** file with the print preview environment variables (for example, PRINT_PREVIEW_ZOOM). A number of UI Toolkit routines give you the option of updating the **synuser.ini** file rather than **synergy.ini** (for example, U_EDITREND, U_SAVELOG, U_SAVESETTINGS, and %U_WNDFONT).

Settings on UNIX

On UNIX systems, you can place environment variables in your log-in file. The log-in file for UNIX systems is **.profile** if you are using Bourne shell or **.login** if you are using C Shell. After setting an environment variable on UNIX, you must export it unless you have the "auto-export" feature turned on in your shell. (Refer to your UNIX reference manual for details on auto-export. Not all UNIX systems offer this option.) For example:

```
DBLDIR=/usr/synergy/dbl ;export DBLDIR
```

If any file specifications in your programs or command files contain directory specifications, you should define these names as search list environment variables. These assignments are case sensitive, and spaces are significant. (See DBLCASE on page 1-57.) For example:

Bourne Shell:

```
SRC=/usr/mine/source,/usr/progs
OBJ=/usr/mine/object
export SRC OBJ
```

C Shell:

```
setenv SRC /usr/mine/source,/usr/progs
setenv OBJ /usr/mine/objeclt
```

You'll find more information about search list environment variables in "Synergy Language Files" in the "Welcome to Synergy Language" chapter of the *Synergy Language Reference Manual*.

Settings on OpenVMS

OpenVMS doesn't use the term "environment variables." You should substitute a logical name anywhere the Synergy/DE manuals use the term "environment variables." On OpenVMS systems, you can place logicals in your log-in file, **login.com**, or the system-wide log-in file located in SYS$MANAGER.

If any file specifications in your programs or command files contain directory specifications, you should define logical names to reference those locations, and use the logical name in the program or command file.

The OpenVMS operating system has the concept of search list logicals. These are logical names that have multiple translation values. When you specify a search list logical in an OPEN statement, the operating system scans all the locations listed to find the named file.

Some of the environment variables described in the following pages require a list of comma-separated values. These are *not* search lists, they are just comma-delimited values used and parsed by the compiler and runtime. When we use the term "search list" in the following pages, it means a comma-delimited value. Use quotation marks to prevent commas from creating multiple translation values.

To define a search list logical, use the following format:

```
$ DEFINE DATA_LOCATION DKA100:[DATA],DKA200:[BACKUP_DATA]
```

To define a comma-delimited value, use the following format:

```
$ DEFINE DBLOPT "7,35,43"
```

The Environment Variables and Initialization Settings

This section describes the environment variables and initialization settings recognized by Synergy/DE.

The procedure for setting an environment variable is shown on page 1-10 for Windows, page 1-15 for UNIX, and page 1-15 for OpenVMS. An example is given for each environment variable in the following sections, but not for all operating systems. Refer to the pages listed above as needed.

Each environment variable and initialization setting specification page includes the Synergy products that are affected by that setting. “Runtime” specifies anything that requires the runtime to run, which includes UI Toolkit, Composer, Repository, ReportWriter, and so forth. “UI Toolkit” specifies anything that uses UI Toolkit, which includes Composer, Repository, ReportWriter, and so forth.

If the Setting location section for an environment variable or initialization setting lists the environment, you can use the GETLOG routine to obtain that variable’s value.

ACTIVEX_LIST – Use ActiveX list control by default

WIN

The ACTIVEX_LIST environment variable specifies which list, ActiveX or UI Toolkit, should be used by default for any lists that are processed.

Value

One of the following flags:

0	Use the Toolkit list. (default)
1	Use the ActiveX list control.

Discussion

If ACTIVEX_LIST is set to 1, the global **g_activex_list** is set to true within U_START. For any list class that does not specify which list (ActiveX or UI Toolkit) to use by specifying the ACTIVEX qualifier on the .LISTCLASS script command, **g_activex_list** is used to determine which list is used.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

.LISTCLASS in the "Script" chapter of the *UI Toolkit Reference Manual* for more information on overloading the global list default on a per list basis.

Examples

In the **synergy.ini** file,

```
[synergy]
ACTIVEX_LIST=1
```

ALT_FONT_HEIGHT – Alternate font height

WIN

The ALT_FONT_HEIGHT environment variable sets the height of the alternate font (in conjunction with ALT_TYPE_FACE and ALT_FONT_WIDTH) if an alternate font is not otherwise specified.

Value

The height, in logical units, of the font.

Discussion

Synergy/DE on Windows uses the specified font height for application windows greater than or equal to 132 columns (if automatic font switching is enabled).

The font height can be specified in one of three ways:

- If the height is greater than 0, it is transformed into device units and matched against the cell height of the available fonts.
- If it is 0, a reasonable default size is used.
- If it is less than 0, it is transformed into device units, and the absolute value is matched against the character height of the available fonts.

If the current font is not available in the requested size, Synergy/DE on Windows substitutes the font that most closely resembles the specified font.

We recommend that you use the FONT_ALTERNATE environment variable rather than ALT_FONT_HEIGHT.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the alternate font.
- %U_WNDFONT in the "Utility Routines" chapter of the *UI Toolkit Reference Manual* for information on automatic font switching.
- FONT_ALTERNATE on page 1-88.

Examples

In the **synergy.ini** file,

```
[synergy]
ALT_FONT_HEIGHT=-18
```

ALT_FONT_WIDTH – Alternate font width

WIN

The ALT_FONT_WIDTH environment variable sets the width of the alternate font (in conjunction with ALT_TYPE_FACE and ALT_FONT_HEIGHT) if an alternate font is not otherwise specified.

Value

The average width, in logical units, of characters in the font.

Discussion

Synergy/DE on Windows uses the specified font width for application windows greater than or equal to 132 columns (if automatic font switching is enabled).

If the width is 0, the aspect ratio of the device is matched against the digitization aspect ratio of the available fonts to find the closest match, determined by the absolute value of the difference.

If the current font is not available in the requested size, Synergy/DE on Windows substitutes the font that most closely resembles the specified font.

We recommend that you use the FONT_ALTERNATE environment variable rather than ALT_FONT_WIDTH.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the alternate font.
- %U_WNDFONT in the "Utility Routines" chapter of the *UI Toolkit Reference Manual* for information on automatic font switching.
- FONT_ALTERNATE on page 1-88.

Examples

In the **synergy.ini** file,

```
[synergy]
ALT_FONT_WIDTH=0
```

ALT_TYPE_FACE – Alternate font

WIN

The ALT_TYPE_FACE environment variable sets the typeface of the alternate font if an alternate font is not otherwise specified.

Value

The name of the desired Windows font (or typeface).

Discussion

Synergy/DE on Windows uses the specified typeface for application windows greater than or equal to 132 columns (if automatic font switching is enabled).

If you want to specify a new font using ALT_TYPE_FACE, it should be a font that comes in standard Windows packages.

You may want to use a fixed font, because the character columns will always have the same alignment from row to row (as they would on a VT-100 or other text terminal), and text positioning will remain consistently aligned. We recommend that you use a fixed font when doing non-Toolkit processing.

Another style consideration when choosing a typeface is whether you want a serif or a sans-serif font. Serifs are the little strokes (or "feet") at the ends of a letter's main strokes. A sans-serif style does not have these ending strokes. Serif typefaces are generally recommended for larger bodies of text because the serifs on each letter actually help guide the eyes. Sans-serif typefaces work well for short phrases, headings, and small amounts of text.

This is an example of a sans-serif typeface.

This is an example of a serif typeface.

To find a list of available typefaces, open the Character Map utility (in the Accessories group). The typeface names are case sensitive.

We recommend that you use the FONT_ALTERNATE environment variable rather than ALT_TYPE_FACE.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the alternate font.
- %U_WNDFONT in the "Utility Routines" chapter of the *UI Toolkit Reference Manual* for information on automatic font switching.
- FONT_ALTERNATE on page 1-88.

Examples

In the **synergy.ini** file,

```
[synergy]
ALT_TYPE_FACE=Terminal
```

ANSICOLOR – Use built-in runtime ANSI color sequences

UNIX

The ANSICOLOR environment variable uses the ANSI color sequences built in to the Synergy runtime to generate color.

Value

Any value.

Discussion

You can use ANSICOLOR to display color on devices that support ANSI color. (Synergy does not support color from **terminfo**; however, ANSICOLOR enables your application to use ANSI color regardless of the value of TERM.) You do not need to use ANSICOLOR if your terminal is a VT100-series terminal (VT100, VT220, etc.), if TERM is set to **ansi** or **xterm**, or if you've already added the Synergy color codes to the **termcap** database and intend to run the **termcap** runtime.

Setting location

The environment.

Used by

Runtime.

See also

- "Colors and the color palette" in the "Synergy Windowing API" chapter of the *Synergy Language Reference Manual*.
- "Synergy Language and the UNIX Terminal Database" in the "Terminal Characteristics on UNIX" chapter of the *Professional Series Portability Guide*.
- "Enabling color" in the "Other UNIX-Specific Information" chapter of the *Professional Series Portability Guide*.

Examples

```
ANSICOLOR=1     ;export ANSICOLOR
```

APP_HEIGHT – Initial application window height

WIN

The APP_HEIGHT environment variable sets the initial height of your application window.

Value

The initial height for the application's container window, specified in character cells.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
APP_HEIGHT=25
```

APP_STATE – Initial application window state

WIN

The APP_STATE environment variable sets the initial state of your application window.

Value

The initial state of the application's container window:

NORMAL	Window is normal.
MAXIMIZED	Window is maximized.
MINIMIZED	Window is minimized.
HIDDEN	Window is hidden.

Discussion

If no initial state is specified, or if the state specified is not one of the above states, the normal state is assumed.

APP_STATE has precedence over the XSHOW environment variable. Unlike XSHOW=1, the Toolkit U_START subroutine does not override the initial state specified by APP_STATE.

Setting APP_STATE to HIDDEN has the same effect as setting the environment variable XSHOW to HIDE.

The behavior of APP_STATE is dependent on setting location. If set in the environment, APP_STATE affects the current program and any programs spawned by the current program. If set in **synergy.ini**, APP_STATE affects only the current program.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
APP_STATE=MAXIMIZED
```

APP_WIDTH – Initial application window width

WIN

The APP_WIDTH environment variable sets the initial width of your application window.

Value

The initial width of the application's container window, specified in character cells.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
APP_WIDTH=80
```

APP_X – Initial horizontal position of window

WIN

The APP_X environment variable sets the initial horizontal position of your application window.

Value

The initial horizontal position of the upper-left corner of the application's container window, specified in pixels offset from the upper-left corner of the Windows desktop.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
APP_X=100
```

APP_Y – Initial vertical position of window

WIN

The APP_Y environment variable sets the initial vertical position of your application window.

Value

The initial vertical position of the upper-left corner of the application's container window, specified in pixels offset from the upper-left corner of the Windows desktop.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
APP_Y=100
```

AXDEBUG – Enable debugging system for ActiveX API

WIN

AXDEBUG lets you analyze problems in programs using the Synergy ActiveX API by enabling a debugging system.

Value

One of the following values:

YES	Enable debugging system and log events to a window.
FILE=*filename*	Enable debugging system and log events to the specified file.
NO	Disable debugging system. (default)
EXTERNAL	Enable debugging system and send debug information to an attached Windows debugger.

Discussion

On 32-bit Windows, when AXDEBUG is set to YES, the creation of the first ActiveX container creates a debug display, which consists of a tree view of the Synergy Language/ActiveX system and a log of the related operations. On 64-bit Windows, YES is equivalent to EXTERNAL.

The tree view of ActiveX objects (in the left pane of debug display system when the YES option is used) is not shown for either the FILE=*filename* or the EXTERNAL option. The FILE and EXTERNAL options only log debug events.

When AXDEBUG is set to FILE=*filename*, events are logged to a file instead of a window, where *filename* is the name of the file to which output is appended.

When AXDEBUG is set to EXTERNAL, all debug logging information goes to an attached Windows debugger instead of the debug display. (On 64-bit Windows, we recommend an external debugger such as the debugview program from http://www.sysinternals.com.)

See “Debugging” in the “Synergy ActiveX API” chapter of the *Synergy Language Reference Manual* for more information about the debugging system.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

If set at the command prompt,

```
set AXDEBUG=FILE=debug.log
```

BADLOCKWAIT – Timeout for conventional lock

UNIX

The BADLOCKWAIT environment variable sets the amount of time that Synergy waits for a conventional lock during an OPEN statement.

Value

The maximum number of seconds to wait before signaling a DE_BADLCK “Lock failure” error. The default value is 5.

Setting location

The environment.

Used by

Runtime, compiler, **fcompare**, **fconvert**, **isutl**.

Examples

```
BADLOCKWAIT=7     ;export BADLOCKWAIT
```

BEGPORT – Beginning of a range of ports to check for *xf*Server

With ENDPORT, the BEGPORT environment variable sets the range of ports that **synxfpng** will check for a running *xf*Server.

Value

The number of the first port in the range to check. The default value is 2330.

Setting location

The environment.

Used by

synxfpng.

See also

- ENDPORT on page 1-84.
- "The synxfpng Utility" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Examples

Using the examples below, **synxfpng** checks ports 2360 – 2370 for a running *xf*Server.

```
BEGPORT=2360
ENDPORT=2370
```

CACHE_STAT – Enable ISAM cache statistics

WIN, UNIX

The CACHE_STAT environment variable turns on the display of Synergy ISAM cache statistics.

Value

Any value.

Discussion

If set to any value, cache statistics are displayed when an ISAM file that was opened with cache enabled is closed. This is for ISAM file access only.

Setting location

The environment. On Windows, if this environment variable is being used by the runtime, it can also be set in the [synergy] or [dbr] section of **synergy.ini**.

Used by

Runtime, compiler, **fcompare**, **fconvert**, **isutl**.

Examples

On Windows, if set at the command prompt,

```
set CACHE_STAT=1
```

CMPBSIZ – Compiler output buffer size

WIN, UNIX

The CMPBSIZ environment variable sets the size of output buffers for the Synergy compiler.

Value

The size of the output buffer in bytes.

Discussion

The default size is 8192 bytes.

The maximum output buffer size is limited by the memory available on your system. A larger output buffer size improves compiler speed, but requires more memory.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dbl] section of **synergy.ini**.

Used by

Compiler.

Examples

On UNIX,

```
CMPBSIZ=16384       ;export CMPBSIZ
```

COLOR*n* – Possible colors

WIN

The COLOR0 through COLOR255 initialization settings can be used to override the default settings for the 256 user colors (color 0 through color 255).

Value

COLOR*n* is specified as follows:

COLOR*n*=*rgb*

where *n* is a number from 0 to 255, representing one of 256 possible colors, and *rgb* is an RGB triplet that defines the color. This triplet must be separated by commas and can be in either hexadecimal or decimal notation. "0x" designates hexadecimal notation.

Discussion

COLOR*n* enables you to define a Synergy user color in **synergy.ini**. COLOR*n* settings override default user colors that are loaded by the Synergy runtime when it starts. Note the following:

- COLOR*n* must be in the [colors] section of **synergy.ini**. If no [colors] section exists, add "[colors]" on a separate line at the end of the file and then add COLOR*n* settings on separate subsequent lines.
- Synergy/DE supports 512 colors, but COLOR*n* settings can be used only for user colors (0 through 255). Specifying a number higher than 255 or lower than 0 will generate an error.

For more information on user colors, see "Colors and the color palette" in the "Synergy Windowing API" chapter of the *Synergy Language Reference Manual*.

Setting location

The [colors] section of **synergy.ini**.

Used by

Runtime.

See also

"Customizing the Look of Your Application" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual*.

Examples

In the **synergy.ini** file,

```
[colors]
COLOR7=0xFF,0xFF,0xFF
```

COMBUF – Input buffer size for COM port

WIN

COMBUF enables you to change the input ring buffer size for serial port communication.

Value

The size of the input buffer in bytes.

Discussion

COMBUF defines the size of the input buffer when a COM PORT is opened. The default size is 4096 bytes.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

If set at the command prompt,

```
set COMBUF=8192
```

If set in the **synergy.ini** file,

```
[synergy]
COMBUF=8192
```

COMSPEC – Shell program to run

WIN

If the SHELL subroutine is called with the second argument set to zero, COMSPEC specifies the name of the shell program to run.

Value

The name of the shell program to run.

Discussion

The default is **cmd.exe**.

If the SHELL environment variable is set, it overrides COMSPEC.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

If set at the command prompt,

```
set COMSPEC=c:\dos\command /c
```

If set in the **synergy.ini** file,

```
[synergy]
COMSPEC=c:\dos\command /c
```

CONNECTDIR – synergyde\connect directory

The CONNECTDIR environment variable specifies the most recently installed synergyde\connect directory.

Value

The connect subdirectory of the synergyde directory.

Discussion

CONNECTDIR is set by the Connectivity Series installation on Windows, or it's set when you run **setsde** on UNIX or **SYS$MANAGER:SYNERGY_STARTUP.COM** on OpenVMS. You should not change environment variables that are set by the system.

Use CONNECTDIR in a .INCLUDE statement to specify the location for SQL Connection programs, as follows:

```
.include "CONNECTDIR:ssql.def"
```

Used by

*xf*ODBC, SQL Connection.

CWD – Working directory

WIN

The CWD initialization setting sets the working directory for your application.

Value

The directory path, including device specification, that will become the current working drive and directory.

Discussion

CWD must be set in the **synergy.ini** file only; setting CWD in the environment has *no* effect. This initialization setting causes the runtime to set its current working directory to the specified path. If CWD is set in the [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file), it takes effect only after the **.dbr** program file is located. For this reason, you cannot use CWD to switch the execution directory of the application's main executable.

Setting location

The [synergy], [dbr], [dbl], [dblink], [dblibr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Compiler, runtime, linker, librarian, **isutl**, **fcompare**.

Examples

In the **synergy.ini** file,

```
[synergy]
CWD=c:\synergy\dbl\apps
```

DBG_BUFFER – Debugger window buffer

WIN

The DBG_BUFFER environment variable specifies the number of lines of display text that the debugger window and the Toolkit debugger window will retain when scrolling.

Value

The number of display lines of debugger window text to retain.

Discussion

The debugger window and the Toolkit debugger window display 25 lines of text at a time (or the value of DBG_HEIGHT), but you can scroll to view additional lines. By default, 300 lines are saved in the display buffer. You can set DBG_BUFFER to override this setting.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Debugger, UI Toolkit debugger.

Examples

```
[synergy]
DBG_BUFFER=400
```

DBG_HEIGHT – Initial debugger window height

WIN

The DBG_HEIGHT environment variable sets the initial height of the Synergy debugger window.

Value

The initial height of the Synergy debugger window, specified in character cells.

Discussion

The default value is 25 lines of text.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
DBG_HEIGHT=30
```

DBG_INIT– Debugger initialization file

The DBG_INIT environment variable specifies a file containing Synergy debugger commands.

Value

The debugger initialization file.

Discussion

The debugger reads the file and executes the commands. The default extension for the initialization file is .**cmd**.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

On OpenVMS,

```
$ DEFINE DBG_INIT MY_INIT.CMD
```

DBG_RMT – Remote debugger startup information

VMS

The DBG_RMT logical indicates that remote debugging is to occur and specifies the port number and optionally the timeout value.

Value

`"-rd` *port[*`:`*timeout]*`"`

where *xxxx* is the port number on which the debug server will listen as a Telnet server for the debug client (1024 to 65535, inclusive) and *timeout* is the number of seconds the debug server will wait for a connection from the debug client (the default is 100). Note that the quotation marks are required.

Setting location

If you specify *timeout*, make sure it is lower than your client connection timeout value.

Used by

Runtime.

See also

The OpenVMS section of "Debugging remotely" in the "Debugging Your Synergy Programs" chapter of *Synergy Language Tools*.

Examples

```
$ DEFINE DBG_RMT "-rd 1024:80"
```

DBG_WIDTH – Initial debugger window width

WIN

The DBG_WIDTH environment variable sets the initial width of the Synergy debugger window.

Value

The initial width of the Synergy debugger window, specified in character cells.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
DBG_WIDTH=80
```

DBG_X – Initial horizontal position of debugger window

WIN

The DBG_X environment variable sets the initial horizontal position of the Synergy debugger window.

Value

The initial horizontal position of the upper-left corner of the Synergy debugger window, specified in pixels offset from the upper-left corner of the Windows desktop.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
DBG_X=100
```

DBG_Y – Initial vertical position of debugger window

WIN

The DBG_Y environment variable sets the initial vertical position of the Synergy debugger window.

Value

The initial vertical position of the upper-left corner of the Synergy debugger window, specified in pixels offset from the upper-left corner of the Windows desktop.

Setting location

The [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
DBG_Y=100
```

DBG$INPUT – Source of debugger input

VMS

DBG$INPUT is the source of debug commands during program execution.

Value

A terminal input device name.

Discussion

The environment variable DBG$INPUT enables Synergy debugger commands to come from a terminal other than the one on which the debugged program is running.

Setting location

The environment.

Used by

Runtime.

Examples

```
$ DEFINE DBG$INPUT TTA0:
```

DBG$OUTPUT – Destination of debugger output

VMS

DBG$OUTPUT is the destination of debug output during program execution.

Value

A terminal device name.

Discussion

The environment variable DBG$OUTPUT enables Synergy debugger output to be directed to a device other than the one on which the debugged program is running.

Setting location

The environment.

Used by

Runtime.

Examples

```
$ DEFINE DBG$OUTPUT TTA0:
```

DBGSRC – Debugger source files

The DBGSRC environment variable sets the default search path for debugger source file lookup.

Value

The directory path, including the device, in which the debugger will find the appropriate source files.

Discussion

You can set DBGSRC either in the environment, before beginning the debugging session, or you can set it during the debugging session using the debugger SET command.

DBGSRC is used only if the debugger can't find the path (including version on OpenVMS) for the file as specified to the compiler.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

On UNIX,

```
DBGSRC=/usr/dblv5/source     ;export DBGSRC
```

DBL_CHRSET – User character set

UNIX, VMS

The DBL_CHRSET environment variable is used by VT-style terminals to specify character sets other than USASCII for the Synergy windowing API.

Value

The escape sequence that designates the character set that should be used for all screen displays:

(B	USASCII
(A	British
(4	Dutch
(Q	French Canadian
(R	French
(K	German
(Y	Italian
(Z	Spanish
(=	Swiss

Discussion

The escape character is not part of the environment variable's definition. It is supplied by the runtime.

Refer to your terminal's programming manual for more details and additional sequences.

Setting location

The environment.

Used by

Runtime.

Examples

On OpenVMS,

```
$ DEFINE DBL_CHRSET "(K"
```

DBL$FATAL_IMAGE – Image to chain to when FATAL has been called

DBL$FATAL_IMAGE (or DBL_FATAL_IMAGE on UNIX) defines the default image to chain to when the FATAL subroutine has been called.

Value

The path of a Synergy Language program.

Discussion

The DBL$FATAL_IMAGE (or DBL_FATAL_IMAGE) environment variable contains the name of a program to be chained to when a fatal error occurs.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

On OpenVMS,

```
$ DEFINE DBL$FATAL_IMAGE DKA0:[USER]TOTAL.EXE
```

DBL$RUNJB_OUTPUT – Output destination for RUNJB

VMS

DBL$RUNJB_OUTPUT defines the output file from the RUNJB subroutine.

Value

The device or filename to which the RUNJB output should be generated.

Discussion

This logical defines the output file for the RUNJB subroutine when called with *io_flag* set to zero.

Setting location

The environment.

Used by

Runtime.

Examples

```
$ DEFINE DBL$RUNJB_OUTPUT OUTPUT.ERR
```

DBLBINDIR – dbl\bin directory

WIN

The DBLBINDIR environment variable specifies the most recently installed dbl\bin directory.

Value

The dbl\bin subdirectory beneath the Synergy/DE directory.

Discussion

DBLBINDIR is set and added to PATH by the Core Components installation and is for internal use only.

Setting location

Global environment. (See “Settings on Windows” on page 1-10.)

Used by

Workbench, Synergy Configuration Program.

DBLBS – ASCII value of the BACKSPACE character

UNIX

DBLBS enables you to change the ASCII value of the BACKSPACE character.

Value

A valid ASCII character.

Discussion

The default value is 8.

Setting location

The environment.

Used by

Runtime.

Examples

```
DBLBS=a     ;export DBLBS
```

DBLCASE – Case translation

WIN, UNIX

The DBLCASE environment variable determines what type of case translation occurs on file specifications in statements such as OPEN, STOP, .INCLUDE, and so forth in the execution of Synergy Language programs and other Synergy/DE development tools.

Value

One of the following codes:

u	Translate logicals and filenames to uppercase.
l	Translate logicals and filenames to lowercase.
u:l	Translate logicals to uppercase; translate filenames to lowercase.
l:u	Translate logicals to lowercase; translate filenames to uppercase.

Discussion

If DBLCASE is not specified, the case remains as it is written; no translation occurs.

We do not recommend setting DBLCASE unless there is a need to do so for your particular system. We also do not recommend using this variable with *xf*ODBC. See "Starting and stopping SQL OpenNet for xfODBC" in the UNIX section of the "Configuring Connectivity Series" chapter of the *Installation Configuration Guide* for information.

Setting location

The environment. On Windows, this environment variable can also be set in the [dbr], [dbl], [dblink], or [dblibr] section of **synergy.ini**.

DBLCASE can be reset by the SETLOG subroutine, and the runtime interprets the new setting.

Used by

Runtime, compiler, linker, librarian, **fcompare**, **fconvert**, **isutl**.

Examples

```
DBLCASE=u:l  ;export DBLCASE
```

Using the example above, the filename specified in the following OPEN statement:

```
open(11, U:I, "MYDIR:MYFILE")
```

is translated as

```
MYDIR:myfile.ism
```

DBLDICTIONARY – Repository files

The DBLDICTIONARY environment variable specifies the full path names of the Repository main and text files.

Value

The file specifications for the Repository main and text files, separated by a comma.

Discussion

The following lists the hierarchy of Repository logicals tested by the compiler:

- If a logical is specified, use it. If it is specified and *not* set, a DDBADLOG error is generated.
- Next, the compiler looks for the DBLDICTIONARY logical and uses it if it's set.
- If the DBLDICTIONARY logical is *not* set, the compiler looks to see if you have specified the RPSMFIL/RPSTFIL logicals. If you have set only the RPSMFIL logical, a DDBADLOG error is generated. If you have set both of these logicals and it is not a version 6 or 7 repository, a DDNOLOG error is generated.
- If the DBLDICTIONARY and RPSMFIL logicals are *not* set, the compiler looks for the ICSMFIL/ICSTFIL logicals. If you have set only the ICSMFIL logical, a DDBADLOG error is generated. If you have set both of these logicals and it is not a version 3 dictionary, a DDNOLOG error is generated.
- If none of the above conditions apply, a DDNOLOG error is generated.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dbl] section of **synergy.ini**.

Used by

Compiler.

See also

RPSMFIL on page 1-155 and RPSTFIL on page 1-156.

Examples

On Windows, if set at the command prompt,

```
set DBLDICTIONARY=RPSDAT:rpsmain.ism,RPSDAT:rpstext.ism
```

On UNIX,

```
DBLDICTIONARY=RPSDAT:rpsmain.ism,RPSDAT:rpstext.ism
export DBLDICTIONARY
```

On OpenVMS,

```
$ DEFINE DBLDICTIONARY "RPSDAT:RPSMAIN.ISM,RPSDAT:RPSTEXT.ISM"
```

DBLDIR – Synergy Language directory

The DBLDIR environment variable tells Synergy which directory contains your Synergy Language distribution. It is required for normal operation of Synergy/DE tools and programs.

Value

The path, including the device, for the directory that contains the Synergy Language distribution files. The path should be the full path specification without any logicals. You can terminate the path with a backslash (\), but it is not necessary.

Discussion

VMS

SYS$COMMON:[SYSMGR]SYNERGY_STARTUP.COM defines the directory variable.

Setting location

- On Windows, the [synergy] section of the distributed **synergy.ini** file installed with Synergy/DE.
- On UNIX, in the environment by the **setsde** script, which you should have run after installing Synergy/DE.
- On OpenVMS, the **SYNERGY_STARTUP.COM** file when Synergy/DE is installed (or **ACTIVATE_SDE.COM** for alternate installations).

You can also set DBLDIR manually in the environment.

Used by

Runtime, **isutl**, *xf*Server, *xf*ServerPlus.

Examples

On Windows, if set in the **synergy.ini** file,

```
[synergy]
DBLDIR=c:\synergyde\dbl\
```

On UNIX,

```
DBLDIR=/usr/synergyde/dbl     ;export DBLDIR
```

On OpenVMS,

```
$ DEFINE/SYSTEM/EXEC DBLDIR SYNERGYDE$ROOT:[DBL]
```

DBLEOF – ASCII value of the EOF character

UNIX

DBLEOF defines the ASCII value of the EOF character.

Value

A valid ASCII character.

Discussion

The default value is 4 (CTRL+D).

Setting location

The environment.

Used by

Runtime, compiler, **fcompare**, **fconvert**, **isutl**.

Examples

```
DBLEOF=d     ;export DBLEOF
```

DBLHIGH64POS – 64-bit file offset for locking

UNIX

Like DBLHIGHPOS, the DBLHIGH64POS environment variable specifies the maximum file offset for locking, except that you can specify a 64-bit position.

Value

The maximum file offset or the following:

WARN	Warn when the lock attempt is rejected.

Discussion

DBLHIGH64POS sets the maximum file offset for locking when system option #33 is not set (conventional locking). By convention, Synergy ISAM files use the highest addressable file offset to lock during OPEN processing for enforcement of exclusive file locking. This value is normally set to 0x7FFFFFFFFFFFFFF on 64-bit machines.

At runtime, a lock attempt is made at the default position. If the lock attempt is rejected at that position, it adjusts down until it succeeds. To be warned of this, set DBLHIGH64POS to “WARN.” If you get a fatal “Record locking problem” error, you can either set system option #33 or use DBLHIGH64POS. We recommend using DBLHIGH64POS so that exclusive file sharing is not disabled.

This environment variable should be set only in extreme circumstances; all users accessing the same data files must also have the same setting, including Synergy/DE *xf*Server.

Setting location

The environment.

Used by

Runtime, **fcompare**, **fconvert**, **isutl**.

Examples

```
DBLHIGH64POS=WARN      ;export DBLHIGH64POS

DBLHIGH64POS=0x7FFFFFFFFFFFFFF
```

DBLHIGHPOS – File offset for locking

UNIX

The DBLHIGHPOS environment variable specifies the maximum file offset for locking.

Value

The maximum file offset or the following:

WARN	Warn when the lock attempt is rejected.

Discussion

DBLHIGHPOS sets the maximum file offset for locking when system option #33 is not set (conventional locking). By convention, Synergy ISAM files use the highest addressable file offset to lock during OPEN processing for enforcement of exclusive file locking. This value is normally set to 0x7FFFFFFF on 32-bit machines.

At runtime, a lock attempt is made at the default position. If the lock attempt is rejected at that position, it adjusts down until it succeeds. To be warned of this, set DBLHIGHPOS to “WARN.” If you get a fatal “Record locking problem” error, you can either set system option #33 or use DBLHIGHPOS. We recommend using DBLHIGHPOS so that exclusive file sharing is not disabled.

⚠ This environment variable should be set only in extreme circumstances; all users accessing the same data files must also have the same setting, including Synergy/DE *xf*Server.

Setting location

The environment.

Used by

Runtime, **fcompare**, **fconvert**, **isutl**.

Examples

```
DBLHIGHPOS=WARN      ;export DBLHIGHPOS

DBLHIGHPOS=0x7FFFFFFE
```

DBLLIBRARY – Library files

The DBLLIBRARY environment variable indicates the search list that specifies the directories where the compiler will search for files specified in .INCLUDE statements that use the LIBRARY option.

Value

The search list logical that specifies one or more directory paths in which to search for the files. See "Search list logicals" in the "Welcome to Synergy Language" chapter of the *Synergy Language Reference Manual* for more information about search lists.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dbl] section of **synergy.ini**.

Used by

Compiler.

Examples

On Windows, if set at the command prompt,

```
set DBLLIBRARY=c:\synergy\dbl\libfiles,c:\usr\mylib
```

DBLMAXERR – Maximum number of errors

The DBLMAXERR environment variable specifies the maximum number of errors that the Synergy compiler will generate before abandoning the compilation with a "Too many errors" error (ERRCNT).

Value

The maximum number of errors the compiler will generate before aborting.

Discussion

The default is 20 errors. This environment variable has no maximum limit although the maximum number of errors displayed ultimately may be limited by available system memory.

If you are compiling with the **-W4** option, you will probably want to set DBLMAXERR to a value higher than the default.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dbl] section of **synergy.ini**.

Used by

Compiler.

Examples

On UNIX,

```
DBLMAXERR=50    ;export DBLMAXERR
```

DBLOPT – Synergy Language system options

The DBLOPT environment variable sets Synergy Language system options.

Value

A string that contains one or more Synergy Language system option numbers, separated by commas. Synergy Language ignores any option numbers that it doesn't recognize or that aren't applicable to the current environment.

Discussion

See chapter 2, "System Options," for a complete list of system options.

VMS

When you define multiple runtime options with DBLOPT, make sure you enclose *option_string* in quotation marks. For example:

```
$ DEFINE DBLOPT "1,7,16,35"
```

If you don't use quotation marks, DCL interprets the logical as a search list, and the runtime only processes the first option specified.

Setting location

The environment. On Windows, if this environment variable is being used by the runtime, it can also be set in the [synergy], [dbl], or [dbr] section of **synergy.ini**.

DBLOPT can be reset by the SETLOG subroutine, and the runtime interprets the new setting.

Used by

Runtime, compiler, linker, librarian, **fcompare**, **fconvert**, **isutl**.

Examples

On Windows, if set at the command prompt,

```
set DBLOPT=11,23,43
```

DBLSTARLET – System services directory

VMS

The DBLSTARLET environment variable specifies the directory of the system include files for use with system services. The default is [SYNERGY.DBL.DBLSTARLET].

Value

The directory that contains the system services include files.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

Compiler.

Examples

```
$ DEFINE/SYS/EXEC DBLSTARLET SYS$SYSDEVICE:[SYNERGY.DBL.DBLSTARLET]
```

DBLTEXT – External subroutine directory

VMS

The DBLTEXT environment variable specifies the directory of the include files that are for use with Synergy Language external subroutines. The default path is [SYNERGY.DBL.DBLSTARLET].

Value

The directory that contains the Synergy Language external subroutines.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER when Synergy/DE is installed (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

Compiler.

Examples

```
$ DEFINE/SYS/EXEC DBLTEXT SYS$SYSDEVICE:[SYNERGY.DBL.DBLSTARLET]
```

DTK_BEEP – Set initial value of g_beep

The DTK_BEEP environment variable determines how U_START will initialize the **g_beep** field (a global value defined in **tkctl.def**), which turns the terminal bell (all platforms) and MessageBeep (Windows) on or off.

Value

One of the following:

0 or **false**	Initializes **g_beep** to false (0).
Any other value	Initializes **g_beep** to true (1). (default)

Discussion

If DTK_BEEP is set to a value of **0** or **false** (case-insensitive), U_START initializes **g_beep** to false. If **g_beep** is set to false, calls to U_BEEP have no effect, and runtime routines that are private to UI Toolkit suppress the terminal bell and MessageBeep. For more information, see "Customizing Configuration Fields" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual*.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** or **synuser.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit

Examples

```
set DTK_BEEP=false
```

DTK_BOUNDS – Enable UI Toolkit bounds checking

The DTK_BOUNDS environment variable enables bounds checking in UI Toolkit.

Value

One of the following values:

0	Turn off bounds checking.
1	Set bounds checking to only report cases where a buffer overrun occurs.
2	Set bounds checking to enforce exact argument size checks in all cases.

Discussion

Setting DTK_BOUNDS catches catastrophic failures whereby a reference exceeds the calling routine's data space, including such failures as data being overwritten and segmentation violations.

DTK_BOUNDS can be set at any point before calling U_START to turn on bounds checking in UI Toolkit. To control the bounds-checking behavior after U_START is called, modify the value of **g_dtkbounds** in **WND:tkctl.def**. See "tkctl.def" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual*.

If UI Toolkit bounds checking is enabled (that is, DTK_BOUNDS is set to 1 or 2), the fatal Toolkit error "Insufficient data spaced passed to input routine for storage of field" occurs if the record passed to I_DISPLAY, I_GETFLD, I_PUTFLD, I_INPUT, I_INIT, or L_INPUT is not large enough to contain the referenced data. If DTK_BOUNDS is set to 2, this error will also occur if a field specified in a call to I_DSPFLD, I_GETFLD, I_INPFLD, or I_PUTFLD is not the exact size as defined for the field. For I_DSPFLD, I_INPFLD, and L_INPFLD, this error will occur as the user arrives on the field if the data area is not the correct size when the field is processed.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- DTKDBG on page 1-74 to enable the UI Toolkit debugger.
- WINDBG in the "Debugging Your Synergy Programs" chapter of *Synergy Language Tools* to invoke the UI Toolkit debugger from the Synergy debugger prompt.
- U_DEBUG in the "Utility Routines" chapter of the *UI Toolkit Reference Manual* for instructions on invoking the Toolkit debugger programmatically.

Examples

On Windows,

```
set DTK_BOUNDS=1
```

DTK_MENU_UP – Disable automatic pull-down menus

WIN

The DTK_MENU_UP environment variable disables the automatic pull-down of menus.

Value

Contains any value if the menu should not be pulled down automatically when UI Toolkit's M_PROCESS subroutine is called.

Discussion

Setting DTK_MENU_UP to any value disables automatic pull-down of menus. The default is *not* set, which means automatic pull-downs are enabled. Calling the UI Toolkit menu subroutine M_DEFCOL(0) has the same effect as setting DTK_MENU_UP.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

M_DEFCOL in the "Menu Routines" chapter of the *UI Toolkit Reference Manual* for instructions on setting a default pull-down menu column or entry.

Examples

If set at the command prompt,

```
set DTK_MENU_UP=1
```

DTK_THROW_ABORT – Set initial value of g_throwabort

The DTK_THROW_ABORT environment variable is used by the UI Toolkit U_START routine to determine the initial value of **g_throwabort** (a global defined in **tkctl.def**).

Value

One of the following:

1	**g_throwabort** is set to true (1).
Any other value	**g_throwabort** is set to false (0).

Discussion

DTK_THROW_ABORT sets the initial value of **g_throwabort**, which determines the behavior of the U_ABORT subroutine. If **g_throwabort** is set to 0 (the default), U_ABORT performs a STOP and displays a fatal error message when invoked. The STOP has an exit status of D_EXIT_FAILURE, and U_ABORT does not return to the calling routine. If **g_throwabort** is set to a nonzero value, U_ABORT calls U_FINISH to shut down the Toolkit environment and then calls EXITE to throw a trappable error. For more information, see U_ABORT in the “Utility Routines” chapter of the *UI Toolkit Reference Manual*.

If DTK_THROW_ABORT is not defined, **g_throwabort** is set to false (0).

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit

Examples

```
set DTK_THROW_ABORT=1
```

DTKDBG – Enable the UI Toolkit debugger

The DTKDBG environment variable enables UI Toolkit debugging.

Value

Any value.

Discussion

If DTKDBG is set and UI Toolkit generates a fatal error, a window is displayed asking if you want to examine the window system. If you answer **y**, the UI Toolkit debugger is started. In addition, while a Toolkit program is running, you can explicitly enter debug mode by pressing CTRL+R while performing any Toolkit input.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- WINDBG in the "Debugging Your Synergy Programs" chapter of *Synergy Language Tools* for information on invoking the UI Toolkit debugger from the Synergy debugger prompt.
- U_DEBUG in the "Utility Routines" chapter of the *UI Toolkit Reference Manual* for instructions on invoking the Toolkit debugger programmatically.
- DTK_BOUNDS on page 1-70 to enable bounds checking in UI Toolkit.

Examples

On Windows,

```
set DTKDBG=1
```

DTKFSWINSIZ – Size of file-stack memory cache

The environment variable DTKFSWINSIZ defines the size of the memory cache used by the UI Toolkit file-stack routines.

Value

Any multiple of 16,384 bytes, up to a maximum of 122,880.

Discussion

The default size is 16,384.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

On OpenVMS,

```
$ DEFINE DTKFSWINSIZ 32768
```

DTKKEYCTLFIL – Key mapping script file for UI Toolkit

The DTKKEYCTLFIL environment variable defines the full path and filename of the UI Toolkit key mapping script file.

Value

The full path and filename, including extension, of the UI Toolkit key mapping script file. This must be a valid path and filename for the OPEN command. (See OPEN in the “Synergy Language Statements” chapter of the *Synergy Language Reference Manual*.)

Discussion

The DTKKEYCTLFIL environment variable enables you to name the key mapping script file and put it in a directory of your choosing. If this environment variable is not set, UI Toolkit looks for a key mapping script file named **keymap.ctl** in the working directory. If Toolkit doesn’t find **keymap.ctl** there, Toolkit looks for this file in the directory specified by WND. If Toolkit can’t find a key mapping script file, the only shortcuts supported by scripts are F1 through F63.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** or **synusr.ini** (where *myprog* is the **.dbr** file used to compile the script).

Used by

UI Toolkit.

See also

“Files and environment variables used for key mapping” in the “Customizing UI Toolkit” chapter of the *UI Toolkit Reference Manual* for information on the key mapping script file.

Examples

On UNIX,

```
DTKKEYCTLFIL=/usr/custom/kmsf1.ctl     ;export DTKKEYCTLFIL
```

DTKMAP – Directory for UI Toolkit key mapping file

The DTKMAP environment variable defines the directory path containing the UI Toolkit key mapping file, **dtkmap.ism**.

Value

The path, including the device, for the directory that contains the UI Toolkit key mapping file.

Discussion

If the environment variable DTKMAPFIL is not set, UI Toolkit looks for the **dtkmap.ism** file in the path specified by DTKMAP.

If DTKMAP is not set, UI Toolkit looks for the **dtkmap.ism** file in the path specified by the WND environment variable.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

On UNIX,

```
DTKMAP=/usr/custom    ;export DTKMAP
```

DTKMAPFIL – UI Toolkit key mapping file

The DTKMAPFIL environment variable defines the full path and filename of the UI Toolkit key mapping file.

Value

The full path and filename of the UI Toolkit key mapping file.

Discussion

If the environment variable DTKMAPFIL is not set, UI Toolkit looks for the key mapping file, **dtkmap.ism**, in the path specified by DTKMAP.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

On Windows, if set in the **synergy.ini** file,

```
[synergy]
DTKMAPFIL=c:\home\mykeymap.ism
```

DTKRND – Directory for UI Toolkit rendition file

The DTKRND environment variable defines the directory path containing the UI Toolkit rendition file, **dtkrnd.ism**.

Value

The path, including the device, for the directory that contains the UI Toolkit rendition file.

Discussion

If the environment variable DTKRNDFIL is not set, UI Toolkit looks for the **dtkrnd.ism** file in the path specified by DTKRND.

If DTKRND is not set, UI Toolkit looks for the **dtkrnd.ism** file in the current directory.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

On OpenVMS,

```
$ DEFINE DTKRND=DKA0:[TOOLKIT]
```

DTKRNDFIL – UI Toolkit rendition file

The DTKRNDFIL environment variable defines the full path and filename of the UI Toolkit rendition file.

Value

The full path and filename of the UI Toolkit rendition file.

Discussion

If the environment variable DTKRNDFIL is not set, UI Toolkit looks for the **dtkrnd.ism** file in the path specified by DTKRND.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

On UNIX,

```
DTKRNDFIL=/usr/custom/rndfil.ism    ;export DTKRNDFIL
```

DTKTERM – Key map for UI Toolkit

The DTKTERM environment variable defines the key map's terminal type and thus identifies the key map to use.

Value

The type of terminal you are using. This can be a user-defined name for a key map that you've created or one of the following distributed key maps:

ANSI	UNIX console
GENERIC	Unknown terminal type
HFT	IBM high-function terminal
IBMPC	PC
MSWINDOWS	Windows
VT100	VT*xxx* terminal
WY60	Wyse-60 and compatibles
XTERM	X-Windows terminals

Discussion

A key map file contains key maps for one or more terminal types. When you install Professional Series, you assign your default terminal type to the DTKTERM environment variable. Then, when the U_START subroutine initializes the window system, it opens the key map file and loads the key map for the terminal type defined in DTKTERM.

If DTKTERM is not set, the value of TERM is used instead.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit, Repository, ReportWriter.

See also

"Customizing Key Mapping for Menu Shortcuts" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for more information about terminal types.

Examples

On UNIX,

```
DTKTERM=ansi         ;export DTKTERM
```

DTKTMP – Directory for UI Toolkit temporary files

The DTKTMP environment variable defines the location where UI Toolkit will create temporary files.

Value

The path, including the device, for the directory that will contain UI Toolkit temporary files.

Discussion

If DTKTMP is not set, the default directory for temporary files is the current directory on UNIX and OpenVMS and TEMP (if it is set) on Windows. If TEMP is not set, UI Toolkit creates temporary files in the current directory.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

On Windows, if set at the command prompt,

```
set DTKTMP=d:\temp
```

EDIT_SYSMENU – Include “Edit” entry on system menu

WIN

The EDIT_SYSMENU environment variable determines whether the “Edit” menu entry is placed on the application window’s system menu.

Value

One of the following flags:

0	Do not include the “Edit” menu entry.
1	Include the “Edit” menu entry. (default)

Discussion

The “Edit” menu entry is actually a submenu containing the entries “Mark,” “Copy,” and “Paste.” As an alternative to using the “Edit” system menu entry, the “Mark,” “Cut,” “Copy,” “Paste,” and “Clear” edit menu entries are available in UI Toolkit to be placed on a designated menu column. See Appendix B in the *UI Toolkit Reference Manual* for a list of the reserved edit menu entries and their functions.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

If set in the **synergy.ini** file,

```
[synergy]
EDIT_SYSMENU=0
```

ENDPORT – Ending of a range of ports to check for *xf*Server

With BEGPORT, the ENDPORT environment variable sets the range of ports that **synxfpng** will check for a running *xf*Server.

Value

The number of the last port in the range to check. The default value is 2339.

Setting location

The environment.

Used by

synxfpng.

See also

- BEGPORT on page 1-33.
- "The synxfpng Utility" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Examples

Using the examples below, **synxfpng** checks ports 2360 – 2370 for a running *xf*Server.

```
BEGPORT=2360
ENDPORT=2370
```

FONT – Palette entry definition

WIN

One or more *FONT* initialization settings define an application's initial font palette. Each *FONT* setting defines a font palette entry.

Value

FONT is specified as follows:

palette_name=face_name;point_size[;sizing_char]

where the arguments are defined as follows:

palette_name

The name of the font palette entry. (**a**)

face_name

The name of the Windows typeface. (**a**)

point_size

The size of the Windows font. (**n**)

sizing_char

(optional) The sizing character. (**a**)

Discussion

Font palette entry names may be referenced within Repository, UI Toolkit window scripts, Composer, and at runtime.

Palette_name is a name of up to 60 characters (a-z, 0-9, _, and $) and is case insensitive. However, *palette_name* must begin with a letter. *Palette_name* should be unique within the [fonts] section of **synergy.ini**. If there is a nonunique *palette_name*, the last *palette_name* defined takes precedence.

Face_name indicates the font's typeface and is case sensitive. For example, Arial is the name of a typeface.

Point_size is the numeric point size of the font. Strictly defined, a point is 1/72 of an inch in terms of height.

Sizing_char is a single character used to determine the cell size within an object for the purposes of positioning. For example, specifying a sizing character of "W" causes an object to be significantly wider than with a sizing character of "i". *Sizing_char* is case sensitive. The default sizing character is a capital "A".

Four palette entry names are reserved, DEFAULT, ALTERNATE, DEBUGGER, and STATUS. Use the first three to specify the global, alternate, and debugger font characteristics, respectively. These are used if FONT_GLOBAL, FONT_ALTERNATE, or FONT_DEBUG is undefined. (DEFAULT is used to define the debugger font if the debugger font is not otherwise specified.) Use STATUS to specify the font for the application window's header, footer, and information line. It is used if FONT_HEADER, FONT_FOOTER, or FONT_INFO is undefined.

The *FONT* initialization settings enable you to define multiple fonts. However, specifying a large number of different fonts uses additional memory, which may increase the program's start-up time.

Setting location

The [fonts] section of **synergy.ini**.

Used by

Runtime.

See also

"Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for general font information and details on the hierarchy used to determine the global, alternate, and debugger fonts.

Examples

In the **synergy.ini** file,

```
[fonts]
Default=Fixedsys;9;A
Alternate=MS Sans Serif;10;W
my_input_font=Times New Roman;12
```

FONT_ALPHAFLD – Alphanumeric field font

WIN

The FONT_ALPHAFLD environment variable sets the default font for all alphanumeric input fields that do not have their own font specification.

Value

The name of a font palette entry. (**a**)

Discussion

FONT_ALPHAFLD can be used to more easily migrate your application to proportional fonts.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for information on the font defaulting hierarchy.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_ALPHAFLD=my_alpha_font
```

FONT_ALTERNATE – Alternate global font

WIN

The FONT_ALTERNATE environment variable sets the alternate global font.

Value

The name of a font palette entry. (**a**)

Discussion

Synergy/DE on Windows uses the alternate global font for application windows greater than or equal to 132 columns and for any object that does not have its own specification, but only when automatic font switching is enabled.

We recommend that you use a fixed font as the alternate global font when doing non-Toolkit processing.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the alternate font.
- %U_WNDFONT in the "Utility Routines" chapter of the *UI Toolkit Reference Manual* for information on automatic font switching.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_ALTERNATE=my_alt_font
```

FONT_DEBUG – Debugger font

WIN

The FONT_DEBUG environment variable sets the font for the Synergy debugger window.

Value

The name of a font palette entry. (**a**)

Discussion

For readability, we recommend that the debugger font be a fixed font, not a proportional font.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the debugger font.

Examples

In the **synergy.ini** file,

```
[synergy]
FONT_DEBUG=my_fixed_font
```

FONT_FOOTER – Initial footer section font

WIN

The FONT_FOOTER environment variable sets the default font for the application window's footer section.

Value

The name of a font palette entry. (**a**)

Discussion

The sizing character specified in the definition of the font palette entry has no effect on the size of the footer section.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for information on the font defaulting hierarchy.
- %E_FONT in the "Environment Routines" chapter of the *UI Toolkit Reference Manual* for setting the footer font at runtime.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_FOOTER=footer_font

[Fonts]
footer_font=MS Sans Serif;10
```

FONT_GLOBAL – Global font

WIN

The FONT_GLOBAL environment variable sets the global font.

Value

The name of a font palette entry. (**a**)

Discussion

Synergy/DE on Windows uses the global font for the following: application windows less than 132 columns, application windows greater than or equal to 132 columns (if automatic font switching is disabled), and any object that does not have its own font specification.

We recommend that you use a fixed font as the global font when doing non-Toolkit processing.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the global font.
- %U_WNDFONT in the "Utility Routines" chapter of the *UI Toolkit Reference Manual* for information on automatic font switching.

Examples

In the **synergy.ini** file,

```
[rps]
FONT_GLOBAL=my_app_font
```

FONT_HEADER – Initial header section font

WIN

The FONT_HEADER environment variable sets the default font for the application window's header section.

Value

The name of a font palette entry. (**a**)

Discussion

The sizing character specified in the definition of the font palette entry has no effect on the size of the header section.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for information on the font defaulting hierarchy.
- %E_FONT in the "Environment Routines" chapter of the *UI Toolkit Reference Manual* for setting the header font at runtime.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_HEADER=header_font

[Fonts]
header_font=Times New Roman;12
```

FONT_HEIGHT – Font height

WIN

The FONT_HEIGHT environment variable sets the height of the global font (in conjunction with TYPE_FACE and FONT_WIDTH) if a global font is not otherwise specified.

Value

The height, in logical units, of the font. (**n**)

Discussion

Synergy/DE on Windows uses the specified font height for application windows less than 132 columns. It is also used by Synergy Language and the debugger.

The font height can be specified in one of three ways:

- If the height is greater than 0, it is transformed into device units and matched against the cell height of the available fonts.
- If it is 0, a reasonable default size is used.
- If it is less than 0, it is transformed into device units, and the absolute value is matched against the character height of the available fonts.

If the current font is not available in the requested size, Synergy substitutes the font that most closely resembles the font that was specified.

We recommend that you use the FONT_GLOBAL environment variable rather than FONT_HEIGHT.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the global font.
- FONT_GLOBAL on page 1-91.

Examples

In the **synergy.ini** file,

```
[synergy]
FONT_HEIGHT=-17
```

FONT_INFO – Initial information line font

WIN

The FONT_INFO environment variable sets the default font for the application window's information line.

Value

The name of a font palette entry. (**a**)

Discussion

The sizing character specified in the definition of the font palette entry has no effect on the size of the information line.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for information on the font defaulting hierarchy.
- %E_FONT in the "Environment Routines" chapter of the *UI Toolkit Reference Manual* for setting the information line font at runtime.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_INFO=info_font

[Fonts]
info_font=Courier;10
```

FONT_LIST – List font

WIN

The FONT_LIST environment variable sets the default font for any list that does not have its own font specification. (The font for a list is controlled by the font assigned to the input window associated with the list.)

Value

The name of a font palette entry. (**a**)

Discussion

FONT_LIST can be used to more easily migrate your application to proportional fonts.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for information on the font defaulting hierarchy.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_LIST=my_list_font
```

FONT_NUMFLD – Numeric field font

WIN

The FONT_NUMFLD environment variable sets the default font for all numeric input fields that do not have their own font specification.

Value

The name of a font palette entry. (**a**)

Discussion

FONT_NUMFLD can be used to more easily migrate your application to proportional fonts.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for information on the font defaulting hierarchy.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_NUMFLD=my_numeric_font
```

FONT_PROMPT – Prompt font

WIN

The FONT_PROMPT environment variable sets the default font for all input field prompts that do not have their own font specification.

Value

The name of a font palette entry. (**a**)

Discussion

FONT_PROMPT can be used to more easily migrate your application to proportional fonts.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for information on the font defaulting hierarchy.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_PROMPT=my_prompt_font
```

FONT_TEXT – Text font

WIN

The FONT_TEXT environment variable sets the default font for the text in any window that does not have its own font specification.

Value

The name of a font palette entry. (**a**)

Discussion

FONT_TEXT can be used to more easily migrate your application to proportional fonts.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- FONT on page 1-85 for defining font palette entries.
- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for information on the font defaulting hierarchy.

Examples

In the **synergy.ini** file,

```
[myapp]
FONT_TEXT=my_text_font
```

FONT_WIDTH – Font width

WIN

The FONT_WIDTH environment variable sets the width of the global font (in conjunction with TYPE_FACE and FONT_HEIGHT) if a global font is not otherwise specified.

Value

The average width, in logical units, of characters in the font.

Discussion

Synergy/DE on Windows uses the specified font width for application windows less than 132 columns. It is also used by Synergy Language and the debugger. We recommend that you use the FONT_GLOBAL environment variable rather than FONT_WIDTH.

If the width is 0, the aspect ratio of the device is matched against the digitization aspect ratio of the available fonts to find the closest match, determined by the absolute value of the difference.

If the current font is not available in the requested size, Synergy/DE on Windows substitutes the font that most closely resembles the font that was specified.

When you use any of the FONT_ initialization settings and use the I_FLDMOD subroutine to change the type of a field (for example, from alpha to numeric), be aware that the corresponding font isn't automatically applied. To update the font, use the D_FLD_FONT option for I_FLDMOD.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the global font.
- FONT_GLOBAL on page 1-91.

Examples

In the **synergy.ini** file,

```
[synergy]
FONT_WIDTH=0
```

GENESIS_HOME – Connect file

Value

The directory that contains the connect file(s) specifying the location of the system catalog.

Discussion

The GENESIS_HOME environment variable is required and is automatically set when you install *xf*ODBC. You can change this setting if necessary.

GENESIS_HOME is used by the *xf*ODBC driver, DBA, and **dbcreate**. It is used when you generate system catalogs, when you modify system catalogs, and when you connect to the database.

Setting location

- For stand-alone configurations on Windows and UNIX, GENESIS_HOME must be set in the environment.
- On UNIX, you can change this setting either manually or by running the **setsde** script file (located in the synergyde directory).
- For stand-alone configurations on OpenVMS, GENESIS_HOME must be set in **CONNECT_STARTUP.COM**.
- In client/server configurations, GENESIS_HOME must be set in the environment on the server, and it must be set before starting the OpenNet server. (Note that the SQL OpenNet server doesn't use settings in **synergy.ini**, and it uses only settings made before it's started.) GENESIS_HOME must also be set in the environment on the client unless you set GENESIS_MSG_FILE.
- GENESIS_HOME cannot be set in **synergy.ini**.

Used by

*xf*ODBC.

See also

"Setting Options and File Locations" in the "Preliminary Steps" chapter of the *xfODBC User's Guide*.

Examples

On Windows,

```
set GENESIS_HOME=c:\synergyde\connect\synodbc
```

GENESIS_INITSQL – SQL options file

Value

The path and filename of your SQL options file.

Discussion

The GENESIS_INITSQL environment variable enables you to specify a file that contains predefined SQL statements. This includes all options that can be set with the SET OPTION command. The SQL statements in this file are executed each time a connection is made to the driver.

Setting location

The environment. For client/server configurations, GENESIS_INITSQL must be set on the server. On Windows, GENESIS_INITSQL must be set in **opennet.srv**.

Used by

*xf*ODBC.

See also

- "Creating a file for query processing options" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*.
- Appendix B of the *xfODBC User's Guide* for information on SET OPTION.

Examples

On Windows,

```
set GENESIS_INITSQL=c:\synergyde\connect\synodbc\options.sql
```

GENESIS_MSG_FILE – Error message file

Value

The path and name of the *xf*ODBC error message file. The default error message file is **sql.msg**.

Discussion

The GENESIS_MSG_FILE environment variable is automatically set when you install *xf*ODBC.

To generate a system catalog, the **dbcreate** utility must be able to locate the error message file. If GENESIS_MSG_FILE is set, **dbcreate** uses this setting to locate the file. If this variable is *not* set, **dbcreate** attempts to locate the file **sql.msg** in the GENESIS_HOME\lib directory.

Setting location

The environment. For client/server configurations, GENESIS_MSG_FILE must be set on the client and the server. For services such as web servers that use the *xf*ODBC driver, you can use the Env. variables field in the xfODBC Setup window to set this environment variable on the client.

Used by

*xf*ODBC.

See also

- GENESIS_HOME on page 1-101.
- "Specifying the name and location of the error message file" in the "Preliminary Steps" chapter of the *xfODBC User's Guide*.

Examples

On Windows,

```
set GENESIS_MSG_FILE=c:\synergyde\connect\synodbc\lib\sql.msg
```

HTTP_NOESCAPE – Disable HTTP escaping and unescaping of URIs

Value

Any value.

Discussion

Setting HTTP_NOESCAPE to any value turns off automatic escaping and unescaping of URIs by the Synergy HTTP document transport API.

In version 8.3, we added automatic escaping of certain unescaped characters in a URI by %HTTP_CLIENT_GET and %HTTP_CLIENT_POST, and automatic unescaping of any escape codes in a URI by %HTTP_SERVER_RECEIVE. If you had already implemented your own escape mechanisms prior to this feature being added, you may want to set HTTP_NOESCAPE to avoid breaking your existing code.

If you set HTTP_NOESCAPE, you will need to translate characters such as spaces manually. For instance, in order to pass a URI that contains spaces, you must first convert each space to the characters "%20" in the URI. To get **my big file.html** from **myserver**, for example, you would pass the following URI to %HTTP_CLIENT_GET:

```
"http://myserver/my%20big%20file.html"
```

Setting location

The environment.

Used by

Synergy HTTP document transport API.

See also

- %SYN_ESCAPE_HANDLE and %SYN_UNESCAPE_HANDLE in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual*.
- %HTTP_CLIENT_GET, %HTTP_CLIENT_POST, and %HTTP_SERVER_RECEIVE in the "Synergy HTTP Document Transport API" chapter of the *Synergy Language Reference Manual*.

Examples

On Windows,

```
set HTTP_NOESCAPE=1
```

HTTP_RAND – File containing random data for HTTPS support

Value

The path and name of a text file or an entropy-gathering device (if available on your system).

Discussion

When the Synergy HTTP document transport API initializes HTTPS support, it uses random data to ensure that the data is secure, because random data helps prevent hackers from guessing patterns. For most systems, this random data can be gathered from recognized system entropy devices or from the screen itself, or from a temporary file filled with random logic. However, on some systems, these methods are not enough, and a "Cannot load random state" error is generated. To eliminate this error, you can define the HTTP_RAND environment variable to point at a file that will be used (as a last resort) to gather random data when the HTTPS system is initialized.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Synergy HTTP document transport API.

See also

"Synergy HTTP Document Transport API" in the *Synergy Language Reference Manual*.

Examples

On Windows,

```
set HTTP_RAND=c:\windows\random.txt
```

On UNIX,

```
HTTP_RAND=/etc/entropy   ;export HTTP_RAND
```

HTTPSLIB – HTTPS runtime support file

VMS

The HTTPSLIB logical specifies the location of the HTTPS runtime support shared image file (**HTTPSLIB.EXE**).

Value

The full path and filename of the **HTTPSLIB.EXE** shared image.

Discussion

HTTPSLIB is used for HTTPS encryption.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

Synergy HTTP document transport API.

Examples

```
$ define/sys/exec HTTPSLIB SYNERGYDE$ROOT:[DBL.BIN]HTTPSLIB.EXE
```

IDATASIZ – Irecovr buffer size

WIN, UNIX

The IDATASIZ environment variable sets the size of the input buffer for the **irecovr** utility.

Value

The size of the input buffer in bytes.

Discussion

The default buffer size is 8192 bytes. The maximum size that you can set is limited to 32,768 bytes. Larger values of *size* can make **irecovr** run faster.

Setting location

The environment.

Used by

Irecovr utility.

Examples

On UNIX,

```
IDATASIZ=4096    ;export IDATASIZ
```

IGNIS2 – Ignore .is2 file

WIN, UNIX

Setting the IGNIS2 environment variable allows a file to be opened when there is a matching **.is2** file.

Value

Any value.

Discussion

When IGNIS2 is not set, an OPEN of an ISAM file fails if a file with the same name and the extension **.is2** exists.

⚠ The existence of an **.is2** file may mean that the previous use of **irecovr** failed and that the contents of the file are invalid.

Setting location

The environment.

Used by

Runtime, compiler, **fcompare**, **fconvert**, **isutl**.

Examples

On UNIX,

```
IGNIS2=1     ;export IGNIS2
```

INTRAFILELOCKS – Control intraprocess file locking behavior

UNIX

The INTRAFILELOCKS environment variable determines whether intraprocess file locks are enforced.

Value

0	Do not enforce intraprocess file locks.
1	Enforce intraprocess file locks. (default)

Discussion

As of version 8.3, file locking is enforced among channels within the same process, and “File in use by another user” errors ($ERR_FINUSE) are generated as appropriate. This is the standard behavior on Windows and OpenVMS and avoids potential file problems that could result in loss of data on UNIX. Because UNIX locks are interprocess (unlike Windows and OpenVMS file locks, which are both inter- and intraprocess) and don’t block multiple uses on different channels from within the same process, we’ve provided this conventional approach to file locking. File-locking rules apply to all statements and routines that involve implicit or explicit file sharing (OPEN, COPY, SORT, DELET, RENAM, ISCLR, and ISAMC).

To override the default behavior (and continue executing version 8.1.7 and earlier applications that violate the above sharing rules), you can set INTRAFILELOCKS to 0.

We do not recommend setting INTRAFILELOCKS to 0. It is only supported for backwards compatibility.

On Sun Solaris, which as of 8.1 supports inter- and intraprocess file sharing, INTRAFILELOCKS has no effect.

Setting location

The environment.

Used by

Runtime, *xf*Server.

Examples

```
INTRAFILELOCKS=0    ;export INTRAFILELOCKS
```

ISAMC_REV – Create files compatible with other Synergy versions

WIN, UNIX

The ISAMC_REV environment variable enables you to create ISAM files compatible with previous versions of Synergy using the ISAMC subroutine, or to convert ISAM files either up to a higher revision level or down to a lower revision level using the **irecovr** or **fconvert** utility.

Value

One of the following revision numbers:

2	Create files compatible with 4.5 and 5.*x* that don't allow index caching.
3	Create Synergy Language 6 ISAM files that do allow index caching.
4	Create Synergy Language 7/8 ISAM files. (default)
5	Revision used when Revision 2 or 3 file is patched for use with **isutl**.

Discussion

By default, all ISAM files created with version 7.1 or higher are Revision 4 and are not recognizable by previous Synergy versions (although your program can still access files from an earlier version). To create files that are recognized by earlier versions of Synergy, or if you need the current Synergy runtime or Synergy/DE *xf*Server to coexist with an earlier runtime, set ISAMC_REV to 3 or 2.

Revision 2 and 3 files cannot be used with the current version of **isutl**. In order to run **isutl** with these older files, you must first convert them to Revision 4 or higher, either by setting ISAMC_REV=4 and running **irecovr** or **fconvert**, or by patching the files to Revision 5 using **isutl -p** (in which case you do not need to set ISAMC_REV). The **isutl -p** option can take a fraction of the time that **irecovr** or **fconvert** require to do the same thing; however, a Revision 5 file is more limited in use than Revision 4. Revision 5 files are limited to 8 keys and are not accessible by pre–version 7.5 Synergy/DE. If you do want to raise the revision, we recommend that you use **isutl -p 5** rather than ISAMC_REV.

If you are using ISAMC_REV to maintain an ISAM file revision other than the default, it must be set by every Synergy user who may create a file.

Setting this variable for the runtime does not affect how *xf*Server handles the file I/O. For example, if you set ISAMC_REV in the **synergy.ini** file or in your application with SETLOG, and you're using *xf*Server for your file I/O, ISAMC_REV will not be used by *xf*Server. You must also set ISAMC_REV on the *xf*Server machine for the *xf*Server process.

Setting location

The environment. ISAMC_REV can be reset by the SETLOG subroutine, and the runtime interprets the new setting.

Used by

Runtime, compiler, **fcompare**, **fconvert**, **isutl**, **irecovr**.

See also

"Patching an ISAM file to another version" in the "Synergy DBMS" chapter of *Synergy Language Tools*.

Examples

On UNIX,

```
ISAMC_REV=2    ;export ISAMC_REV
```

ISLOGMAX – Maximum size of isutl log file

WIN, UNIX

The ISLOGMAX environment variable sets the maximum size of the **isutl** log file.

Value

The maximum size of the **isutl** log file, in kilobytes, or one of the following values:

-1	Log file size is unlimited.
0	Disable logging.

Discussion

The default maximum size of the **isutl** log file is 1 megabyte (1,024 kilobytes). If the log file exceeds the maximum size, the oldest 25 percent of the file will be removed.

Setting location

The environment.

Used by

isutl utility.

See also

- isutl in the “Synergy DBMS” chapter of *Synergy Language Tools*.
- ISUTLLOG on page 1-113.

Examples

On UNIX,

```
ISLOGMAX=2048     ;export ISLOGMAX
```

ISUTLLOG – isutl log filename

WIN, UNIX

The ISUTLLOG environment variable specifies the name of the log file for the **isutl** utility.

Value

The path and name of the **isutl** log file.

Discussion

The default log filename is **DBLDIR:isutl.log** on Windows XP/2003 and UNIX and **TEMP:isutl.log** on Vista/2008 and higher.

Setting location

The environment.

Used by

isutl utility.

See also

- isutl in the "Synergy DBMS" chapter of *Synergy Language Tools*.
- ISLOGMAX on page 1-112.

Examples

On UNIX,

```
ISUTLLOG=DBLDIR:isam.log     ;export ISUTLLOG
```

JBWAIT – Set wait time for RUNJB

The JBWAIT environment variable was removed from Synergy Language in version 8.3.

KEEP_BORDER – Always keep window border

WIN

The KEEP_BORDER environment variable causes the WPO_KEEPBRDR setting of the W_PROC WP_OPTION subfunction to be used.

Value

1.

Discussion

By default, when a window is one less in height or width than the size of the application window, the window border is turned off. WPO_KEEPBRDR prevents window borders from being turned off due to the size of the window. The WPO_HIDEBRDR option restores the default behavior.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Facilities

Runtime.

Example

```
KEEP_BORDER=1
```

LIBBSIZ – Librarian buffer size

WIN, UNIX

The LIBBSIZ environment variable sets the size of output cache buffers for the Synergy librarian. Using a larger buffer size may improve the performance on large libraries.

Value

The size of a cache buffer in bytes.

Discussion

The default buffer size is 16384 bytes. The maximum size you can set is limited by the memory available on your system.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dblibr] section of **synergy.ini**.

Used by

Librarian.

Examples

On Windows, if set in the **synergy.ini** file,

```
[synergy]
LIBBSIZ=16384
```

LIBNBUF – Number of librarian cache buffers

WIN, UNIX

The LIBNBUF environment variable sets the number of cache buffers for the Synergy librarian.

Value

The number of buffers.

Discussion

The default is eight buffers. Higher *buffer_counts* may improve the performance on large libraries. The maximum number of buffers you can set is limited only by the memory available on your system.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dblibr] section of **synergy.ini**.

Used by

Librarian.

Examples

On UNIX,

```
LIBNBUF=16    ;export LIBNBUF
```

LNKBSIZ – Linker buffer size

WIN, UNIX

The LNKBSIZ environment variable sets the size of output cache buffers for the Synergy linker.

Value

The size of a cache buffer in bytes.

Discussion

The default buffer size is 16384 bytes. Using a larger buffer size may improve performance on large programs. The maximum size you can set is limited by the memory available on your system.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dblink] section of **synergy.ini**.

Used by

Linker.

Examples

On Windows, if set at the command prompt,

```
set LNKBSIZ=16384
```

LNKNBUF – Number of linker cache buffers

WIN, UNIX

The LNKNBUF environment variable sets the number of cache buffers for the Synergy linker.

Value

The number of buffers.

Discussion

The default is eight buffers. The maximum number of buffers you can set is limited by the memory available on your system.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dblink] section of **synergy.ini**.

Used by

Linker.

Examples

On UNIX,

```
LNKNBUF=16    ;export LNKNBUF
```

MAKEDBR_LEAN – No client/server support

UNIX

The MAKEDBR_LEAN environment variable enables the runtime to be built without client/server support.

Value

none.

Discussion

By default, the runtime is built with client/server support. If you set MAKEDBR_LEAN to “none,” **makedbr** builds a runtime without this support.

Use this environment variable on systems that don’t have the appropriate client/server support routines.

Setting location

The environment.

Used by

Runtime.

Examples

```
MAKEDBR_LEAN=none    ;export MAKEDBR_LEAN
```

MAKEDBR_NORPC – No client/server support

UNIX

The MAKEDBR_NORPC environment variable is functionally equivalent to MAKEDBR_LEAN. See page 1-120 for more information.

MAKEDBR_TLIB – Termcap vs. terminfo

UNIX

The MAKEDBR_TLIB environment variable specifies the library with which the runtime should be built.

Value

One of the following libraries:

termcap	Build the runtime with the **termcap** library.
terminfo	Build the runtime with the **terminfo** library. (default)

Discussion

By default, the runtime is built with the **terminfo** library. Setting MAKEDBR_TLIB to “termcap” builds the runtime with the **termcap** library instead of the **terminfo** library.

Setting location

The environment.

Used by

Runtime.

Examples

```
MAKEDBR_TLIB=termcap    ;export MAKEDBR_TLIB
```

MAXMEM – Maximum allocated memory

WIN, UNIX

The MAXMEM environment variable specifies the maximum amount of memory (in bytes) that the runtime allocates for user program code and data before removing inactive segments from memory.

Value

The maximum amount of memory (in bytes) that you want allocated before inactive memory segments are removed from memory.

Discussion

The default value for MAXMEM on UNIX is 1,310,720 bytes (1.25 MB). The default value on Windows is 8,388,608 bytes (8 MB). On Windows, a MAXMEM minimum of 4 MB is enforced, even if an explicit MAXMEM value has been specified. The maximum number of bytes that you can specify is limited by the memory available on your system.

Larger values of *memory* may dramatically improve performance. Smaller values cause the memory segments to be released more often, slowing down performance but conserving memory.

If memory is not large enough to run the program after releasing all available segments, the internal value for *memory* is increased to be large enough to contain the minimum required program segments for execution. It is never reduced. If it exceeds any value specified by MAXMEMMAX, an error occurs.

If more than 10 segment reclamations occur in a one-second interval, MAXMEM is automatically increased by 100,000. To see the current MAXMEM value for your program, use SHOW MEMORY in the debugger.

If MAXMEM is greater than 50,000,000, the runtime never reclaims segments. This is to assist with VAX DIBOL code migration, where records in routines are assumed to be static records.

To check whether memory reclamation is occurring with a specific value of MAXMEM, set the MEMDBG environment variable to 1. The system beeps on each reclamation.

WIN

For optimal performance, set *memory* per program in the [*myprog*] section of your **synergy.ini** file.

When UI Toolkit is installed, a MAXMEM setting of 16 MB (16 x 2^{20}) is added to the [COMPOSER] section of **synergy.ini**.

When Repository is installed, a MAXMEM setting of 3 MB (3 x 2^{20}) is added to the [RPS] section of **synergy.ini**.

When ReportWriter is installed, a MAXMEM setting of 3 MB (3 x 2^{20}) is added to the [RPT] section of **synergy.ini**.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

On Windows, if set in the **synergy.ini** file,

```
[synergy]
MAXMEM=2097152
```

MAXMEMMAX – Test maximum allocated memory

WIN, UNIX

MAXMEMMAX is a debugging aid to see how your application would work if its memory was limited to a certain value.

Value

The maximum amount of memory (in bytes).

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

On UNIX,

```
MAXMEMMAX=2000000    ;export MAXMEMMAX
```

MAXRECURSELEVEL – Maximum number of recursion levels

MAXRECURSELEVEL enables you to specify a limit that is greater than the default 1500 levels of method, function, or subroutine calls (or the maximum for the system stack, if it is less, as on Windows x64).

Value

The maximum number of recursion levels that can occur before an error is generated.

Discussion

By default, the runtime detects when approximately 1500 levels of method, subroutine, or function calls (or the maximum for the system stack) have been made and generates a fatal "Runtime stack overflow" error (STKOVR) when that level is exceeded. If the system stack allows, you can set MAXRECURSELEVEL to a value greater than 1500 to override the limit.

Based on the system stack, the specified value may or may not be possible. If it is not possible, your program could stop running without warning.

Setting location

The environment.

Used by

Runtime.

Examples

On UNIX,

```
MAXRECURSELEVEL=2500   ;export MAXRECURSELEVEL
```

MEMDBG – Monitor memory reclamation

WIN, UNIX

The MEMDBG environment variable enables you to monitor memory reclamation.

Value

Any value.

Discussion

When MEMDBG is set, the runtime rings the terminal bell every time memory reclamation is performed. This allows you to tune your MAXMEM setting.

Do not set MEMDBG in the **synergy.ini** file. Doing so may result in serious problems.

Setting location

The environment.

Used by

Runtime.

Examples

On Windows, if set at the command prompt,

```
set MEMDBG=1
```

MINIMIZE_LEADING – Reduce line spacing

WIN

The MINIMIZE_LEADING environment variable reduces the amount of space between lines of text in windows on the screen.

Value

One of the following values:

0	Normal leading (8 pixels) (default)
1	Less leading (6 pixels)
2	No leading (0 pixels)
3	Minimal leading (2 pixels)

Discussion

Setting MINIMIZE_LEADING to 1, 2, or 3 can enable you to fit more lines of text on your screen. It applies to all windows, including the application container window. (In contrast, the UI Toolkit's %U_WNDSTYLE function only applies to one window.)

By default, the leading between rows is large enough to allow space for an edit control frame and its 3-D effects and to avoid any clipping of text within that edit control. Edit controls are used to frame input fields.

Setting MINIMIZE_LEADING to 1 reduces the amount of leading to just the amount required to display the frame. Some clipping of text may occur, and 3-D effects of adjacent fields may overlap.

Setting MINIMIZE_LEADING to 2 eliminates all leading between lines. This means that no extra space is allowed for an edit control frame. If edit controls are displayed, they overlap one another noticeably.

Setting MINIMIZE_LEADING to 3 may be useful when using applications that are purely ACCEPT/DISPLAY/READS programs or that use only the Synergy windowing API and no UI Toolkit calls.

Calling %U_WNDSTYLE for a window and specifying D_NO_LEADING is equivalent to setting MINIMIZE_LEADING=2 for a single window. Specifying D_NO_LEADING overrides any setting of MINIMIZE_LEADING for that window. However, if you call %U_WNDSTYLE and specify D_LEADING, the MINIMIZE_LEADING setting applies to that window.

If you use MINIMIZE_LEADING=2 to eliminate all inter-row spacing and you are using a Synergy window as a “background” display window, you may want to use

```
xcall w_proc(WP_POSITION, wndid, 0, 0)
```

to position that window at the exact upper-left corner of the application window. Otherwise, on some platforms (especially XP) and with some font combinations, the window may be placed several pixels below the very top.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

```
set MINIMIZE_LEADING=1
```

MSGWAIT – Licensing and message controller

UNIX

The MSGWAIT environment variable specifies the amount of time to wait for queued messages.

Value

The total time in seconds to wait for queued messages.

Discussion

MSGWAIT is normally set to 5 seconds. In excessively slow environments, this number can be increased such that a message time-out does not occur.

Setting location

The environment.

Used by

Runtime.

Examples

```
MSGWAIT=10    ;export MSGWAIT
```

NUMBUFS – Data file cache buffers

WIN, UNIX

The NUMBUFS environment variable defines the number of fixed cache buffers used for each file opened with data cache.

Value

The number of buffers.

Discussion

NUMBUFS sets the number of cache buffers used for each file opened with data cache. See the **/cache** option in the OPTIONS qualifier for the OPEN statement in the “Synergy Language Statements” chapter of the *Synergy Language Reference Manual* or system option #3 on page 2-6 for more information on using ISAM data cache.

By default, files opened with data cache allocate buffers dynamically depending on the size of the file. The number of buffers is based on the best mix of cache buffers for a particular file with respect to usage, and that is the sum of all keys’ index depths plus two times the number of keys. The size of each cache buffer is about the size of the index PAGE size for a file. See “Page size” in the “Synergy DBMS” chapter of *Synergy Language Tools* for more information.

The **irecovr** and **fconvert** utilities set NUMBUFS to 64. See irecovr and fconvert in the “Synergy DBMS” chapter of *Synergy Language Tools* for more information.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbl], [dbr], [dblink], [dblibr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime, compiler, **fcompare**, **fconvert**, **isutl**, **irecovr**.

Examples

On Windows, if set in the **synergy.ini** file,

```
[synergy]
NUMBUFS=3
```

OPENNET_POLL_TIME – SQL OpenNet service poll interval

WIN

Value

The period of time in milliseconds that **sqld** waits before polling the Synergy/DE OpenNet Server service (**SynSQL**).

Discussion

The OPENNET_POLL_TIME environment variable enables you to specify how often the service program **sqld** will check to see if the daemon(s) (**vtxnetd/vtxnet2**) are running. By default, **sqld** polls at 600,000-millisecond (10-minute) intervals.

Setting location

OPENNET_POLL_TIME must be set in **opennet.srv**.

Used by

*xf*ODBC and SQL Connection.

See also

- "Understanding SQL OpenNet on Windows" in the "Configuring Connectivity Series" chapter of the *Installation Configuration Guide*.
- "The sqld program" in the "Configuring Connectivity Series" chapter of the *Installation Configuration Guide*.

Examples

The following example sets the poll interval to one minute (60,000 milliseconds):

```
OPENNET_POLL_TIME=60000
```

OPENSSLLIB – OpenSSL library

WIN

The OPENSSLLIB environment variable specifies the location of the OpenSSL libraries.

Value

The directory path to the OpenSSL libraries.

Discussion

OPENSSLLIB is used for network and data encryption. If OPENSSLLIB is not set, the location specified in PATH is used.

Setting location

The environment for the Runtime. To be used by *xf*Server and *xf*ServerPlus, OPENSSLLIB must be set in the Windows registry. (You can use the Synergy Configuration Program to set *xf*Server and *xf*ServerPlus environment variables on Windows.)

Used by

Runtime, *xf*Server, *xf*ServerPlus.

Examples

```
OPENSSLLIB=c:\externtools\OpenSSL\out32dll
```

OPTIMIZE_REDRAW – Disable redraw optimization

WIN

The OPTIMIZE_REDRAW environment variable disables the Synergy runtime's optimization of screen redraw and causes all repainting to occur immediately.

Value

0.

Discussion

The Synergy runtime on Windows attempts to minimize the repainting of windows by occasionally turning updates off. Updates are automatically resumed when a W_UPDT subroutine (or its implied equivalent) or any operation that causes physical window updates to occur (such as field input, menu processing, and so forth) is processed. However, some applications have experienced undesirable display effects while updates are off. Especially if updates remain off for an extended period of time, other windows on the desktop may "show through" the Synergy application.

To disable the optimization of redraw, set OPTIMIZE_REDRAW in the environment prior to starting **dbr.exe**, or put it in **synergy.ini**. Currently, we only look for a value of 0, and if that is found, we disable repainting optimization. All other values for this variable currently have the same behavior as not setting the variable and cause normal redraw optimization to occur. However, we recommend that you only use a value of 0 or leave the variable unset, so that we may reserve other values for other possible operational behaviors in the future.

Setting OPTIMIZE_REDRAW may cause excessive repainting to occur in the application, but it may also solve other display anomalies. We recommend that you do *not* use this setting unless you need it to cure known display problems.

If OPTIMIZE_REDRAW is set, XCALL U_UPDATE(FALSE) (to explicitly turn updates off) has no effect.

If the container is not moved, only the container and contained windows are updated. However, if the container is moved or resized, the whole desk top is repainted when updates are turned back on. This is a Microsoft Windows limitation.

Setting location

The environment or in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

```
set OPTIMIZE_REDRAW=0
```

PALETTE – Synergy color palette

WIN

The PALETTE setting defines entries in the color palette.

Value

PALETTE is specified as follows:

PALETTE=*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*,*f*/*b*

where each *f*/*b* pair is a color palette entry with a foreground color (*f*) and a background color (*b*). The first *f*/*b* pair is for color palette entry 1, the second is for color palette entry 2, and so forth.

Discussion

The color palette has 16 entries that are used to assign colors to renditions, which define the appearance of user interface elements in UI Toolkit programs and Synergy programs that use the windowing API. A default color palette is loaded into memory when the Synergy runtime starts, but you can use PALETTE to override the entire default color palette or just specific entries in the default color palette. Note that

- there are 256 user colors (color 0 through color 255) and 30 system colors (color 256 through color 285) that you can assign to palette entries. See "Colors and the color palette on Windows" in the "Synergy Windowing API" chapter of the *Synergy Language Reference Manual* for more information.
- a PALETTE setting does not need to include all 16 color palette entries. You can omit an entry, but you must include a comma placeholder for the omitted entry unless it is at the end of the string. (See Examples below.)
- PALETTE settings override the default memory-resident color palette (or specific entries in the default color palette).

Setting location

The [colors] section of **synergy.ini**.

Used by

Runtime and UI Toolkit.

See also

- COLORn on page 1-36.
- "Customizing the Look of Your Application" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual*.

Examples

The following example defines the entire color palette.

```
[colors]
PALETTE=264/271,0/15,264/271,270/269,0/7,0/7,0/7,0/7,0/7,0/7,0/7,0/7,0/7,
0/7,0/7,0/15
```

The next example defines the second and fourth color palette entries as **270/269**. All other color palette entries will have their default values (or values set in **synergy.ini** if this setting is in **synuser.ini**).

```
[colors]
PALETTE=,270/269,,270/269
```

PCMD – Print command

UNIX

The PCMD environment variable specifies a print command and causes all LPQUE options to be ignored.

Value

A command that LPQUE uses in place of the standard System V print command (for example, “lp”).

Discussion

If you don’t have a printer connected but you want to test your programs, you can set PCMD equal to “cat,” and LPQUE will send your output to your terminal. (We recommend, however, that you use system option #22 and **dblpq**.)

System option #22 overrides PCMD.

Setting location

The environment.

Used by

Runtime.

Examples

```
PCMD="lpr -q hp-laserjet"  ;export PCMD
```

Using this example, the statement

```
lpque(prog.dbl)
```

produces the following command:

```
lpr -q hp-laserjet prog.dbl
```

PRINT_METHOD – LPQUE print method

WIN

The PRINT_METHOD environment variable enables you to choose the print method when using the LPQUE statement.

Value

escape.

Discussion

When PRINT_METHOD=escape is set, data from files printed with the LPQUE statement goes directly to the printer, without Print Manager affecting the escape codes. The LPQUE statement uses printer driver default values, and any changes made to the driver are ignored.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file). To be used by *xf*Server, PRINT_METHOD must be set in the Windows registry. Refer to "Defining environment variables" in the Windows section of the "Configuring xfServer" chapter of the *Installation Configuration Guide* for information on setting environment variables in the Windows registry.

Used by

Runtime, *xf*Server.

Examples

If set in the **synergy.ini** file,

```
[synergy]
PRINT_METHOD=escape
```

PRINT_PREVIEW_BOTTOM – Bottom coordinate for print preview window

WIN

The PRINT_PREVIEW_BOTTOM environment variable specifies the lower coordinate for the print preview window in the Synergy Windows printing API.

Value

The bottom coordinate, in pixels.

Discussion

The print preview window enables users to preview any page of a document to see how it will look when it is printed. They can change the size or position of the print preview window. When you exit the print preview function, the bottom coordinate of the window is stored in PRINT_PREVIEW_BOTTOM by the Synergy Windows printing API. Do not edit this value manually.

Setting location

Written to the [synergy] section of the **synergy.ini** file by the Synergy Windows printing API.

Used by

Synergy Windows printing API.

PRINT_PREVIEW_LEFT – Left coordinate for print preview window

WIN

The PRINT_PREVIEW_LEFT environment variable specifies the left coordinate for the print preview window in the Synergy Windows printing API.

Value

The left coordinate, in pixels.

Discussion

The print preview window enables users to preview any page of a document to see how it will look when it is printed. They can change the size or position of the print preview window. When you exit the print preview function, the left coordinate of the window is stored in PRINT_PREVIEW_LEFT by the Synergy Windows printing API. Do not edit this value manually.

Setting location

Written to the [synergy] section of the **synergy.ini** file by the Synergy Windows printing API.

Used by

Synergy Windows printing API.

PRINT_PREVIEW_SCROLL – Scrolling percentage in print preview window

WIN

The PRINT_PREVIEW_SCROLL environment variable specifies what percentage of the display area to scroll (if it is scrollable) when an arrow key is pressed.

Value

A number between 1 and 100, inclusive, that specifies the percentage of the display area to scroll when an arrow key is pressed.

Discussion

The print previewer evaluates this environment variable each time the previewer is opened, so that setting the value programmatically can take effect on the next preview.

If PRINT_PREVIEW_SCROLL is not set, or if it is set to an invalid value, the value defaults to 5 (1/20th of the display area).

Setting location

The environment or the [synergy] section of **synergy.ini**.

Used by

Synergy Windows printing API.

Examples

The following example from the **synergy.ini** file scrolls 10 percent of the display area when an arrow key is pressed:

```
[synergy]
PRINT_PREVIEW_SCROLL=10
```

PRINT_PREVIEW_TOP – Top coordinate for print preview window

WIN

The PRINT_PREVIEW_TOP environment variable specifies the upper coordinate for the print preview window in the Synergy Windows printing API.

Value

The top coordinate, in pixels

Discussion

The print preview window enables users to preview any page of a document to see how it will look when it is printed. They can change the size or position of the print preview window. When you exit the print preview function, the top coordinate of the window is stored in PRINT_PREVIEW_TOP by the Synergy Windows printing API. Do not edit this value manually.

Setting location

Written to the [synergy] section of the **synergy.ini** file by the Synergy Windows printing API.

Used by

Synergy Windows printing API.

PRINT_PREVIEW_ZOOM – Zoom factor for print preview window

WIN

The PRINT_PREVIEW_ZOOM environment variable specifies the zoom factor for the print preview window in the Synergy Windows printing API.

Value

The zoom factor, which is a percentage.

Discussion

The print preview window enables users to preview any page of a document to see how it will look when it is printed. They can zoom in or out, changing the size of the page as displayed in the window. When you exit the print preview function, the current zoom factor is stored in PRINT_PREVIEW_ZOOM by the Synergy Windows printing API.

If you want to set a system-wide default print preview zoom factor, you can set PRINT_PREVIEW_ZOOM in **synergy.ini**.

Setting location

Written to [synergy] section of the **synergy.ini** file by the Synergy Windows printing API.

Used by

Synergy Windows printing API.

Examples

The following example from the **synergy.ini** file sets the zoom factor to 75 percent:

```
[synergy]
PRINT_PREVIEW_ZOOM=75
```

PROFILE_PROCESSOR_TIME – Profile using accumulated CPU time

WIN

The PROFILE_PROCESSOR_TIME environment variable tells the Synergy Language Profiler utility to calculate accumulated CPU time rather than elapsed CPU time.

Value

Any value.

Discussion

The Synergy Language Profiler calculates elapsed CPU time according to the high-granularity system clock. To calculate accumulated CPU time, which is only updated every 20 milliseconds, set the PROFILE_PROCESSOR_TIME environment variable. Note that on a very fast processor, accumulated CPU time results can be so imprecise as to be almost meaningless, but may be advantageous when profiling significant amounts of input.

Setting location

The environment or the [synergy] section of **synergy.ini**.

Used by

Synergy Language Profiler utility.

See also

"The Synergy Language Profiler" in the "General Utilities" chapter of *Synergy Language Tools*.

Examples

```
set PROFILE_PROCESSOR_TIME=1
```

PROXY_HOST – Proxy host for remote URI requests

The PROXY_HOST environment variable enables you to specify the proxy host name to use for remote URI requests through the Synergy HTTP document transport API.

Value

The name of the proxy host server.

Discussion

If PROXY_HOST is set, remote URI requests are sent to this proxy host. If it's not set, a direct connection is attempted to the host contained in the URI.

Setting location

The environment.

Used by

Synergy HTTP document transport API.

See also

"Synergy HTTP Document Transport API" chapter of the *Synergy Language Reference Manual*.

Examples

```
set PROXY_HOST="mymachine"
```

PROXY_LOCAL – Override handling of local URIs

The PROXY_LOCAL environment variable enables you to override the handling of local URIs by the Synergy HTTP document transport API.

Value

true.

Discussion

By default, the HTTP document transport API connects directly to the host of a local URI. If PROXY_LOCAL is set to **true**, the HTTP document transport API instead connects to the proxy server, identified by the PROXY_HOST environment variable, for local URI requests.

Setting location

The environment.

Used by

Synergy HTTP document transport API.

See also

"Synergy HTTP Document Transport API" chapter of the *Synergy Language Reference Manual*.

Examples

```
set PROXY_LOCAL="true"
```

PROXY_PORT – Proxy server port for remote URI requests

The PROXY_PORT environment variable enables you to specify the port on the proxy server to use for remote URI requests through the Synergy HTTP document transport API.

Value

The port number of the proxy host. The default is 80.

Discussion

If PROXY_PORT is set, the specified port is used when connecting to the proxy host. If it's not set, port 80 is used.

Setting location

The environment.

Used by

Synergy HTTP document transport API.

See also

"Synergy HTTP Document Transport API" chapter of the *Synergy Language Reference Manual*.

Examples

```
set PROXY_PORT="85"
```

PROXY_SUBNET – Subnet mask to identify local URIs

The PROXY_SUBNET environment variable specifies the subnet mask to use to identify local URIs within the Synergy HTTP document transport API.

Value

The local subnet mask used to identify whether a given URI is a local URI. The default is 255.255.0.0.

Discussion

This subnet mask is bitwise applied to the IP address of the host contained in the requested URI and compared to a known local IP address filtered by the same subnet mask to determine if the URI request is for a local server. Specify the subnet mask in dotted text form (for example, 255.255.255.0).

To help determine the correct value for PROXY_SUBNET, you can view your subnet mask by running **ipconfig**. The following is sample output:

```
Windows XP IP Configuration

Ethernet adapter El90x1:

     IP Address. . . . . . . . . : 10.1.3.76
     Subnet Mask . . . . . . . . : 255.255.0.0
     Default Gateway . . . . . . : 10.1.1.1
```

Setting location

The environment.

Used by

Synergy HTTP document transport API.

See also

“Synergy HTTP Document Transport API” chapter of the *Synergy Language Reference Manual*.

Examples

```
set PROXY_SUBNET="255.255.255.0"
```

RECVCTL – Alter irecovr behavior

WIN, UNIX

The RECVCTL environment variable alters the default behavior of the **irecovr** utility.

Value

NO_DSCAN.

Discussion

Setting RECVCTL to “NO_DSCAN” tells **irecovr** not to scan the data file before recovering. Normally, **irecovr** scans the data file before it starts recovery to determine if the data file is capable of being recovered.

Setting location

The environment.

Used by

Irecovr utility.

Examples

On UNIX,

```
RECVCTL=NO_DSCAN    ;export RECVCTL
```

RETAIN_CONTEXT_CHANGE_ON_SIGNAL – Set initial value of g_retaincontext

The RETAIN_CONTEXT_CHANGE_ON_SIGNAL environment variable is used by the UI Toolkit U_START routine to determine the initial value of **g_retaincontext** (a global defined in **tkctl.def**).

Value

One of the following:

1	**g_retaincontext** is set to true (1).
Any other value	**g_retaincontext** is set to false (0).

Discussion

RETAIN_CONTEXT_CHANGE_ON_SIGNAL sets the initial value of **g_retaincontext**, which determines how context is handled when a menu entry is signaled from a method during I_INPUT processing and a user action (such as a mouse click, ENTER, TAB, SHIFT+TAB, etc.) moves input context to a subsequent field or button. Prior to UI Toolkit 8.3, if a menu entry was signaled in this situation, Toolkit would incorrectly apply the menu entry processing to the subsequent field or button rather than to the field whose method called %M_SIGNAL. This was corrected in Toolkit 8.3. By default, context now remains on the field whose method signaled the menu entry, which matches Toolkit behavior on UNIX and OpenVMS. You can, however, restore the previous behavior. If **g_retaincontext** is set to true, Toolkit emulates Windows behavior for Toolkit versions prior to 8.3 (it moves input context to the subsequent field or button). If **g_retaincontext** is set to false (the default), context remains on the field whose method signaled the menu entry.

For more information, see "tkctl.def" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual*.

If RETAIN_CONTEXT_CHANGE_ON_SIGNAL is not defined, **g_retaincontext** is set to false (0).

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

```
set RETAIN_CONTEXT_CHANGE_ON_SIGNAL=1
```

RPS – Repository directory

The RPS environment variable specifies the directory that contains your Repository distribution. It is required for normal operation of the Repository maintenance program and utilities.

Value

The path, including the device, for the directory that contains the Repository distribution files.

Setting location

- On Windows, the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).
- On UNIX, the **setsde** file.
- On OpenVMS, the **SYNERGY_STARTUP.COM** file (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

Repository.

Examples

On UNIX,

```
RPS=/usr/synergy/rps     ;export RPS
```

RPSDAT – Repository data files directory

The RPSDAT environment variable defines the directory where Repository data files are located.

Value

The path, including the device, for the directory that contains the Repository data files.

Discussion

The default directory for Repository data files is the rpsdat subdirectory below the directory where Repository was installed. The RPSDAT directory contains the default repository files, **rpsmain.ism** and **rpstext.ism**, and the optional cross-reference file, **rpsxref.ism**.

Setting location

- On Windows, the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).
- On UNIX, the **setsde** file.
- On OpenVMS, the **SYNERGY_STARTUP.COM** file (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

Repository, ReportWriter, UI Toolkit, *xf*ODBC, **fcompare**.

Examples

On OpenVMS,

```
$ DEFINE/SYS/EXEC RPSDAT DKA0:[SYNERGY.RPS.RPSDAT]
```

RPSLIB – Repository subroutine library directory

The RPSLIB environment variable defines the directory that contains the Repository subroutine library.

Value

The path, including the device, for the directory that contains the Repository subroutine library.

Discussion

The default directory for Repository subroutine library is the lib subdirectory below the directory where Repository was installed. The RPSLIB directory contains the subroutine library **ddlib.elb** (**ddlib.olb** on OpenVMS), as well as the definition file **ddinfo.def**.

Setting location

- On Windows, the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).
- On UNIX, the **setsde** file.
- On OpenVMS, the **SYNERGY_STARTUP.COM** file (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

Repository.

Examples

On Windows,

```
[synergy]
RPSLIB=%SYNERGYDE%rps\lib
```

RPSMFIL – Repository main file

The RPSMFIL environment variable specifies the full path and filename of the Repository main file.

Value

The full path and filename of the Repository main file.

Discussion

If the environment variable RPSMFIL is not set, Repository looks for the **rpsmain.ism** file in the path specified by RPSDAT. If RPSDAT is not set, Repository looks for the **rpsmain.ism** file in the current directory.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Compiler, Repository, ReportWriter, UI Toolkit, *xf*ODBC, **fcompare**.

See also

DBLDICTIONARY on page 1-58.

Examples

On Windows, if set in the **synergy.ini** file,

```
[synergy]
RPSMFIL=c:\home\rpsmain.ism
```

RPSTFIL – Repository text file

The RPSTFIL environment variable specifies the full path and filename of the Repository text file.

Value

The full path and filename of the Repository text file.

Discussion

If the environment variable RPSTFIL is not set, Repository looks for the **rpstext.ism** file in the path specified by RPSDAT. If RPSDAT is not set, Repository looks for the **rpstext.ism** file in the current directory.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Compiler, Repository, ReportWriter, UI Toolkit, *xf*ODBC, **fcompare**.

See also

DBLDICTIONARY on page 1-58.

Examples

On UNIX,

```
RPSTFIL=/usr/rpstext.ism    ;export RPSTFIL
```

RPSTMP – Repository temporary files directory

The RPSTMP environment variable defines the location where Repository will create temporary files.

Value

The path, including the device, for the directory that will contain Repository temporary files.

Discussion

Repository puts any temporary files that it creates in RPSTMP if it is set. If RPSTMP is not set, Repository puts temporary files in TEMP if it is set. If TEMP is not set either, temporary files are placed in the current directory.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Repository.

Examples

On Windows, if set at the command prompt,

```
set RPSTMP=d:\temp
```

RPSXFIL – Repository cross-reference file

The RPSXFIL environment variable specifies the full path and filename of the Repository cross-reference file.

Value

The full path and filename of the Repository cross-reference file.

Discussion

If the environment variable RPSXFIL is not set, Repository looks for the **rpsxref.ism** file in the path specified by RPSDAT. If RPSDAT is not set, Repository looks for the **rpsxref.ism** file in the current directory.

See "Generating a Cross-Reference File" in the "Utility Functions" chaper of the *Repository User's Guide* for more information about Repository cross-reference files.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Repository, ReportWriter.

Examples

On OpenVMS,

```
$ DEFINE RPSXFIL DKA0:[REPORTS]RPSXREF.ISM
```

RPT – ReportWriter directory

The RPT environment variable specifies the directory that contains your ReportWriter distribution. It is required for normal operation of the ReportWriter program and utilities.

Value

The path, including the device, for the directory that contains the ReportWriter distribution files.

Setting location

- On Windows, the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).
- On UNIX, the **setsde** file.
- On OpenVMS, the **SYNERGY_STARTUP.COM** file (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

ReportWriter.

Examples

On UNIX,

```
RPT=/usr/synergy/rpt     ;export RPT
```

RPTDAT – ReportWriter data files directory

The RPTDAT environment variable defines the directory where ReportWriter data files are located.

Value

The path, including the device, for the directory that contains the ReportWriter data files.

Discussion

The default directory for ReportWriter data files is the RPTDAT subdirectory below the directory where the ReportWriter was installed. The RPTDAT directory contains the default report definition file, **reports.rpt**, and the window library file, **rptctl.ism**.

Setting location

- On Windows, the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).
- On UNIX, the **setsde** file.
- On OpenVMS, the **SYNERGY_STARTUP.COM** file (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

ReportWriter.

Examples

On OpenVMS,

```
$ DEFINE/SYS/EXEC RPTDAT DKA0:[SYNERGY.RPT.RPTDAT]
```

RPTDATE – Date input order

The RPTDATE environment variable specifies the input order for dates in ReportWriter. Setting RPTDATE affects the default date order that is used when dates are entered into ReportWriter input fields. Because it changes the UI Toolkit global variable **g_date_order**, it affects your application if you are using the ReportWriter's external subroutine interface.

Value

One of the following input orders for date fields:

1	DDMMYY
2	YYMMDD

Discussion

If RPTDATE is not set, the default input order for dates is MMDDYY.

If you are accessing ReportWriter from your application by external subroutine interface, using RPTDAT can affect the value of **g_date_order** in your application.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

ReportWriter.

Examples

On OpenVMS,

```
$ DEFINE RPTDATE 2
```

RPTEURO – European formatting

The RPTEURO environment variable specifies that European formatting conventions are applied to numeric values.

Value

One of the following values:

0	MMDDYY or MMDDYYYY format
1	DDMMYY or DDMMYYYY format
2	YYMMDD or YYYYMMDD format

All other values are treated as 0.

Discussion

If RPTEURO is set to any value, ReportWriter uses commas as decimal points and periods as separators in numeric values.

When using RPTEURO, you must place a space between the literals and the comma in a subscript specification. For example:

```
"FIELD[1 , 2]"
```

This rule applies to subscripts specified within ReportWriter input fields and report schemas.

We recommend using the Synergy Language LOCALIZE subroutine instead of RPTEURO in conjunction with ReportWriter's external subroutine interface. See LOCALIZE in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual*.

If the date is a period date, RPTEURO is ignored.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

ReportWriter.

Examples

On UNIX,

```
RPTEURO=1    ;export RPTEURO
```

RPTLIB – ReportWriter header file and shared library

The RPTLIB environment variable specifies the location of the ReportWriter header file (**reports.def**) and shared library (**synrpt.elb**, or **synrpt.exe** on OpenVMS).

Value

The directory in which the ReportWriter header file and shared library reside.

Discussion

RPTLIB should point to the rpt/lib directory under the directory where Synergy was installed.

Setting location

- On Windows, the [synergy] section of **synergy.ini**.
- On UNIX, the **setsde** file.
- On OpenVMS, the **SYNERGY_STARTUP.COM** file (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

ReportWriter.

Examples

On UNIX,

```
RPTLIB=/synergy/rpt/lib    ;export RPTLIB
```

RPTRFIL – ReportWriter report definition file

The RPTRFIL environment variable specifies the full path and filename of the ReportWriter report definition file.

Value

The full path and filename of the ReportWriter report definition file.

Discussion

If the environment variable RPTRFIL is not set, ReportWriter looks for the **reports.rpt** file in the path specified by RPTDAT. If RPTDAT is not set, Repository looks for the **reports.rpt** file in the current directory.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

ReportWriter.

Examples

On UNIX,

```
RPTRFIL=/usr/reports.ism    ;export RPTRFIL
```

RPTTUT – ReportWriter tutorial data directory

The RPTTUT environment variable specifies the directory that contains the ReportWriter tutorial files.

Value

The path for the directory in which the ReportWriter tutorial files are located.

Discussion

The repository associated with the tutorial uses RPTTUT to locate the example data files. Do not modify this setting. It is unset automatically when ReportWriter is exited.

Setting location

- On Windows, in the [rpt] section of **synergy.ini**. (This is set when ReportWriter is installed.)
- On UNIX, by **tutorial**.
- On OpenVMS, by **@tutorial.com**.

Used by

ReportWriter.

Examples

On Windows,

```
[rpt]
RPTTUT=%SYNERGYDE%rpt\tutor
```

RPTUSR – ReportWriter argument string

The RPTUSR environment variable specifies the argument string used when chaining or spawning ReportWriter.

Value

Refer to "Spawning and Chaining to ReportWriter" in the "Accessing ReportWriter from Your Application" chapter of the *ReportWriter User's Guide* for the syntax and discussion of the ReportWriter argument string.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

ReportWriter.

Examples

On UNIX,

```
RPTUSR="RPT:reports.rpt|myprog.dbr"    ;export RPTUSR
```

RSFILPATH – Default file path for *xf*Server

WIN

The RSFILPATH environment variable defines the default directory on the *xf*Server machine for files that do not have a path specification.

Value

The path on the server that you would like to use as the default file path.

Discussion

The path set with RSFILPATH is used as a default location for files without a path specification. The directory you specify must already exist and must have write permission. If RSFILPATH is not specified, the default directory is C:\Documents and Settings\All Users\Shared Documents (XP/2003) or C:\Users\Public\Public Documents (all other Windows platforms).

You can check the location of the default file path setting by running **synckusr** with the **-s** and **-p** options.

If you have a local account (not a domain-level account) on the server, you can use the Local Path setting for the Home Folder in the user's account profile as the location in which to put files without a path specification. This setting overrides both RSFILPATH and the default directory mentioned above. To enable this, you will need to set the registry setting ENABLEUSERHIVE to 1 in **HKEY_LOCAL_MACHINE\ SOFTWARE\Synergex\Synergy xfServer***service_name***\default** on the server machine.

Setting location

In the Windows registry. We recommend that you use the Synergy Configuration Program to set environment variables for *xf*Server. See "Defining environment variables" in the Windows section of the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Used by

*xf*Server.

See also

- "Using xfServer on Windows" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.
- "The synckusr Utility" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

RSLOGMAX – Maximum *xf*Server log file size

UNIX

The RSLOGMAX environment variable specifies the maximum log file size allocated by the Synergy/DE *xf*Server daemon, **rsynd**.

Value

The size of the log file in bytes.

Discussion

The minimum log file size is 1,024 bytes. The default log file size is 10,000 bytes. This is enough room to log approximately 100 *xf*Server starts and stops. Use RSLOGMAX to specify a larger log file size.

Use RSYNDLOG to specify a log file other than the default, **/usr/lib/rsynd.log**.

The RSLOGMAX environment variable must be set prior to starting **rysnd**. If **rsynd** is started from an **rc** system startup file, you must also set RSYNDLOG and/or RSLOGMAX there as well.

Setting location

On the server in the **synrc** file in the /etc directory (for all clients) or the **.synrc** file in the users $HOME directory (for specific users).

Used by

*xf*Server.

See also

"Using xfServer on UNIX" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Examples

The following example sets the maximum *xf*Server log file size to 20,000 bytes.

```
RSLOGMAX=20000
```

RSYNDLOG – Alternate *xf*Server log file

UNIX

The RSYNDLOG environment variable specifies an alternate log filename for the Synergy/DE *xf*Server daemon, **rsynd**.

Value

The filename to which *xf*Server information is logged.

Discussion

By default, Synergy/DE *xf*Server logs start and stop activity, along with unexpected signals, to the file **/usr/lib/rsynd.log**. You can specify an alternate log filename using RSYNDLOG.

Each log entry contains the following information:

- The entire path name of **rsynd**
- The date and time started
- The Synergy/DE *xf*Server version number
- The port number
- The pid
- Synergy/DE *xf*ServerPlus enablement (if specified)
- Secure/nonsecure/trusted status
- Monitor enablement (if specified)

By default, the maximum log file size is 10K. This is enough room to log approximately 100 *xf*Server starts and stops. When this size is exceeded, the file is truncated to 0 bytes and then refilled with new activity. Use the RSLOGMAX environment variable to specify a larger log file size.

RSYNDLOG must be set prior to starting **rsynd**. If **rsynd** is started from an **rc** system startup file, you must also set RSYNDLOG and/or RSLOGMAX there as well.

Setting location

On the server in the **synrc** file in the /etc directory (for all clients) or the **.synrc** file in the users $HOME directory (for specific users).

Used by

*xf*Server.

See also

"Using xfServer on UNIX" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Examples

```
RSYNDLOG=/usr/synergyde/dbl/rsynd.log    ;export RSYNDLOG
```

RUSER – Client/server user name

WIN, UNIX

The RUSER environment variable specifies a client/server user name.

Value

One of the following:

user_name[/password]	Pass the specified user name and optional password to the server.
SSPI	(Windows) Tell *xf*Server to use Windows authentication to impersonate the client connection.

Discussion

The RUSER environment variable is used to implement RUSER security with *xf*Server. See "Understanding xfServer security" in the "Configuring xfServer" chapter of the *Installation Configuration Guide* for a description of RUSER security. The value of the RUSER environment variable (or registry setting) defines the user credentials (user name and password) passed to *xf*Server via the client. The server session created impersonates that user. (See the "Configuring xfServer" chapter of the *Installation Configuration Guide* for more information on server modes.) When specifying a password, *xf*Server expects it to be in the form returned by the **setruser** utility.

On Windows, the most common way to set RUSER is in the registry via the **setruser** utility. RUSER set in the environment overrides RUSER set in the registry.

TIP

To test where RUSER is set on Windows, run the **synckusr** utility as follows:

```
synckusr -r
```

This command displays the user name and password in masked form (for example, \261j\263W\256j). More importantly, it specifies which RUSER setting the runtime client uses. For more information about **synckusr**, see "The synckusr Utility" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Set RUSER as follows:

On	Do this
Windows	To set RUSER in the registry, run **setruser** to create a Windows registry entry for the current user (in **HKEY_CURRENT_USER\Software\Synergex**) or **setruser -g** to set RUSER in a global registry location (**HKEY_LOCAL_MACHINE\Software\Synergex**) so that multiple users on a single machine all have access to the value. To set RUSER in the environment, run **setruser -n**, and set RUSER to the string that is generated: `RUSER="`*setruser_value*`"`
UNIX	Run the **setruser** utility, and set RUSER to the string that is generated: `RUSER="`*setruser_value*`" ;export RUSER` This command sets RUSER for the current log-in session. To set RUSER for every subsequent log-in session by this user, you can redirect the output from **setruser** to a file using the following command: `setruser >`*filename* Then, place the output in your **.profile** file as follows: `RUSER="`*setruser_value*`" ;export RUSER` (Make sure you include the quotation marks.)

For a list of options to the **setruser** utility, see "The setruser Utility" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

WIN

An RUSER value of SSPI (all uppercase) instructs *xf*Server to use Windows authentication. You can set RUSER to this value by specifying "SSPI" as the user name (with no password) when prompted by the **setruser** utility. The only time you might need to do this is if you need to override a global RUSER (registry) setting set with **setruser -g**. (Normally, however, you can just run your Windows server in Restricted mode.)

Setting location

The environment and/or the Windows registry on Windows and the environment on UNIX. If RUSER is set in both the **HKEY_LOCAL_MACHINE\Software\Synergex** and **HKEY_CURRENT_USER\Software\Synergex** locations of the registry, the **HKEY_CURRENT_USER** setting takes precedence. If RUSER is set in both the environment and the registry on Windows, the setting in the environment takes precedence.

When RUSER is set in the **Synrc** node in the registry, as is required when configuring *xf*ServerPlus for remote data access, it is read into the environment when **rsynd** starts up, and it is therefore considered to be set in the environment.

RUSER only needs to be set on the client.

Used by

Runtime, compiler, **fcompare**, **fconvert**, **isutl**.

See also

"Understanding xfServer security" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Examples

The following example on Windows runs the **setruser** utility, which sets RUSER for the current user in a Windows registry entry.

```
setruser
```

The following example on Windows passes the user name Mark and the password \231\251o\2451p\228x to the server. (You would first need to run **setruser -n** to display the encoded password string to the screen. See "The setruser Utility" in the "Configuring xfServer" chapter of the *Installation Configuration Guide* for more information.)

```
set RUSER=Mark/\231\251o\2451p\228x
```

The following example on UNIX passes the user name John, without a password, to the server.

```
RUSER=John    ;export RUSER
```

The following example on UNIX is an automated process that, as a result of using the grave accent characters (`), invokes **setruser**, assigns the output to RUSER, and exports it. (When **setruser** is invoked, it prompts you for a name and password. The name and password you type are immediately and automatically assigned to RUSER, with no cutting and pasting required.)

```
RUSER=`setruser`     ;export RUSER
```

SCRIPT_SH – SCRIPT_SH.EXE shared image

VMS

The SCRIPT_SH logical specifies the location of the shared image that contains the external routine interface to the Script compiler (%SCR_CLOSELIBRARY, %SCR_ERRORCOUNT, %SCR_OPENLIBRARY, and %SCR_PROCESS).

Value

The full path and filename of the **SCRIPT_SH.EXE** shared image.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

UI Toolkit.

Examples

```
$ DEFINE/SYS/EXEC SCRIPT_SH WND:SCRIPT_SH.EXE
```

SCSCOMPR – Client/server data compression

The SCSCOMPR environment variable enables data record compression between the server and clients.

Value

One of the following values:

1	Compression is requested.
0	Compression is not requested. (default)

Discussion

Compression can significantly improve the performance of read, write, and store operations on low-speed or busy networks, especially WANs. If you do not use SCSCOMPR, Synergy/DE *xf*Server will not compress transferred data records.

SCSCOMPR can be set on the server, on the client, or both. The system-specific sections below describe the various ways to set it and how the setting locations interact on each operating system.

To compress data sent to and from all clients, set SCSCOMPR to 1 on the server. Or, to compress data sent to and from individual clients, set SCSCOMPR to 1 on those clients only (and not on the server). To turn compression off, set SCSCOMPR to 0 (or don't set it at all) on both the client and the server. Setting SCSCOMPR to 0 in the client's environment does *not* turn off compression if it's requested on the server.

WIN

SCSCOMPR can be set in the registry on the server or in the environment on the client. On the Windows server, use the Synergy Configuration program to set SCSCOMPR as follows:

1. From the xfServer/xfServerPlus tab, click the Add xfServer Service button, or select a service from the list of services and click the Modify Service button. The xfServer Information dialog box is displayed.
2. Select the Compress data packets check box. This sets SCSCOMPR in the registry under **HKEY_LOCAL_MACHINE\SOFTWARE\Synergex\Synergy xfServer\***service_name***\Default** (for a specific instance of *xf*Server) or **HKEY_LOCAL_MACHINE\SOFTWARE\Synergex\Synergy xfServer\Default** (if you selected the <Default> entry).

UNIX

SCSCOMPR can be set in the **/etc/synrc** file or a client's **.synrc** file on the server, or in the environment on the client. Setting SCSCOMPR to 0 in the **.synrc** file in the home directory of the client's account on the server will override the setting in the master **/etc/synrc** file, and no compression request will be made to that client from the server. This is a way of setting compression for individual clients from the server. To turn compression completely off, set SCSCOMPR to 0 (or don't set it at all) in all server setting locations (**/etc/synrc** and **.synrc** for each client), as well as in the client's local environment.

VMS

SCSCOMPR can be set in the **SERVER_INIT.COM** file, the **DBLDIR:SYNRC.COM** file, or a client **SYNRC.COM** file. Setting SCSCOMPR to 0 in the **SYNRC.COM** file in the home directory of the client's account on the server will override the setting in the master **SERVER_INIT.COM** and **DBLDIR:SYNRC.COM** files, and no compression request will be made to that client from the server. This is a way of setting compression for individual clients from the server. To turn compression completely off, set SCSCOMPR to 0 (or don't set it at all) in all server setting locations (**SERVER_INIT.COM**, **DBLDIR:SYNRC.COM**, and **SYNRC.COM** for each client), as well as in the client's local environment.

Setting location

See the Discussion section above.

Used by

Runtime, *xf*Server.

See also

"Using xfServer on Windows," "Using xfServer on UNIX," and "Using xfServer on OpenVMS" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Examples

On Windows, if set at the command prompt,

```
set SCSCOMPR=1
```

SCSPORT – Client/server port

The SCSPORT environment variable specifies the client runtime port used to attach to a Synergy server.

Value

A valid port number on which a Synergy/DE *xf*Server is running.

Discussion

The default *xf*Server port is 2330. Set SCSPORT when you want to use a port other than 2330.

On OpenVMS, SCSPORT is only used by **synxfpng**.

Setting location

On the client, in the environment. On Windows, this environment variable can also be set in the [synergy], [dbl], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

SCSPORT is not set in the environment on the server. Instead, set the port number on UNIX by running **rsynd** with the **-p** option or on Windows using the Synergy Configuration Program.

Used by

Runtime, *xf*Server, **synxfpng**.

See also

- "Using xfServer on Windows" and "Using xfServer on UNIX" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.
- "The synxfpng Utility" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.

Examples

On UNIX,

```
SCSPORT=2331    ;export SCSPORT
```

SCSPREFETCH – Enable READS prefetch support for *xf*Server

The SCSPREFETCH environment variable enables READS prefetch support by defining the maximum amount of memory to use for the prefetch buffer.

Value

A number between 0 and 32, inclusive, that specifies the size of the prefetch buffer in kilobytes. Prefetching is off by default. Once prefetching is on, you can turn if off by setting SCSPREFETCH to 0.

Discussion

To improve sequential READS performance when using *xf*Server, a prefetch feature is available that enables the client to prefetch (or cache) sequential records for files of any type that are open in input mode, relative files that are open for update, or ISAM files that are open for update and use the LOCK:Q_NO_TLOCK option.

To enable or disable this feature on a file-by-file basis, regardless of the environment setting, use the SETLOG routine to set or clear SCSPREFETCH prior to the OPEN statement. The value of SCSPREFETCH is checked on every remote file open. (Note that after you open a file with SCSPREFETCH set, you cannot turn prefetching off on that channel until you close and reopen the file with SCSPREFETCH set to 0.)

We recommend that you set SCSPREFETCH to 4 initially and then change it if necessary.

If SCSPREFETCH is set to an invalid value, a default value of 4 will be used.

Setting location

On the client, the environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

*xf*Server.

See also

- READS in the “Synergy Language Statements” chapter of the *Synergy Language Reference Manual*.
- System option #55 on page 2-47.

Examples

On UNIX,

```
SCSPREFETCH=8    ;export SCSPREFETCH
```

SDE – Location of the main Synergy/DE directory

UNIX

The SDE environment variable specifies the root directory of the Synergy/DE files.

Value

The path for the directory in which the Synergy/DE product subdirectories reside (dbl, lm, connect, rps, rpt, and toolkit).

Setting location

The **setsde** script file (located in the synergyde directory).

Used by

Runtime, Connectivity Series, License Manager, Repository, ReportWriter, UI Toolkit.

Examples

```
set SDE=/usr/synergyde     ;export SDE
```

SDMS_AUDIT – Enable auditing of nonserver file operations

UNIX

The SDMS_AUDIT environment variable enables auditing of nonserver file operations and defines the pathname of the auditing file.

Value

The full path and filename of the auditing log file.

Setting location

The environment.

Used by

Synergy DBMS, *xf*ODBC.

Example

On UNIX,

```
set SDMS_AUDIT=sdmsa.log    ;export SDMS_AUDIT
```

SDMS_AUDIT_FILENAME – Limit audit output to specified file

WIN, UNIX

The SDMS_AUDIT_FILENAME environment variable filters the audit log by limiting it to operations made on a specific file. Up to three concurrent opens of this file are supported.

Value

The full path and filename of the file to be audited.

Setting location

The environment.

Used by

Synergy DBMS, *xf*ODBC.

Example

In the environment on Windows,

```
set SDMS_AUDIT_FILENAME=DBLDIR:myfile.dbl
```

SDMS_AUDIT_FLUSH – Flush entries for log events

WIN, UNIX

The SDMS_AUDIT_FLUSH environment variable specifies that entries will be flushed for each log event.

Value

Any value.

Discussion

This is an option when the SDMS_AUDIT environment variable is used.

Setting location

The environment.

Used by

Synergy DBMS.

Examples

On UNIX,

```
set SDMS_AUDIT_FLUSH=1     ;export SDMS_AUDIT_FLUSH
```

SDMS_AUDIT_FULL – Log additional audit information

WIN, UNIX

The SDMS_AUDIT_FULL environment variable specifies that additional information should be logged for each file operation.

Value

Any value.

Discussion

This is an option when the SDMS_AUDIT environment variable is used. It adds the first 50 bytes of each record to the log file.

Setting location

The environment.

Used by

Synergy DBMS, *xf*ODBC.

Examples

On UNIX,

```
set SDMS_AUDIT_FULL=1      ;export SDMS_AUDIT_FULL
```

SDMS_AUDIT_MODE – Log I/O modes when auditing

WIN, UNIX

The SDMS_AUDIT_MODE environment variable specifies that I/O control modes should be logged for each file operation.

Value

Any value.

Discussion

This is an option when the SDMS_AUDIT environment variable is used. It adds the I/O control modes to each logged entry.

Setting location

The environment.

Used by

Synergy DBMS, *xf*ODBC.

Examples

On UNIX,

```
set SDMS_AUDIT_MODE=1    ;export SDMS_AUDIT_MODE
```

SDMS_AUDIT_ROUTINE – Log routine name

WIN, UNIX

The SDMS_AUDIT_ROUTINE environment variable specifies that the name of the Synergy routine that executed the I/O statement, as well as the name of the routine that called the executing routine, should be logged for each file operation.

Value

Any value.

Discussion

This is an option when the SDMS_AUDIT environment variable is used, and it only applies when running the runtime. It adds the name of the routine that executed the I/O statement to the log file, and if that routine is not the main routine, it also adds the name of the routine's caller.

Setting location

The environment.

Used by

Runtime, Synergy DBMS.

Examples

On UNIX,

```
set SDMS_AUDIT_ROUTINE=1   ;export SDMS_AUDIT_ROUTINE
```

SDMS_AUDIT_SRV – Enable auditing of server file operations

WIN

The SDMS_AUDIT_SRV environment variable enables auditing of file operations on a server and defines the pathname of the auditing file.

Value

The full path and filename of the auditing log file.

Discussion

Use SDMS_AUDIT_SRV instead of SDMS_AUDIT when multiple threads are used on Windows systems.

Setting location

The environment.

Used by

Synergy DBMS, *xf*ODBC.

Examples

If set at the command prompt on Windows, the following example creates one audit file named **c:\temp\audit.log_*PID***, where *PID* is the process ID.

```
set SDMS_AUDIT_SRV=c:\temp\audit
```

SDMS2_FULL – Log additional ODBC calls to the database

The SDMS2_FULL environment variable causes additional ODBC calls to be recorded to a log file for support purposes.

Value

1.

Discussion

On Windows and UNIX machines, use this variable with SDMS_AUDIT or SDMS_AUDIT_SRV to provide ODBC-specific information.

Setting location

In the environment with system-wide logicals. If you need to use Synergy DBMS logging in a client/server configuration, set the logicals on the server.

Used by

Synergy DBMS, *xf*ODBC.

See also

- SDMS_AUDIT on page 1-181.
- SDMS_AUDIT_SRV on page 1-187.
- "Synergy DBMS logging" in the "Data Access Errors and Error Logging" chapter of the *xfODBC User's Guide*.

Examples

On OpenVMS,

```
$ DEF/SYS/EXE SDMS2_FULL 1
```

SDMS2_LOG – *xf*ODBC log file on OpenVMS

VMS

The SDMS2_LOG environment variable defines the path and name of the Synergy DBMS log file on OpenVMS systems for use with *xf*ODBC.

Value

The path and name of the Synergy DBMS log file.

Setting location

In the environment with system-wide logicals. If you need to use Synergy DBMS logging in a client/server configuration, set the logicals on the server.

Used by

*xf*ODBC.

See also

"Synergy DBMS logging" in the "Data Access Errors and Error Logging" chapter of the *xfODBC User's Guide*.

Examples

```
$ DEF/SYS/EXE SDMS2_LOG DEVICE:[DIRECTORY]FILE
```

SFWINIPATH – Location of synergy.ini file

WIN

The SFWINIPATH environment variable defines the location of **synergy.ini**.

Value

The path, including the device, for the directory that contains the **synergy.ini** file. *Path* must be the full path specification without any logicals. Do not include the filename (**synergy.ini**) in this path specification.

Discussion

If the directory specified by SFWINIPATH does not contain a **synergy.ini** file, or if SFWINIPATH is not defined, the default location is assumed. The default location of the **synergy.ini** file is synergyde\dbl. (However, since this is just a default file that will be removed during an upgrade or uninstallation, we recommend that you copy this file elsewhere and specify the path to the copied file with SFWINIPATH.)

This logic does not apply to the service runtimes (**dbs** and **dbssvc**). If the directory specified by SFWINIPATH does not contain a **synergy.ini** file, or if SFWINIPATH is not defined, the service runtimes do not read the file.

You can run the program **synckini** in the dbl\bin directory to locate the **synergy.ini** file that will be accessed.

Do not set SFWINIPATH in the **synergy.ini** file.

Setting location

The environment.

Used by

Compiler, runtime, linker, librarian, **isutl**, **fcompare**.

See also

"Synergy initialization files" on page 1-11 for more information about **synergy.ini**.

Examples

If set at the command prompt,

```
set SFWINIPATH=c:\synergy\user1
```

SHELL – Default shell

WIN, UNIX

The SHELL environment variable specifies the shell that the SHELL subroutine will use.

Value

The shell you want the SHELL subroutine to use.

Discussion

WIN

If this environment variable is set, the SHELL subroutine and the debugger use the shell that you specify in *shell*, rather than the default shell specified by COMSPEC. (For more information about COMSPEC settings, see your Windows documentation.)

UNIX

Common shells are **sh**, **csh**, and **ksh**.

Other UNIX tools, such as **vi**, also use this environment variable.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

On Windows, if set in the **synergy.ini** file,

```
[synergy]
SHELL=c:\winnt\system32\cmd.exe /c
```

or

```
SHELL=c:\windows\dosprmpt.pif /c
```

SIG_CORE – No signal trapping

WIN, UNIX

The SIG_CORE environment variable disables signal trapping. It is primarily used for system fault diagnosis.

Value

Any value.

Discussion

When SIG_CORE is set to any value, you'll get a system core dump on any signal that would normally be interpreted by the runtime, and you'll have to reset your terminal settings with the **stty** command.

To turn SIG_CORE off, issue the following command at the command line:

```
unset SIG_CORE
```

Setting location

The environment.

Used by

Runtime.

Examples

On UNIX,

```
SIG_CORE=on  ;export SIG_CORE
```

SODBC_CNVFIL – *xf*ODBC conversion setup file

The SODBC_CNVFIL environment variable indicates that an *xf*ODBC conversion setup file is to be used and specifies the path and name of the conversion setup file.

Value

The path and name of the conversion setup file for the system catalog you want to regenerate.

Discussion

You can specify a conversion setup file *before* you use **dbcreate** or DBA using the SODBC_CNVFIL environment variable. Alternatively, you can specify a conversion setup file *as* you use **dbcreate** or DBA via **dbcreate**'s **-i** command line option or the Conversion setup field in the Generate System Catalog window of DBA.

If SODBC_CNVFIL is set ahead of time, DBA and **dbcreate** automatically use the conversion setup file whenever you regenerate the system catalog. In addition, tables you delete in DBA are also marked for deletion in the conversion setup file.

SODBC_CNVFIL should not be set until the conversion setup file has been created.

Setting location

The environment. For client/server configurations, SODBC_CNVFIL must be set on the server. (The conversion setup file must also be on the server.)

Used by

*xf*ODBC, **fcompare**.

See also

- "Specifying a conversion setup file" in the "Preliminary Steps" chapter of the *xfODBC User's Guide*.
- "Generating the System Catalog" in the "Creating a System Catalog" chapter of the *xfODBC User's Guide*.
- "Generating and Editing a Conversion Setup File" in the "Viewing and Customizing the System Catalog" chapter of the *xfODBC User's Guide*.

Examples

In the environment on Windows,

```
set SODBC_CNVFIL=GENESIS_HOME:sodbccnv.ini
```

SODBC_CNVOPT – Ignore “Excluded by ReportWriter” Repository setting

The SODBC_CNVOPT environment variable includes all fields in the system catalog, regardless of their “Excluded by ReportWriter” settings in S/DE Repository.

Value

1.

Discussion

By default, if the “Excluded by ReportWriter” option in Repository is checked for a field, **dbcreate** does not include the field in the system catalog. To make the field available to ODBC-enabled applications, you can use Repository to clear this option. However, if you want *all* fields to be included in the generated system catalog, regardless of their “Excluded by ReportWriter” settings, set the SODBC_CNVOPT environment variable.

If a field is used as a structure tag or key segment, it is automatically included in the system catalog, regardless of the report exclusion flag or the SODBC_CNVOPT setting.

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

See also

“Setting catalog generation options” in the “Preliminary Steps” chapter of the *xfODBC User’s Guide*.

Examples

In the environment on Windows,

```
set SODBC_CNVOPT=1
```

SODBC_COLLAPSE – Reduce number of columns for ODBC-enabled applications

The SODBC_COLLAPSE environment variable compresses arrayed fields into a single column if the number of elements in a single field is greater than or equal to a limit you specify.

Value

A positive number that specifies the limit for the number of elements that can become individual columns.

Discussion

Use the SODBC_COLLAPSE environment variable if a generated table has more than 254 columns, a condition that might occur if the original structure contains multi-dimensional arrayed fields that have been mapped into separate element columns during system catalog generation. Some ODBC-enabled applications do not permit tables with more than 254 columns.

For example, if you decide that a field with more than 10 elements should be combined into one column, set SODBC_COLLAPSE to 10. If your structure has three fields—one with 6 elements, one with 10 elements, and one with 12 elements—the generated table has eight columns: 6 + 1 + 1. Because the second and third fields have reached the limit, each of these becomes a single column.

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

See also

"Setting catalog generation options" in the "Preliminary Steps" chapter of the *xfODBC User's Guide*.

Examples

In the environment on Windows,

```
set SODBC_COLLAPSE=10
```

SODBC_DBA – DBA location

The SODBC_DBA environment variable specifies the location of the *xf*ODBC Database Administrator (DBA) program.

Value

The directory that contains the DBA program.

Discussion

The filename for the DBA program on Windows and UNIX is **xfdba.dbr**. On OpenVMS, it's **XFDBA.EXE**.

SODBC_DBA is required and is automatically set when you install *xf*ODBC.

Setting location

- On Windows, the environment or the [synergy] section of **synergy.ini**.
- On UNIX, in the **setsde** script file (located in the synergyde directory).
- On OpenVMS, **CONNECT_STARTUP.COM**.
- For client/server configurations, SODBC_DBA must be set on the server.

Used by

*xf*ODBC.

See also

Appendix A of the *xfODBC User's Guide*

Examples

In the **synergy.ini** file,

```
SODBC_DBA=c:\synergyde\connect\synodbc\dba
```

SODBC_INIFIL – *xf*ODBC environment setup file

The SODBC_INIFIL environment variable specifies the name and path of the *xf*ODBC environment setup file.

Value

The path and filename of your *xf*ODBC environment setup file.

Discussion

An environment setup file is a file that you write. It typically has a **.ini** filename extension and is placed in the GENESIS_HOME directory, although these are not requirements.

SODBC_INIFIL is used by the *xf*ODBC driver when you connect to a database. It is not used when you create or modify a system catalog.

Setting location

The connect file or the environment. In a client/server configuration, SODBC_INIFIL must be set in the environment on the server. It is not set during installation. On Windows, SODBC_INIFIL can be set in **opennet.srv**.

Used by

*xf*ODBC, **fcompare**.

See also

"Setting Options and File Locations" in the "Preliminary Steps" chapter of the *xfODBC User's Guide*.

Examples

In the connect file,

```
SODBC_INIFIL=c:\synergyde\connect\synodbc\dat\envset.ini
```

SODBC_MCBA – Skip records with MCBA deleted-record characters

The SODBC_MCBA environment variable instructs *xf*ODBC to skip records that contain the MCBA deleted-record characters, which are four right brackets (]]]]) at the beginning or end of a record.

Value

Any value.

Discussion

By default, SODBC_MCBA is not set; *xf*ODBC does *not* skip records that contain the MCBA deleted-record characters. To instruct *xf*ODBC to skip records that contain the MCBA deleted-record characters, set the SODBC_MCBA environment variable to any value.

SODBC_MCBA is used by the *xf*ODBC driver when it accesses data. It is not used when you create or modify a system catalog.

Setting location

The connect file, the current environment, or the system-level. For client/server configurations, set SODBC_MCBA in the connect file on the server.

Used by

*xf*ODBC.

See also

"Recognizing the MCBA deleted-record characters" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*.

Examples

```
SODBC_MCBA=1
```

SODBC_NOGROUPNAME – Omit group and struct names from ODBC column names

The SODBC_NOGROUPNAME environment variable causes group and struct field names to be omitted from column names in the system catalog.

Value

Any value.

Discussion

By default, if a field is part of a group or struct field in the repository, the group or struct field name is added as a prefix to the field name to create the column name in the system catalog. Setting SODBC_NOGROUPNAME to any value supersedes this default.

Set SODBC_NOGROUPNAME only if you are certain the resulting column names will be unique.

If you're going to set this variable, make sure you do so before generating your system catalog.

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

See also

"Removing group and struct names from column names" in the "Preliminary Steps" chapter of the *xfODBC User's Guide*.

Examples

In the environment on Windows,

```
set SODBC_NOGROUPNAME=1
```

SODBC_NONULL – Set “Null allowed” property for system catalog columns

The SODBC_NONULL environment variable setting determines how the "Null allowed" property is set in the system catalog for repository fields whose "Null allowed" Repository value is Default (rather than Yes or No).

Value

One of the following values:

0	Set all columns to “null allowed.”
2	Set all columns to “null allowed” except those that are part of a primary key.
3	Set all columns to “null allowed” except those that are part of an access key.
4	Set all columns to “null allowed” except integer and binary columns and non-date columns that are part of the primary key.
Any other value	Set only date columns that aren’t part of a primary key to “null allowed.”

Discussion

By default, SODBC_NONULL is not set, which is the equivalent of setting it to 4. Set it before generating the system catalog.

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

See also

“Preventing null updates and interpreting spaces, zeros, and null values” in the “Preliminary Steps” chapter of the *xfODBC User’s Guide*.

Examples

In the environment on Windows,

```
set SODBC_NONULL=4
```

SODBC_NOUNSIGNED – Ignore the “Negative allowed?” Repository setting

The SODBC_NOUNSIGNED environment variable instructs **dbcreate** to ignore “Negative allowed?” Repository settings for fields when creating the system catalog.

Value

Any value.

Discussion

By default, SODBC_NOUNSIGNED is not set; **dbcreate** checks the “Negative allowed?” Repository setting to determine if the resulting column will be signed or unsigned.

If you set SODBC_NOUNSIGNED to any value, **dbcreate** sets all numeric fields to signed unless they have validation ranges that are limited to positive values (in which case the resulting columns will be unsigned). This matches the behavior for Connectivity Series versions prior to 8.3.

If you set this environment variable, set it before generating your system catalog.

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

Examples

In the environment on Windows,

```
set SODBC_NOUNSIGNED=1
```

SODBC_ODBCNAME – Use Repository “Alternate name” field

The SODBC_ODBCNAME environment variable causes *xf*ODBC to use the values in the Repository “Alternate name” field as column names.

Value

1.

Discussion

Not all Repository field names make good ODBC column names. As an alternative, you can use the value in the Repository “Alternate name” field (if one is specified) as the column name by setting the SODBC_ODBCNAME environment variable.

(Note that this is not related to the Repository ODBC table name option, which enables you to assign ODBC *table* names to file/structure combinations.)

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

See also

“Setting catalog generation options” in the “Preliminary Steps” chapter of the *xfODBC User’s Guide*.

Examples

In the environment on Windows,

```
set SODBC_ODBCNAME=1
```

SODBC_TMPOPT – Exclude temporary files from system catalog

The SODBC_TMPOPT environment variable causes *xf*ODBC to exclude tables attached to temporary files from the system catalog.

Value

1.

Discussion

By default, when **dbcreate** generates a system catalog, it includes tables that describe temporary files (that is, files for which the Repository Temporary flag is set). To omit tables that describe temporary files, set SODBC_TMPOPT.

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

See also

"Renaming columns for clarity" in the "Preliminary Steps" chapter of the *xfODBC User's Guide*.

Examples

In the environment on Windows,

```
set SODBC_TMPOPT=1
```

SODBC_TOKEN – Change the character used for arrayed fields and groups

The SODBC_TOKEN environment variable enables you to change the character that the **dbcreate** utility (or DBA) uses when it generates column names for an arrayed field or group.

Value

Any character that is valid as an SQL identifier character for your ODBC applications.

Discussion

When you generate a system catalog for a repository that has an arrayed field, each element in the arrayed field is mapped as a separate column with a name that consists of the array name, the element's position in the array, and pound signs (#), by default, to delineate position values. (For example, a [2,2] arrayed field with the name **myarray** will be mapped to the following: **myarray#1#1**, **myarray#1#2**, **myarray#2#1**, and **myarray#2#2**.) The same is true of groups that are arrays. See "Arrays" in the "Preliminary Steps" chapter of the *xfODBC User's Manual* for more information.

You can change the character used to delineate position values by setting SODBC_TOKEN to the character you want to use. For example, you can instruct **dbcreate** to use underscore (_) rather than #:

```
set SODBC_TOKEN=_
```

For the **myarray** field described above, this would result in the following columns: **myarray_1_1**, **myarray_1_2**, **myarray_2_1**, and **myarray_2_2**.

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

See also

"Changing the position delimiter used for arrays" in the "Preliminary Steps" chapter of the *xfODBC User's Manual*.

SODBC_USEFORMAT – Use decimal information in format string

The SODBC_USEFORMAT environment variable instructs **dbcreate** to generate implied-decimal system catalog columns from repository fields with format strings that include decimal points.

Value

1.

Discussion

If your repository has a field that is not an implied decimal but that has a format string with a decimal point, you can instruct **dbcreate** to use the decimal information in the format string to create an implied-decimal column in the system catalog. To do this, set the SODBC_USEFORMAT environment variable to 1 before generating the system catalog. For example, if SODBC_USEFORMAT is set to 1 and your repository has a **d5** field with an XXX.XX format string, the field will appear as a **d5.2** column in the system catalog.

Setting location

The environment where you run **dbcreate**.

Used by

*xf*ODBC.

See also

"Using decimal information in the repository format string" in the "Preliminary Steps" chapter of the *xfODBC User's Guide*.

Examples

In the environment on Windows,

```
set SODBC_USEFORMAT=1
```

SORTMEM – Memory used by SORT

WIN, UNIX

The SORTMEM environment variable specifies the amount of memory that is used by the SORT statement and the **isutl** utility.

Value

The number of kilobytes of memory to be used by the sort. This value must be in the range 512 to 32767. The default value is 1024 for the runtime and *xf*Server. The **isutl** default is 10240 for nonterabyte ISAM files and 20480 for terabyte ISAM files.

Discussion

SORTMEM is used by the SORT verb in the runtime, either locally or when both files specified to the SORT are remote. SORTMEM is used on the server when the sort is on the server. It is used by the runtime when the sort is local and not on a server.

Setting location

For a local file, in the environment or **synergy.ini** (Windows).

For **isutl**, in the environment.

On a Windows server, in the Windows registry. Use the Synergy Configuration Program, or set SORTMEM as follows:

- For a specific instance of *xf*Server, set SORTMEM under **HKEY_LOCAL_MACHINE\SOFTWARE\Synergex\Synergy xfServer\***servername***\synrc**.
- For a system-wide setting, set SORTMEM under **HKEY_LOCAL_MACHINE\SOFTWARE\Synergex\Synergy xfServer\synrc**.

On a UNIX server, in the **synrc** file in the /etc directory (for all clients) or the **.synrc** file in the users $HOME directory (for specific users).

Used by

Runtime, *xf*Server, **isutl** utility.

Examples

On UNIX,

```
SORTMEM=8192   ;export SORTMEM
```

SQLJUSTINTIME – Generate cursor status on error condition

The SQLJUSTINTIME environment variable enables cursor status to be generated when an operation fails.

Value

1.

Discussion

To determine why an operation fails, set SQLJUSTINTIME. This creates the **ssqlerr.log** file, which contains an open cursor listing. If **ssqlerr.log** already exists, new logging is appended to the file, which may create a very large file.

Setting location

The environment or **synergy.ini**. For client/server configurations, set SSQLLOG on the client.

Used by

SQL Connection.

Examples

```
set SQLJUSTINTIME=1
```

SSQLEXT – Enable detailed logging

The SSQLEXT environment variable enables detailed logging for use by Synergy/DE Developer Support.

Value

1.

Discussion

To contact Synergy/DE Developer Support, see "Product support information" on page xiv.

Setting location

The environment or **synergy.ini**. For client/server configurations, set SSQLLOG on the client.

Used by

SQL Connection.

Examples

```
set SSQLEXT=1
```

SSQLLOG – Log SQL Connection function calls

The SSQLLOG environment variable enables logging for SQL Connection operations.

Value

1.

Discussion

For information on the sequence of SQL operations for a Synergy application, set SSQLLOG. This creates the **ssqlx.log** file, which contains a log of %SSC_*xxx* function calls. If E_CIDSEL, E_CIDNOSEL, or E_NOCIDS errors occur, **ssql.log** also contains errors and a list of open cursors.

Setting location

The environment or **synergy.ini**. For client/server configurations, set SSQLLOG on the client.

Used by

SQL Connection.

Examples

```
set SSQLLOG=1
```

SYN_3D_TOOLBAR – Apply 3-D edges to toolbar buttons

WIN

The SYN_3D_TOOLBAR environment variable gives toolbar buttons 3-D edges whenever themes are active.

Value

1.

Discussion

In Windows, "hot" is the style applied to toolbar buttons that are under the mouse pointer. When set to **1**, SYN_3D_TOOLBAR applies the hot style to all toolbar buttons in UI Toolkit applications on Windows, even those that aren't under the mouse pointer. This gives the toolbar buttons 3-D edges.

Setting Location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

In the **synergy.ini** file,

```
[synergy]
SYN_3D_TOOLBAR=1
```

SYN_TRANSPARENCY_THRESHOLD – Transparent color range threshold

WIN

The SYN_TRANSPARENCY_THRESHOLD environment variable specifies a value that can be added to or subtracted from each RGB component of the transparent color to constitute an acceptable transparency range.

Value

The threshold above or below each RGB component of the transparent color to accept as within the transparency range. This value must be in hexadecimal notation, preceded by "0x", as follows:

0x*hh*

Discussion

This environment variable is primarily useful with JPEG images. Because JPEGs use a lossy compression algorithm, a pixel will not necessarily be returned in exactly the same color as it started out. Defining a transparency threshold can help eliminate transparency problems. For example, on a white background, a threshold of 0x10 can avoid white-looking blotches in the background. With the following settings

```
SYN_TRANSPARENT_COLOR=0xFF,0xFF,0xFF
SYN_TRANSPARENCY_THRESHOLD=0x10
```

Synergy/DE will accept colors as dark as 0xEF,0xEF,0xEF as transparent, which works well for JPEGs with a white background.

For non-JPEG images, a threshold is typically not needed, unless you are not certain about the exact background color (for instance, when images are scanned in or from a third-party source).

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit, Composer, Synergy Windows printing API.

Examples

In the **synergy.ini** file,

```
[synergy]
SYN_TRANSPARENCY_THRESHOLD=0x10
```

SYN_TRANSPARENT_COLOR – Color treated as transparent

WIN

The SYN_TRANSPARENT_COLOR environment variable specifies a color to be treated as transparent.

Value

The RGB triplet for the color you want to designate as transparent. This triplet must be in hexadecimal notation, separated by commas and preceded by “0x”, as follows:

0x*rr*,0x*gg*,0x*bb*

where *rr* is the value for red, *gg* is the value for green, and *bb* is the value for blue.

Discussion

Only one color can be designated as transparent, and the specified color will be transparent in all images that contain it. *Transparent* means that pixels that correspond to that color be left unpainted, so that they show whatever color was painted at that location “behind” the image (for example, the face color on a button, or whatever else may have been printed in the same location on a page by the Synergy Windows printing API). This feature accommodates for the fact that not all users use the same background colors for toolbar and other buttons.

If SYN_TRANSPARENT_COLOR is not set, or if it is not set correctly, transparency will not be imposed on any images.

You can only use the SETLOG routine to set the value of SYN_TRANSPARENT_COLOR before loading the first image. After the first image is loaded, you cannot use SETLOG.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit, Composer, Synergy Windows printing API.

Examples

In the **synergy.ini** file,

```
[synergy]
SYN_TRANSPARENT_COLOR=0xFF,0xFF,0xFF
```

SYN_ULIMIT – ulimit value

UNIX

The SYN_ULIMIT environment variable defines ulimit for the Synergy runtime and *xf*Server.

Value

One of the following:

- A resource limit specified as the number of 512-byte blocks (the standard notation used by ulimit)
- A resource limit specified as the number of kilobytes, megabytes, or gigabytes, in the format *number*Kb, *number*Mb, or *number*Gb, respectively
- **-1**, to set ulimit to unlimited

Discussion

To activate SYN_ULIMIT, you must add the setuid bit to the Synergy runtime and *xf*Server executables (**dbr**, **dbs**, and **rsynd**). This is no longer the default for a distributed runtime.

In order for the Synergy runtime or *xf*Server to change the ulimit value, **dbr** and **rsynd** must be owned by root and have the setuid bit set. To make root own **dbr** and **rsynd** and set the setuid bit, enter the following at the command line:

```
chown root dbr rsynd
chmod u+s dbr rsynd
```

If the setuid bit is not set, files created by the Synergy runtime or *xf*Server are limited to the system ulimit.

By default, when SYN_ULIMIT is activated, files created with the Synergy runtime or *xf*Server are limited to 204800 blocks (104 MB) or the system ulimit if it is higher.

Do not use SYN_ULIMIT if you use Connectivity Series. Instead, use your operating-system equivalent.

Setting location

The environment.

Used by

Runtime, compiler, linker, librarian, **fcompare**, **fconvert**, **isutl**.

Examples

In the example below, files are limited to 204800 512-byte blocks (which is 104,857,600 bytes, or 104 MB) or the system ulimit, if it is higher.

```
SYN_ULIMIT=204800  ;export SYN_ULIMIT
```

In the following example, files are limited to 104 MB, which is equivalent to the first example.

```
SYN_ULIMIT=104Mb   ;export SYN_ULIMIT
```

The example below sets an unlimited file limit.

```
SYN_ULIMIT=-1      ;export SYN_ULIMIT
```

SYNBACKUP – Enable backup mode feature

UNIX

The SYNBACKUP environment variable enables the backup mode feature, which means I/O update operations can be frozen so Synergy system backup can safely be performed.

Value

Any value.

Discussion

The SYNBACKUP environment variable is available for use by the **synbackup** utility, which enables you to back up your Synergy system without terminating all applications or processes accessing the files. On UNIX, the current backup mode (Pending, On, Off, or Not Allowed) is maintained in a shared memory segment on the system. The **synbackup** utility initializes and maintains this shared memory segment, as well as sets the backup mode to Pending, On, or Off. The base address of this memory segment is stored in the file **DBLDIR:synbackup.cfg**. So that the runtime does not have to repeatedly open this file, it first checks the environment variable SYNBACKUP. The runtime only opens the **synbackup.cfg** file to retrieve the base address if SYNBACKUP is set.

When SYNBACKUP is not set, the backup mode feature is disabled.

Setting location

The **setsde** file created by the installation, but it is commented out at installation and must be uncommented by the system administrator.

Used by

synbackup.

See also

“The synbackup Utility” in the “General Utilities” chapter of *Synergy Language Tools*.

Examples

```
SYNBACKUP=1
```

SYNBASEDATE – Base date for *xf*ODBC Julian day conversions

The SYNBASEDATE environment variable sets the base date for conversion of date fields with the *JJJJJJ* format.

Value

The date you want to use as a base date, in *YYYY-MM-DD* format. By default, SYNBASEDATE is set to 1752-09-14 (September 14, 1752).

Discussion

When you enter a date into a field with the *JJJJJJ* format, *xf*ODBC stores the date as the difference between the date and the SYNBASEDATE value. Changing this value changes the way dates are stored and read.

Once you've modified data in the database, do not change the SYNBASEDATE value or the database will be corrupted. In addition, if you use the Julian functions %JPERIOD or %NDATE, do *not* use SYNBASEDATE.

If you do not set this variable, *xf*ODBC uses the same base date as the Synergy runtime.

SYNBASEDATE does not affect the way system catalogs are generated.

Setting location

The connect file, the current environment, or the system-level. For client/server configurations, set SYNBASEDATE in the connect file on the server.

Used by

*xf*ODBC.

See also

"Setting the base date for Julian day conversions" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*.

Examples

```
SYNBASEDATE=1582-10-04
```

SYNBIN – Composer directory

WIN

The SYNBIN environment variable specifies the directory that contains Composer. Workbench uses this environment variable to locate Composer.

Value

The path for the directory in which Composer is installed.

Setting location

In the [synergy] section of the distributed **synergy.ini** file installed with Synergy/DE.

Used by

Workbench.

Examples

```
SYNBIN=%SYNERGYDE%composer
```

SYNCENTURY – Two-digit year used to determine default century

The SYNCENTURY environment variable defines a sliding window for Synergy/DE default century methods by specifying a two-digit year to be used as a defining point. Years prior to the specified year use one century, while years the same as or later than the specified year use a different century.

Value

A two-digit year in the range 00 – 99.

Discussion

UI Toolkit and ReportWriter enable you to define a method that is called whenever a default century is required. If no century method is registered but SYNCENTURY is defined, an internal century method is called. If SYNCENTURY is not defined or is invalid, Toolkit and *xf*ODBC use the current system century, while ReportWriter uses 19 as the century for all dates stored in YYMMDD, YYJJJ, and YYPP formats.

The default century method used by Toolkit, ReportWriter, and *xf*ODBC provides a sliding window whereby the default century for each side of the window is determined by SYNCENTURY and the current year. If the current two-digit year falls between 0 and SYNCENTURY, years between 0 and SYNCENTURY use the current system century, while years between SYNCENTURY (inclusive) and 99 use the previous century. If the current two-digit year falls between SYNCENTURY (inclusive) and 99, years between 0 and SYNCENTURY use the next century, while years between SYNCENTURY (inclusive) and 99 use the current system century.

The following table contains the translated four-digit year when SYNCENTURY is set to 50:

Current year	Year entered by user	Translated four-digit year
1998	0	2000
1998	49	2049
1998	50	1950
1998	99	1999
2000	0	2000
2000	49	2049
2000	50	1950
2000	99	1999

Selection and sorting of dates in YYMMDD, YYJJJ, or YYPP formats are not optimized if SYNCENTURY is set or if %RW_CENTURY_METHOD is specified.

Setting location

For ReportWriter and UI Toolkit:

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

For *xf*ODBC:

For standalone configurations, set SYNCENTURY in the connect file or in the environment. For client/server configurations, set it in the connect file on the server.

Used by

ReportWriter, UI Toolkit, *xf*ODBC, **fcompare**.

See also

- E_METHOD and %ECENTURY_METHOD in the "Environment Routines" chapter of the *UI Toolkit Reference Manual* for more information on providing your own century method for Toolkit.
- "Specifying a Century for Two-digit Years" in the "Customizing ReportWriter Routines" chapter of the *ReportWriter User's Guide* for information on providing your own century method for ReportWriter with %RW_CENTURY_METHOD.

Examples

On Windows, if set in the **synergy.ini** file,

```
[synergy]
SYNCENTURY=50
```

SYNCMPOPT – Compiler options for all compiles

The SYNCMPOPT environment variable specifies a list of compiler options that will be added to all compiles.

Value

One or more valid compiler options, separated by a space.

Discussion

SYNCMPOPT enables you to specify compiler options that are always used so you don't have to specify them manually each time you compile. These options will be inserted before any command line options entered on a single command line invocation of the compiler, or before each line in a list of commands that are either entered interactively or read from a redirected input file.

Setting location

The environment or the [synergy] section of **synergy.ini**.

Used by

Compiler.

Examples

If SYNCMPOPT is set on Windows as follows:

```
set SYNCMPOPT=-qnet -qalign
```

entering the command

```
dbl -l tt: source.dbl
```

causes the compiler to respond as if you had entered

```
dbl -qnet -qalign -l tt: source.dbl
```

If, however, you enter the command

```
dbl
```

and then enter the filename after the prompt, as follows:

```
DBL> source.dbl
```

the compiler responds as if you had entered the command

```
dbl -qnet -qalign source.dbl
```

SYNCSCOPT – C# compiler options

WIN

The SYNCSCOPT environment variable specifies a list of C# compiler options that will be used when building a Synergy .NET assembly with *xf*NetLink .NET.

Value

One or more valid C# compiler options, separated by a space.

Discussion

You can use SYNCSCOPT to add C# compiler options to the command line that *xf*NetLink .NET uses to build the Synergy assembly. Set SYNCSCOPT before running the batch file generated by **gencs** or, if you are using Workbench, before selecting Build > Build Assembly.

If you set SYNCSCOPT to an option that is already included on the generated command line, the option defined by SYNCSCOPT will be used.

See your C# documentation for a list of compiler options and their syntax.

Setting location

At the command prompt, in the global environment (i.e., from the System Properties dialog box), on the Workbench command line, or on the Open tab of the Project properties dialog box in Workbench.

Used by

Workbench, **gencs**.

See also

"Creating an Assembly in Workbench" and "Creating an Assembly from the Command Line" in the "Creating Synergy .NET Assemblies" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

If set at the command prompt,

```
set SYNCSCOPT=/baseaddress:0x123456 /warn:1
```

SYNDEFNS – Default namespace name

The SYNDEFNS environment variable specifies the name of the default namespace, which will be used for implicit imports or if no namespace is declared.

Value

The name of the namespace to use as the default.

Discussion

The namespace specified by SYNDEFNS will be used if your code does not declare a namespace.

System and **Synergex** (and namespaces nested within them) and **synglobal** are reserved namespace names.

If SYNDEFNS is not defined, **synglobal** is the default namespace.

If the SYNUSERDEF environment variable is set, it takes precedence over SYNDEFNS. The only effect SYNDEFNS has in this case is to assign routines that aren't explicitly members of a namespace to the defined namespace; it no longer implies that the namespace is implicitly imported. If no namespace is set in SYNDEFNS and prototypes exist for routines not explicitly in a namespace, they are added to the **synglobal** namespace. Make sure the file specified by SYNUSERDEF explicitly imports the **synglobal** namespace in this scenario.

Setting location

The environment or the [synergy] section of **synergy.ini**.

Used by

Compiler, Workbench, Synergy Prototype utility.

See also

SYNUSERDEF on page 1-236.

Examples

```
set SYNDEFNS = MyCompany

set SYNDEFNS = MyCompany.MyDivision
```

SYNDLOG – Alternate License Manager debug log file

UNIX

The SYNDLOG environment variable specifies an alternate location and name for the Synergy/DE License Manager debug log file, **synd.log**.

Value

The directory path and name of the log file you want to create. The default is **/usr/lib/synd.log**.

Discussion

If the SYNDLOG value exceeds 96 characters, **/usr/lib/synd.log** is used.

Setting location

The environment.

Used by

License Manager server.

See also

"Error logging" in the UNIX section of the "Configuring License Manager" chapter in the *Installation Configuration Guide*.

Examples

```
SYNDLOG=/usr/lib/logs/synd.log   ;export SYNDLOG
```

SYNERGY_NOLOCALE – Don't use local user's language settings

Setting SYNERGY_NOLOCALE stops the runtime from using the local user's language localization settings to determine valid uppercase and lowercase characters in the 8-bit range (above 128).

Value

Any value.

Discussion

Uppercase and lowercase characters are read from the local system LOCALE for the 8-bit (128–255) space. On UNIX and OpenVMS, no 8-bit characters are defined; you must define the LOCALE or set the LANG environment variable for your operating system to enable the correct multilanguage 8-bit characters. On Windows, LOCALE is an operating system definition (the ANSI international character set by default), and no action is necessary on your part unless you want pre–Synergy/DE 7.3.1a behavior. Prior to Synergy/DE 7.3.1a, A–Z and a–z with the 8th bit set were considered characters (although these rarely matched actual international characters). To revert to this behavior on any operating system, set SYNERGY_NOLOCALE.

You must set SYNERGY_NOLOCALE if you want to set DBLOPT=32 for two-byte characters.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy] or [dbr] section of **synergy.ini**.

Used by

Runtime.

Examples

```
SYNERGY_NOLOCALE=1         ;export SYNERGY_NOLOCALE
```

SYNERGYDE – Synergy/DE directory

WIN

The SYNERGYDE environment variable specifies where Synergy/DE resides.

Value

The Synergy/DE directory where Synergy/DE was installed.

Discussion

This setting is set by the Core Components installation. You should not change environment variables that are set by the system.

Setting location

The environment.

Used by

Runtime, Workbench, Composer.

Examples

```
SYNERGYDE=c:\synergyde\
```

SYNERGYDE32 – Synergy/DE directory for 32-bit installations

WIN

The SYNERGYDE32 environment variable specifies where Synergy/DE resides for a 32-bit installation.

Value

The Synergy/DE directory where Synergy/DE was installed.

Discussion

This setting is set by the 32-bit Core Components installation. This occurs when installing on either a 32-bit or a 64-bit operating system. You should not change environment variables that are set by the system.

Setting location

The environment.

Used by

Runtime, Workbench, Composer.

Examples

```
SYNERGYDE32=c:\synergyde\
```

SYNERGYDE64 – Synergy/DE directory for 64-bit installations

WIN

The SYNERGYDE64 environment variable specifies where Synergy/DE resides for a 64-bit installation.

Value

The Synergy/DE directory where Synergy/DE was installed.

Discussion

This setting is set by the 64-bit Core Components installation. You should not change environment variables that are set by the system.

Setting location

The environment.

Used by

Runtime, Workbench, Composer.

Examples

```
SYNERGYDE64=c:\synergyde64\
```

SYNERGYDE$ROOT – Root directory for Synergy/DE files

VMS

SYNERGYDE$ROOT is a "rooted" logical that points to the root directory of the location of the Synergy/DE files.

Value

The path, including the device, for the directory that contains the Synergy/DE files.

Discussion

Every Synergy logical is based in some way or another on the SYNERGYDE$ROOT logical.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER.

Used by

Every Synergy product and utility that is installed on OpenVMS uses SYNERGYDE$ROOT in some way.

Examples

```
$ DEFINE/TRANS=CONCEALED SYNERGYDE$ROOT DKA0:[SYNERGYDE.]
```

SYNEXPDIR – Export directory for prototype files

The SYNEXPDIR environment variable specifies the directory to which the Synergy Prototype utility will export prototypes if no directory is specified at the **dblproto** command line.

Value

The path for the directory to which the Synergy Prototype utility will export prototype files.

Discussion

If SYNEXPDIR is not defined, the Synergy Prototype utility will export prototypes to the current directory.

Setting location

The environment or the [synergy] section of **synergy.ini**.

Used by

Synergy Prototype utility.

Examples

```
SYNEXPDIR=c:\mydir\prototypes\
```

SYNIMPDIR – Directories to search for import files

The SYNIMPDIR environment variable specifies the default directories to search when importing prototypes if a directory is not specified in the IMPORT statement.

Value

One or more directory paths, separated by a comma.

Discussion

When importing prototypes, all specified directories are searched in order of their appearance within the SYNIMPDIR environment variable. This list is searched after any directories specified by the **-qimpdir**=*import_dir* compiler option.

TIP

SYNIMPDIR makes it possible to create prototype files that have the same name (for example, **MyNamespace.dbp**) in different directories. If you set SYNIMPDIR to point to all of those directories, the statement

```
import MyNamespace
```

will go through each directory, picking up prototype files for that namespace.

In order for Workbench to see imported prototypes, SYNIMPDIR must be set.

Setting location

The environment or the [synergy] section of **synergy.ini**.

Used by

Compiler, Workbench, Synergy Prototype utility.

See also

IMPORT in the "Synergy Language Statements" chapter of the *Synergy Language Reference Manual*.

Examples

```
SYNIMPDIR=c:\mydir\,d:\mydir
```

SYNNET_DEBUG – Enable debugging for the Synergy .NET assembly API

WIN

The SYNNET_DEBUG environment variable enables debugging for the Synergy .NET assembly API.

Value

If the SYNNET_DEBUG environment variable is set to a value of 1 when the first assembly is loaded, debugging will be enabled.

Discussion

When debugging is enabled, debug messages are logged using the OutputDebugString Windows API function. You can view these messages using Visual Studio, other compatible C-level debuggers, or the **debugview** program from **http://www.sysinternals.com**.

Setting location

Inherited from the parent process environment; in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file); or set explicitly using the SETLOG routine before loading the first assembly.

Used by

Synergy .NET assembly API.

See also

"Debugging" in the introduction to the "Synergy .NET Assembly API" chapter of the *Synergy Language Reference Manual*.

Examples

```
SYNNET_DEBUG=1
```

SYNRPT – Location of the SYNRPT.EXE shared image

VMS

The SYNRPT logical points to **RPTLIB:SYNRPT.EXE** for use in activating the shared image that contains the ReportWriter shared library.

Value

The full path and filename of the **SYNRPT.EXE** shared image.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

ReportWriter.

Examples

```
$ DEFINE/SYS/EXEC SYNRPT RPTLIB:SYNRPT.EXE
```

SYNSSL_RAND – File containing random data for Synergy SSL encryption

Value

The path and name of a text file or an entropy-gathering device (if available on your system).

Discussion

When encrypting sensitive data, Synergy DBMS uses random data to ensure that the data is secure, because random data helps prevent hackers from guessing patterns. For most systems, this random data can be gathered from recognized system entropy devices or from the screen itself, or from a temporary file filled with random logic. However, on some systems, these methods do not generate enough random data to seed cryptographic algorithms, and a "Cannot load random state" error is generated. To eliminate this error, you can define the SYNSSL_RAND environment variable to point at a file that will be used (as a last resort) to gather random data when encryption occurs.

Setting location

The environment on the client and/or the server. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

*xf*Server, *xf*ServerPlus.

See also

- "Using Encryption" in the "Configuring xfServer" chapter of the *Installation Configuration Guide*.
- "Using Encryption" in the "Configuring and Running xfServerPlus" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

On Windows,

```
set SYNSSL_RAND=c:\windows\random.txt
```

On UNIX,

```
SYNSSL_RAND=/etc/entropy   ;export SYNSSL_RAND
```

SYNSSLLIB – Synergy SSL runtime support library

VMS

The SYNSSLLIB logical specifies the location of the Synergy SSL runtime support library (**synssllib.exe**).

Value

The full path and filename of the **SYNSSLLIB.EXE** library.

Discussion

SYNSSLLIB is used for network and data encryption. SYNSSLLIB is set by the installation.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

Runtime, *xf*Server, *xf*ServerPlus.

Examples

```
$ define/sys/exec SYNSSLLIB SYNERGYDE$ROOT:[DBL.BIN]SYNSSLLIB.EXE
```

SYNTXT – Message text file directory

The SYNTXT environment variable points to the directory where the **syntxt** ISAM file can be found.

Value

The directory that contains **syntxt.ism**.

Discussion

SYNTXT points to the directory that contains the message text file **syntxt.ism**. If this environment variable is not set, then the default directory is the one pointed to by DBLDIR.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

Examples

On Windows,

```
SYNTXT=s:\synergy\german
```

SYNUSERDEF – File to include at beginning of compilation unit

The SYNUSERDEF environment variable specifies a file to implicitly include at the beginning of a compilation unit.

Value

The file specification to include. The path can be full or relative. The default extension is **.dbl**.

Discussion

SYNUSERDEF is useful if you want to import multiple namespaces into a project without adding explicit IMPORT statements to your source code. It also enables you to define identifiers that will apply across all files in a compilation unit. First create a file that contains the desired IMPORT statements, .DEFINE directives, etc., and store the file in an accessible location. Set SYNUSERDEF to the path and name of this file, and then compile your project.

Whereas SYNDEFNS only allows you to implicitly import one namespace, SYNUSERDEF enables you to import many at the same time. If both SYNUSERDEF and SYNDEFNS are set, SYNUSERDEF takes precedence and no other implicit imports occur.

Setting location

The environment or the [synergy] section of **synergy.ini**.

Used by

Compiler, Workbench, Synergy Prototype utility.

See also

SYNDEFNS on page 1-222.

Examples

```
set SYNUSERDEF = c:\MyFiles\contacts.def
```

SYNXML – SYNXML.EXE shared image

VMS

The SYNXML logical specifies the location of the shared image that contains the Synergy XML API library.

Value

The full path and filename of the **SYNXML.EXE** shared image.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

Synergy XML API.

Examples

```
$ DEFINE/SYS/EXEC SYNXML DBLDIR:SYNXML.EXE
```

TABSET_STYLE – Tab set style

WIN

The TABSET_STYLE environment variable sets the initial style for all tabbed dialogs.

Value

(optional) One or more of the following styles, separated by commas: (**n**)

MULTILINE	Multiple lines of tabs
VERTICAL	Left or right side
BOTTOM	Bottom or right
RIGHT	Bottom or right

If TABSET_STYLE is set to nothing, the default style is used.

Discussion

If you're using the SETLOG subroutine to set TABSET_STYLE, do so prior to calling the Toolkit U_START subroutine.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

UI Toolkit.

See also

- The DTS_DEFAULT subfunction of %TS_TABSET in the "Tab Set Routines" chapter of the *UI Toolkit Reference Manual* for setting the default tab set style at runtime.
- The DTS_CREATE subfunction of %TS_TABSET in the "Tab Set Routines" chapter of the *UI Toolkit Reference Manual* for setting the tab set style per tab set and for descriptions of the various tab set styles.

Examples

The following example sets the global default tab set style to contain multiple rows of vertical tabs, on the right side of the tabbed dialog.

```
[synergy]
TABSET_STYLE=MULTILINE, VERTICAL, RIGHT
```

TBUF – Terminal output buffer size

UNIX

The TBUF environment variable specifies the size of the terminal output buffer.

Value

The size of the output buffer in bytes. It must be a value greater than 0 but less than 2048.

Discussion

The default value is 128 bytes. Setting TBUF to a value greater than 128 may result in longer delays between updates.

TBUF only works if system option #12 is set with DBLOPT.

Setting location

The environment.

Used by

Runtime.

Examples

```
TBUF=256  ;export TBUF
```

TERM – Terminal type

The TERM environment variable specifies the terminal type to use.

Value

The type of terminal you are using.

Discussion

WIN

If you are using UI Toolkit, TERM should be set to **MSWINDOWS**. However, if neither DTKTERM nor TERM are set, the Toolkit defaults to MSWINDOWS.

UNIX

You must set this environment variable if you are using UI Toolkit or the windowing API. Other UNIX tools, such as **vi**, also use this environment variable.

Synergy Language looks in the directory **/usr/lib/terminfo/*** or in the file **/etc/termcap** for your terminal type. The terminal type for your process may already be set. If it is, make sure that it's a valid type. If you don't set TERM or if you set it to a nonexistent terminal type, Synergy Language returns an error at startup (unless system option #30 is set with the DBLOPT environment variable).

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

TERM can be reset by the SETLOG subroutine, and the runtime interprets the new setting.

Used by

Runtime.

See also

"Customizing Key Mapping for Menu Shortcuts" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for more information about terminal types.

Examples

On Windows, if set at the command prompt,

```
set TERM=mswindows
```

TERMCAP – Termcap file

UNIX

When **termcap** is linked into the Synergy runtime, the TERMCAP environment variable either determines the file to use instead of **/etc/termcap** or is set to the **termcap** entry itself.

Value

The name of either a **termcap** file or a **termcap** entry.

Setting location

The environment.

Used by

Runtime.

Examples

```
TERMCAP=/usr/dbl/ansi.tc  ;export TERMCAP
```

TKLIB_SH – TKLIB_SH.EXE shared image

VMS

The TKLIB_SH logical specifies the location of the shared image that contains the UI Toolkit libraries.

Value

The full path and filename of the **TKLIB_SH.EXE** shared image.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

UI Toolkit.

Examples

```
$ DEFINE/SYS/EXEC TKLIB_SH WND:TKLIB_SH.EXE
```

TNMBR – Current terminal number

The TNMBR environment variable specifies the current terminal number.

Value

The number you want the TNMBR subroutine to return for your terminal.

Discussion

It is your responsibility to ensure that you don't give the same terminal number to more than one terminal.

WIN

If you don't set TNMBR, the runtime uses the default value. In all Windows environments, this value is 0.

UNIX

Avoid setting TNMBR to 254 or 255 when using SEND and RECV.

When the service runtime (**dbs** or **dbssvc**) is used, TNMBR defaults to -1.

Setting location

The environment. On Windows, this environment variable can also be set in the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

TNMBR can be reset by the SETLOG subroutine, and the runtime interprets the new setting.

Used by

Runtime.

Examples

On OpenVMS,

```
$ DEFINE TNMBR 1
```

TRIM_HOME – *xf*ODBC system catalog caching file

The TRIM_HOME environment variable specifies the location of the **trim.ini** (UNIX only) and **trim.msg** files. The **trim.ini** file specifies the amount of space allocated for each shared memory segment used by the system catalog cache. The **trim.msg** file supplies error text for the **syngenload** program.

Value

The directory with a lib subdirectory that contains either **trim.msg** (on Windows and OpenVMS) or both **trim.ini** and **trim.msg** (on UNIX).

Discussion

The Connectivity Series installation sets this environment variable. Do not change this setting.

Setting location

The **opennet.std** file, which must be on the server.

Used by

SQL OpenNet and *xf*ODBC.

See also

"System Catalog Caching" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*.

TYPE_FACE – Font name

WIN

The TYPE_FACE environment variable sets the typeface of the global font if a global font is not otherwise specified.

Value

The name of the desired Windows font (or typeface).

Discussion

Synergy/DE on Windows uses the specified typeface for application windows less than 132 columns. It is also used by Synergy Language and the debugger.

If you want to specify a new font using TYPE_FACE, it should be a font that comes in standard Windows packages.

You may want to use a fixed font because the character columns will always have the same alignment from row to row (as they would on a VT-100 or other text terminal), and text positioning will remain consistently aligned. We recommend that you use a fixed font when doing non-Toolkit processing.

Another style consideration when choosing a typeface is whether you want a serif or a sans-serif font. Serifs are the little strokes (or "feet") at the ends of a letter's main strokes. A sans-serif style does not have these ending strokes. Serif typefaces are generally recommended for larger bodies of text because the serifs on each letter actually help guide the eyes. Sans-serif typefaces work well for short phrases, headings, and small amounts of text.

This is an example of a sans-serif typeface.

This is an example of a serif typeface.

To find a list of available typefaces, open the Character Map utility (in the Accessories group). The typeface names are case sensitive.

We recommend that you use the FONT_GLOBAL environment variable rather than TYPE_FACE.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

See also

- "Using Fonts on Windows" in the "Customizing UI Toolkit" chapter of the *UI Toolkit Reference Manual* for the hierarchy used to determine the global font.
- FONT_GLOBAL on page 1-91.

Examples

In the **synergy.ini** file,

```
[synergy]
TYPE_FACE=Terminal
```

UMASK – File mode creation mask

UNIX

The UMASK environment variable sets the file mode creation mask (umask) for *xf*Server and *xf*ServerPlus.

Value

An octal umask value you want to set. Refer to the octal form of umask(1) in the UNIX man pages for details.

Discussion

Some security implementations cause the umask value to be cleared. You must use the UMASK environment variable in those instances.

On SCO, you must always set UMASK; otherwise, files created by *xf*Server and by programs started by *xf*ServerPlus will be created with permissions for read and write by owner only.

Setting location

The **synrc** file or the environment. If set in the environment, UMASK must be set before *xf*Server or *xf*ServerPlus is started. Setting it in the **synrc** file is the better choice because it guarantees that the UMASK setting will be used.

Used by

*xf*Server, *xf*ServerPlus.

Examples

The example below creates a file **-rw-rw-r--** (read and write by owner, read and write by group, and read only by others).

```
UMASK=02      ;export UMASK
```

VFYCTL – Change ismvfy behavior

WIN, UNIX

The VFYCTL environment variable changes the default behavior of the **ismvfy** utility.

Value

NO_DSCAN.

Discussion

Setting VFYCTL to “NO_DSCAN” tells **ismvfy** not to scan the data file before verifying the index. Normally, **ismvfy** prescans the data file to determine if the data file is capable of being recovered.

Setting location

The environment.

Used by

Ismvfy utility.

Examples

On UNIX,

```
VFYCTL=NO_DSCAN    ;export VFYCTL
```

VORTEX_API_LOGFILE – Connectivity Series client-side log file

The VORTEX_API_LOGFILE environment variable turns on logging and specifies the path and name of the file that will log statements issued to the database by the *xf*ODBC driver or SQL Connection.

Value

The path and name of the Vortex API log file you want to produce. You do not need to specify a filename extension.

Discussion

Vortex API logging is helpful in debugging queries. By recording Vortex API calls made by the *xf*ODBC driver on a Windows client, you can see the exact SQL statement issued to the database, debug SQL statement errors, and verify optimization.

Use the VORTEX_API_LOGOPTS environment variable to set options for the log file. If you set VORTEX_API_LOGFILE without setting VORTEX_API_LOGOPTS, the log file includes a list of all operations along with a total count for each operation.

Once you have successfully logged the error, turn logging off by unsetting the VORTEX_API_LOGFILE and VORTEX_API_LOGOPTS environment variables (and reboot if necessary). Logging slows performance, and the log files can quickly fill your disk.

Setting location

The environment. We recommend that you use log files to debug in a stand-alone configuration. If you need to use Vortex API logging in a client/server configuration, set the environment variables on the client. For services such as web servers that use the *xf*ODBC driver, you can use the Env. variables field in the xfODBC Setup window to set this environment variable on the client.

Used by

*xf*ODBC, SQL Connection.

See also

- VORTEX_API_LOGOPTS on page 1-250.
- "Vortex API logging" in the "Data Access Errors and Error Logging" chapter of the *xfODBC User's Guide*.
- "Troubleshooting and Error Logging" in the "Error Logging and Messages" chapter of the *SQL Connection Reference Manual*.

Examples

```
VORTEX_API_LOGFILE=c:\vortex
```

VORTEX_API_LOGOPTS – Connectivity Series client-side log options

The VORTEX_API_LOGOPTS environment variable specifies options for the file determined by VORTEX_API_LOGFILE.

Value

One or more of the following, separated by a plus sign (+):

APPEND	Append logging information to existing log file.
ERROR	Log only statements with errors.
FULL	Perform full logging.
PLAY	Set an option that enables Synergy/DE Developer Support to play back an operation.
RECORD	Log data for support.
TIME	Log execution time for statements.

Discussion

Vortex API logging is helpful in debugging queries. By recording Vortex API calls made by the *xf*ODBC driver and SQL Connection, you can see the exact SQL statement issued to the database, debug SQL statement errors, and verify optimization.

Use the VORTEX_API_LOGFILE environment variable to specify the name and location of the log file that is generated. If you set VORTEX_API_LOGFILE without setting VORTEX_API_LOGOPTS, the log file includes a list of all operations along with a total count for each operation.

Once you have successfully logged the error, turn logging off by unsetting the VORTEX_API_LOGFILE and VORTEX_API_LOGOPTS environment variables (and reboot if necessary). Logging slows performance, and the log files can quickly fill your disk.

Setting location

The environment. For services such as web servers that use the *xf*ODBC driver, you can use the Env. variables field in the xfODBC Setup window to set this environment variable on the client.

Used by

*xf*ODBC, SQL Connection.

See also

- VORTEX_API_LOGFILE on page 1-249.
- "Vortex API logging" in the "Data Access Errors and Error Logging" chapter of the *xfODBC User's Guide*.
- "Troubleshooting and Error Logging" in the "Error Logging and Messages" chapter of the *SQL Connection Reference Manual*.

Examples

To record a full operation log and CPU time,

```
VORTEX_API_LOGOPTS=FULL+TIME
```

VORTEX_HOME – SQL Connection default settings file

The VORTEX_HOME environment variable specifies the location of **net.ini**, the SQL Connection default settings file.

Value

The directory that contains a lib subdirectory ([.lib] on OpenVMS) that contains the **net.ini** file.

Discussion

SQL Connection looks for **net.ini** in the current directory or in **VORTEX_HOME/lib**. The Connectivity Series installation (Windows), setsde (UNIX), or SYS$MANAGER:CONNECT_STARTUP.COM (OpenVMS) sets this environment variable. Do not change this setting.

Setting location

The **opennet.std** file, which must be on the server.

Used by

SQL Connection.

See also

"Using network initialization files to set network defaults" in the "Welcome to SQL Connection" chapter of the *SQL Connection Reference Manual*.

VORTEX_HOST_HIDEGPF – Prevent failed thread from stopping SQL OpenNet server

WIN

When set, the VORTEX_HOST_HIDEGPF environment variable prevents an SQL OpenNet server from shutting down when a thread fails.

Value

Any value.

Discussion

The Connectivity Series installation automatically sets this environment variable. We don't recommend changing this setting.

Setting location

The **opennet.std** file, which must be on the server.

Used by

*xf*ODBC, SQL Connection.

Examples

```
VORTEX_HOST_HIDEGPF=1
```

VORTEX_HOST_LOGFILE – SQL OpenNet log file

The VORTEX_HOST_LOGFILE environment variable turns on logging and specifies the path and name of the log file used to log statements passed to SQL OpenNet from the *xf*ODBC driver and SQL Connection.

Value

The path and name of the Vortex host log file you want to produce. You do not need to specify a filename extension.

Discussion

VORTEX_HOST_LOGFILE specifies the file that logs calls and errors generated through SQL OpenNet server. You can use it to determine the driver API calls made between the driver client and the OpenNet server.

Vortex host logging applies only to a client/server configuration. You can use Vortex host logs only with **vtxnet2**. **Vtxnetd** (on Windows) does not support the VORTEX_HOST_LOGFILE environment variable.

Use the VORTEX_HOST_LOGOPTS environment variable to set options for the log file. If you set VORTEX_HOST_LOGFILE without setting VORTEX_HOST_LOGOPTS, the log file includes a list of all operations along with a total count for each operation.

Once you have successfully logged the error, turn logging off by unsetting the VORTEX_HOST_LOGFILE and VORTEX_HOST_LOGOPTS environment variables (and reboot if necessary). Logging slows performance, and the log files can quickly fill your disk.

Setting location

The server, as follows:

- On Windows, the **opennet.srv** file before starting **vtxnet2**.
- On UNIX, the environment before starting **vtxnetd**.
- On OpenVMS, a system-wide logical before starting the server program.

Used by

*xf*ODBC, SQL Connection.

See also

- VORTEX_HOST_LOGOPTS on page 1-256.
- "Vortex host logging" in the "Data Access Errors and Error Logging" chapter of the *xfODBC User's Guide*.
- "Troubleshooting and Error Logging" in the "Error Logging and Messages" chapter of the *SQL Connection Reference Manual*.
- The "Configuring Connectivity Series" chapter in the *Installation Configuration Guide* for information on **vtxnetd** and **vtxnet2**.

Examples

```
VORTEX_HOST_LOGFILE=c:\vortex
```

VORTEX_HOST_LOGOPTS – SQL OpenNet log options

The VORTEX_HOST_LOGOPTS environment variable specifies options for the file determined by VORTEX_HOST_LOGFILE.

Value

One or more of the following, separated by a plus sign (+):

FULL	Perform full logging.
ERROR	Log only statements with errors.
TIME	Log execution time for statements.
RECORD	Log data for support.
PLAY	Set an option that enables Synergy/DE Developer Support to play back an operation.

Discussion

Vortex host logging applies only to a client/server configuration.

Use the VORTEX_HOST_LOGFILE environment variable to specify the name and location of the log file that is generated. If you set VORTEX_HOST_LOGFILE without setting VORTEX_HOST_LOGOPTS, the log file includes a list of all operations along with a total count for each operation.

Once you have successfully logged the error, turn logging off by unsetting the VORTEX_HOST_LOGFILE and VORTEX_HOST_LOGOPTS environment variables (and reboot if necessary). Logging slows performance, and the log files can quickly fill your disk.

Setting location

The server, as follows:

- On Windows, the **opennet.srv** file before starting **vtxnet2** or **vtxnetd**.
- On UNIX, the environment before starting **vtxnetd**.
- On OpenVMS, the system-wide environment before starting the server program.

Used by

*xf*ODBC, SQL Connection.

See also

- VORTEX_HOST_LOGFILE on page 1-254.
- "Vortex host logging" in the "Data Access Errors and Error Logging" chapter of the *xfODBC User's Guide*.
- "The vtxnetd and vtxnet2 Programs" in the "Configuring Connectivity Series" chapter in the *Installation Configuration Guide* for information on **vtxnetd** and **vtxnet2**.
- "Troubleshooting and Error Logging" in the "Error Logging and Messages" chapter of the *SQL Connection Reference Manual*.

Examples

To record a full operation log and CPU time,

```
VORTEX_HOST_LOGOPTS=FULL+TIME
```

VORTEX_HOST_SYSLOG – Generate system messages for fatal SQL OpenNet errors

The VORTEX_HOST_SYSLOG environment variable instructs the SQL OpenNet server to generate messages for the event log (Windows), **syslog** (UNIX), or the operator console (OpenVMS) when an attempt to connect to an SQL OpenNet server causes fatal errors.

Value

Any value.

Discussion

The Connectivity Series installation automatically sets this environment variable. We don't recommend changing this setting.

Setting location

The server, as follows:

- On Windows, the **opennet.std** file, which must be on the server.
- On UNIX, the environment (the **startnet** file) before starting **vtxnetd**.
- On OpenVMS, a system-wide logical before starting the server program (**CONNECT_STARTUP.COM**).

Used by

*xf*ODBC, SQL Connection.

See also

- "Error Logging" in the "Data Access Errors and Error Logging" chapter of the *xfODBC User's Guide*.
- "Troubleshooting and Error Logging" in the "Error Logging and Messages" chapter of the *SQL Connection Reference Manual*.
- The "Configuring Connectivity Series" chapter of the *Installation Configuration Guide* for information on **vtxnetd** and **vtxnet2**.

Examples

On Windows,

```
VORTEX_HOST_SYSLOG=1
```

VORTEX_ODBC_CHAR – ODBC string descriptions

The VORTEX_ODBC_CHAR environment variable determines how strings are passed and described.

Value

One of the following values:

1	Pass strings as SQL_VARCHAR but describe them as SQL_CHAR.
12	Pass and describe strings as SQL_VARCHAR. (default)

Discussion

Some applications, such as Microsoft Data Transformation Services (DTS), require strings to be passed as they are described, which is the ODBC standard. By default, however, the *xf*ODBC driver passes strings as SQL_VARCHAR (that is, with trailing spaces removed), but describes them as SQL_CHAR.

VORTEX_ODBC_CHAR is used by the *xf*ODBC driver when it sends data to the application. It does not affect the way system catalogs are generated.

Setting location

The system environment. For client/server configurations, VORTEX_ODBC_TIME must be set on the client. (For a service, such as IIS or SQL Server, you must reboot after setting VORTEX_ODBC_TIME.)

Used by

*xf*ODBC.

See also

"Passing strings as SQL_VARCHAR" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*.

Examples

```
VORTEX_ODBC_CHAR=12
```

VORTEX_ODBC_DATETIME – Control how datetime columns are retrieved

The VORTEX_ODBC_DATETIME environment variable determines how datetime columns are retrieved.

Value

The integer representation for an ODBC data type.

Discussion

By default, datetime columns are retrieved as SQL_TIMESTAMP (an ODBC data type with an integer value of 11). You can, however, instruct *xf*ODBC to retrieve datetime columns as a different ODBC data type by setting this environment variable to the integer value for the data type. For example, to retrieve SQL_DATETIME columns as SQL_CHAR, set VORTEX_ODBC_DATETIME to 1.

Setting location

The system environment. For client/server configurations, VORTEX_ODBC_DATETIME must be set on the client.

Used by

*xf*ODBC.

See also

"Specifying formats for dates and times" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*.

Examples

The following instructs *xf*ODBC to retrieve timestamp columns as SQL_CHAR values:

```
VORTEX_ODBC_DATETIME=1
```

VORTEX_ODBC_TIME – Describe time columns as SQL_TIMESTAMP or SQL_TIME

WIN

The VORTEX_ODBC_TIME environment variable determines how time columns are described.

Value

One of the following values:

10	Describe time columns as SQL_TIME. (default)
11	Describe time columns as SQL_TIMESTAMP.

Discussion

ADO.NET retrieves SQL_TIME columns (the default data type for time columns in *xf*ODBC) as System.TimeSpan, which is a .NET data type that represents a time interval, rather than a specific time. So unless an application is written to use a time interval, the time must generally be calculated from timespan values. VORTEX_ODBC_TIME, however, can make these calculations unnecessary by instructing the *xf*ODBC driver to describe time columns as SQL_TIMESTAMP. The VORTEX_ODBC_TIME environment variable can be set to one of the following:

10	Describe time columns as SQL_TIME.
11	Describe time columns as SQL_TIMESTAMP.

Note the following:

- If VORTEX_ODBC_TIME is not set (which is the default when the Synergy/DE Data Provider for .NET is not in use), the *xf*ODBC driver describes time columns as SQL_TIME.
- When you use the Synergy/DE Data Provider for .NET, VORTEX_ODBC_TIME is automatically set to 11, and you cannot override this setting.
- SQL_TIMESTAMP values have both a date and a time, so to create a SQL_TIMESTAMP value, *xf*ODBC includes the date 1-1-1.

Setting location

The system environment. For client/server configurations, VORTEX_ODBC_TIME must be set on the client. Note that for a service, such as IIS or SQL Server, you must reboot after setting VORTEX_ODBC_TIME, unless you set it in a DSN (see "Adding a user or system DSN" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*).

Used by

*xf*ODBC.

See also

"Time columns and ADO.NET" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*.

Examples

```
VORTEX_ODBC_TIME=11
```

VORTEX_SHM_BASE – Base address for system catalog caching

The VORTEX_SHM_BASE environment variable sets the base shared memory address for system catalog caching.

Value

A hexadecimal value for the shared memory subsystem address that will be used as the base address for system catalog caching. This value is either 8 or 16 digits and must be in the correct format for the underlying architecture. (For example, on x86 systems, use 00000008 for address 80000000.) If this environment variable is not set (which is the default), the operating system determines this address.

Discussion

On UNIX systems you may need to set the base address for system catalog caching. See "Adjusting the shared memory subsystem settings" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*. Synergy/DE Developer Support may ask you to set this on other operating systems as well.

Setting location

The environment or

- the **opennet.srv** file on Windows.
- the **startnet** script on UNIX.
- the **STARTNET.COM** file on OpenVMS.

Note that when VORTEX_SHM_BASE is set in both the environment and one of the above files (or script), the settings must be identical.

Used by

*xf*ODBC.

VORTEX_SHM_FILE – .dat file for system catalog caching

The VORTEX_SHM_FILE environment variable specifies the location and name of the .dat file that *xf*ODBC uses for system catalog caching.

Value

The path and name of the **synodbccache.dat** file.

Discussion

To run **syngenload** from the command line on Windows or OpenVMS, you must set VORTEX_SHM_FILE. (On UNIX, this is set for you. See “Running syngenload from the command line (Windows and OpenVMS)” in the “Configuring Data Access” chapter of the *xfODBC User’s Guide*. Synergy/DE Developer Support may ask you to set this in other cases as well.

Setting location

The environment or

- the **opennet.srv** file on Windows.
- the **startnet** script on UNIX.
- the **STARTNET.COM** file on OpenVMS.

Note that when VORTEX_SHM_FILE is set in both the environment and one of the above files (or script), the settings must be identical.

Used by

*xf*ODBC.

VTXIPC_SO – VTXIPC_SO.EXE shared image

VMS

The VTXIPC_SO logical specifies the full path and filename of an important shared image that is distributed with *xf*ODBC.

Value

The full path and filename of **VTXIPC_SO.EXE**, a shared image distributed with *xf*ODBC. The filename is always **VTXIPC_SO.EXE**.

Discussion

The *xf*ODBC installation sets the VTXIPC_SO logical in **CONNECT_STARTUP.COM**, a file that's read when the system is started. VTXIPC_SO is required; it must be set.

Setting location

The **CONNECT_STARTUP.COM** file.

Used by

*xf*ODBC.

Examples

```
$ DEFINE/SYS VTXIPC_SO DKA300:[SYNERGYDE.CONNECT]VTXIPC_SO.EXE
```

WBNOINC – Suppress .INCLUDE processing while Workbench is tagging

WIN

The WBNOINC environment variable suppresses the processing of include files while Workbench is tagging.

Value

Any value.

Discussion

You may want to use WBNOINC if your machine has a slow processor or insufficient memory to cache your include files, since processing include files can involve a lot of overhead and may slow Workbench significantly while you are typing.

Setting location

The [synergy] section of the **synergy.ini** file or the global environment. (See "Settings on Windows" on page 1-10.)

Used by

Workbench.

WND – UI Toolkit directory

The WND environment variable specifies the directory that contains your UI Toolkit distribution. It is required for normal operation of UI Toolkit tools and programs.

Value

The path, including the device, for the directory that contains the UI Toolkit distribution files.

Setting location

- On Windows, the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).
- On UNIX, the **setsde** file.
- On OpenVMS, the **SYNERGY_STARTUP.COM** file (or **ACTIVATE_SDE.COM** for alternate installations).

Used by

UI Toolkit.

Examples

On UNIX,

```
WND=/usr/synergy/toolkit     ;export WND
```

WNDC – Default colors

UNIX, VMS

The WNDC environment variable enables color and defines the default color palette for the windowing API and UI Toolkit applications.

Value

palette1, palette2, …, palette16

Each color palette entry (*palette1*, *palette2*, and so forth) defines the foreground (character) color and the background color in the form *foreground/background*, where *foreground* and *background* are single characters that each signify a color. These characters, which are not case sensitive, are as follows:

D	Dark (black)
B	Blue
G	Green
C	Cyan
R	Red
M	Magenta
Y	Yellow
W	White

For example, D/W, W/D, B/W, B/Y, R/W.

Discussion

Color palette entries are separated by commas, and you can skip a color palette entry by placing consecutive commas in the string. If you omit a palette entry at the end of the string, you don't need to mark its place with a comma. Any omitted palette entries default to the colors for palette entry one, and if palette entry one is not specified, it defaults to white characters on a black background (W/D). The background color for palette entry number one is the default background color for the entire screen.

If WNDC is set, Synergy Language attempts to do color processing. If you're not using a color terminal, don't set WNDC.

UNIX

You must also add the following codes to the **termcap** file entry for the terminal you intend to use:

bB=*color definition sequence for blue background*
bC=*color definition sequence for cyan background*
bD=*color definition sequence for black background*
bG=*color definition sequence for green background*
bM=*color definition sequence for magenta background*
bR=*color definition sequence for red background*
bW=*color definition sequence for white background*
bY=*color definition sequence for yellow background*
fB=*color definition sequence for blue foreground*
fC=*color definition sequence for cyan foreground*
fD=*color definition sequence for black foreground*
fG=*color definition sequence for green foreground*
fM=*color definition sequence for magenta foreground*
fR=*color definition sequence for red foreground*
fW=*color definition sequence for white foreground*
fY=*color definition sequence for yellow foreground*

where **b** indicates background, **f** indicates foreground, and **B**, **C**, **D**, **G**, **M**, **R**, **W**, and **Y** indicate the colors listed above. To find out the escape sequence for each color, refer to the manual for the terminal you're using.

VMS

The WNDC logical must exist for color to be enabled.

The screen is updated using ANSI color escape sequences.

The WNDC environment variable is not used on Windows. Instead, a default color palette and a default set of Synergy color definitions are loaded into memory when the Synergy runtime starts. See "Colors and the color palette" in the "Synergy Windowing API" chapter of the *Synergy Language Reference Manual* for more information.

Setting location

The environment.

Used by

Runtime.

Examples

UNIX

The example below defines palette entry one as green characters on a blue background, entry two as red characters on a yellow background, entry four as black characters on a white background, and all other palette entries as green characters on a blue background (like palette entry one).

```
WNDC=g/b,r/y,,d/w  ;export WNDC
```

The example below defines palette entry three as white characters on blue, palette entry five as black characters on red, and all other palette entries as white characters on black.

```
WNDC=,,w/b,,d/r   ;export WNDC
```

The example below enables color but initializes all palette entries to white on black.

```
WNDC=,  ;export WNDC
```

VMS

The example below defines palette entry one as green characters on a blue background, entry two as red characters on a yellow background, entry four as black characters on a white background, and all other palette entries as green characters on a blue background (like palette entry one).

```
$ DEFINE WNDC "G/B,R/Y,,D/W"
```

The example below defines palette entry three as white characters on blue, palette entry five as black characters on red, and all other palette entries as white characters on black.

```
$ DEFINE WNDC ",,W/B,,D/R"
```

The example below enables color but initializes all palette entries to white on black.

```
$ DEFINE WNDC ","
```

XF_REMOTE_HOST – Host name for *xf*ServerPlus

The XF_REMOTE_HOST environment variable defines the host name of the machine on which *xf*ServerPlus is listening for remote calls from *xf*NetLink Synergy.

Value

The name of the machine where *xf*ServerPlus is running.

Discussion

Specifying the host name enables *xf*NetLink to read it as a default.

Setting location

On Windows, in the **synergy.ini** file. On UNIX and OpenVMS, in the environment.

Used by

*xf*NetLink Synergy Edition.

See also

"Specifying the Host Name and Port Number" in the "Configuring and Testing *xf*NetLink Synergy" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

```
XF_REMOTE_HOST = elmo
```

XF_REMOTE_PORT – Port number for *xf*ServerPlus

The XF_REMOTE_PORT environment variable defines the port number of the machine on which *xf*ServerPlus is listening for remote calls from *xf*NetLink Synergy.

Value

The port number of the machine where *xf*ServerPlus is running.

Discussion

Specifying the port number enables *xf*NetLink to read it as a default.

Setting location

On Windows, in the **synergy.ini** file. On UNIX and OpenVMS, in the environment.

Used by

*xf*NetLink Synergy Edition.

See also

"Specifying the Host Name and Port Number" in the "Configuring and Testing *xf*NetLink Synergy" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

```
XF_REMOTE_PORT = 2367
```

XF_RMT_DBG_TIMOUT – Connect session time-out for *xf*NetLink Synergy (debug)

The XF_RMT_DBG_TIMOUT environment variable controls the length of time that *xf*NetLink Synergy waits for an acknowledgment from the session started by the *xf*ServerPlus logic server when running in debug mode.

Value

The desired time-out value, in seconds.

Discussion

The connect session time-out is set separately for normal and debug operation. The default debug connect session time-out is 10 minutes.

Setting location

On Windows, in the **synergy.ini** file. On UNIX and OpenVMS, in the environment.

Used by

*xf*NetLink Synergy Edition.

See also

"Specifying Time-out Values" in the "Configuring and Testing *xf*NetLink Synergy" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

To set a six-minute time-out,

```
XF_RMT_DBG_TIMOUT = 360
```

XF_RMT_TIMOUT – Call time-out for *xf*NetLink Synergy

The XF_RMT_TIMOUT environment variable controls the length of time that *xf*NetLink Synergy Edition waits for the results of a remote routine call to *xf*ServerPlus.

Value

The desired time-out value, in seconds.

Discussion

The call time-out is measured for each send–receive request between *xf*NetLink and *xf*ServerPlus. The default is 30 minutes (1800 seconds).

Setting location

On Windows, in the **synergy.ini** file. On UNIX and OpenVMS, in the environment.

Used by

*xf*NetLink Synergy Edition.

See also

"Specifying Time-out Values" in the "Configuring and Testing *xf*NetLink Synergy" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

To set a 20-minute time-out,

```
XF_RMT_TIMOUT = 1200
```

XF_RMTCONN_TIMOUT – Connect session time-out for *xf*NetLink Synergy (normal)

The XF_RMTCONN_TIMOUT environment variable controls the length of time that *xf*NetLink Synergy Edition waits for an acknowledgment from the session started by the *xf*ServerPlus logic server when running in normal mode.

Value

The desired time-out value, in seconds.

Discussion

The connect session time-out is set separately for normal and debug operation. For normal operation, the default connect session time-out is two minutes.

Setting location

On Windows, in the **synergy.ini** file. On UNIX and OpenVMS, in the environment.

Used by

*xf*NetLink Synergy Edition.

See also

"Specifying Time-out Values" in the "Configuring and Testing *xf*NetLink Synergy" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

To set a three-minute time-out,

```
XF_RMTCONN_TIMOUT = 180
```

XFNLS_LOGFILE – Filename for debug trace information

WIN, UNIX

The XFNLS_LOGFILE environment variable specifies the filename for debug trace information on *xf*NetLink Synergy.

Value

The filename. Include the full path if the file is not in the working directory.

Discussion

When you run an *xf*ServerPlus session in debug mode from *xf*NetLink Synergy, you can use XFNLS_LOGFILE to specify that the packets be written to a file instead of to the screen. The file is created if it does not exist. If it already exists, additional material is appended to the end.

Setting location

On Windows, in the **synergy.ini** file. On UNIX, in the environment.

Used by

*xf*NetLink Synergy Edition.

See also

"Specifying Debug Options" in the "Configuring and Testing *xf*NetLink Synergy" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

```
XFNLS_LOGFILE = Myfile.txt
```

XFODBCUSR_SO – Shared image for routines for user-defined data

VMS

The XFODBCUSR_SO logical specifies the full path and filename for the shared image that contains routines for user-defined data.

Value

The full path and filename of the shared image for user-defined data routines. The filename is always **XFODBCUSR_SO.EXE**.

Discussion

The *xf*ODBC installation sets the XFODBCUSR_SO logical in **CONNECT_STARTUP.COM**, a file that's read when the system is started.

XFODBCUSR_SO must be set. It specifies the location of the **XFODBCUSR_SO.EXE** file, which contains routines to handle user-defined data. The Connectivity Series distribution includes a default **XFODBCUSR_SO.EXE** file, but you can create your own routines and overwrite this file.

Setting location

The **CONNECT_STARTUP.COM** file.

Used by

*xf*ODBC.

See also

Appendix A of the *xfODBC User's Guide*

Examples

```
$ DEFINE/SYS XFODBCUSR_SO DKA300:[SYNERGYDE.CONNECT]XFODBCUSR_SO.EXE
```

XFPL_API – XFPL_API.EXE shared image

VMS

The XFPL_API logical specifies the location of the shared image that contains the *xf*ServerPlus API routines (SET_XFPL_TIMEOUT, XFPL_LOG, and XFPL_REGCLEANUP).

Value

The full path and filename of the shared image **XFPL_API.EXE**.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER.

Used by

*xf*ServerPlus.

Examples

```
$ DEFINE/SYS/EXEC XFPL_API SYNERGYDE$ROOT:[DBL]XFPL_API.EXE
```

XFPL_DBR – Run *xf*ServerPlus with dbr instead of dbs

WIN, UNIX

The XFPL_DBR environment variable specifies that *xf*ServerPlus should use **dbr** instead of **dbs**.

Value

Any value.

Discussion

By default, the *xf*ServerPlus program (**xfpl.dbr**) uses the service runtime, **dbs**, which is a reduced-size runtime intended for detached programs. However, there may be circumstances where your application requires that you run *xf*ServerPlus with the regular runtime, **dbr**. For example, your application might need to use the Synergy windowing API.

Although it is possible to run *xf*ServerPlus with **dbr**, we do not recommend doing so because it will adversely affect performance. When using **dbr**, *xf*ServerPlus sessions will start slower and require more memory than when using **dbs**.

If XFPL_DBR is set and session logging is turned on, "XFPL_DBR logical is set" will be written to the **xfpl.log** file.

Setting location

On Windows, in the registry. (You can use the Synergy Configuration Program to set *xf*ServerPlus environment variables on Windows.) On UNIX, in the **synrc** file. You can also set it in the environment on UNIX, but it must already be set when **rsynd** is started.

XFPL_DBR can be set so that it applies to a specific instance of *xf*ServerPlus or to all instances of *xf*ServerPlus. Set it the same as you would XFPL_SMCPATH or XFPL_INIPATH.

Used by

*xf*ServerPlus.

Examples

```
XFPL_DBR = 1
```

XFPL_DTL – XFPL_DTL.EXE shared image

VMS

The XFPL_DTL logical specifies the location of the shared image that contains the C DLL for *xf*ServerPlus.

Value

The full path and filename of the shared image **XFPL_DTL.EXE**.

Setting location

The **SYNERGY_STARTUP.COM** file found in SYS$MANAGER.

Used by

*xf*ServerPlus.

Examples

```
$ DEFINE/SYS/EXEC XFPL_DTL SYNERGYDE$ROOT:[DBL.BIN]XFPL_DTL.EXE
```

XFPL_INIPATH – xfpl.ini file location

The XFPL_INIPATH environment variable specifies the location of the **xfpl.ini** file when it is not in the default location.

Value

The full pathname for the **xfpl.ini** file.

Discussion

By default, *xf*ServerPlus reads the **xfpl.ini** file from the DBLDIR directory. Use the XFPL_INIPATH environment variable when **xfpl.ini** is not in DBLDIR.

Setting location

On Windows, in the registry. (You can use the Synergy Configuration Program to set *xf*ServerPlus environment variables on Windows.) On UNIX, in the environment or in the **synrc** file. On OpenVMS, in **SYNRC.COM**.

XFPL_INIPATH can be set so that it applies to a specific instance of *xf*ServerPlus or to all instances of *xf*ServerPlus. See “Setting the XFPL_INIPATH Environment Variable” in the “Configuring and Running *xf*ServerPlus” chapter of the *Developing Distributed Synergy Applications* manual for details.

Used by

*xf*ServerPlus.

See also

“Setting the XFPL_INIPATH Environment Variable” in the “Configuring and Running *xf*ServerPlus” chapter of the *Developing Distributed Synergy Applications* manual.

Examples

In the environment on UNIX,

```
XFPL_INIPATH = /usr/temp     ;export XFPL_INIPATH
```

XFPL_SMCPATH – Synergy Method Catalog files location

The XFPL_SMCPATH environment variable specifies the location of the Synergy Method Catalog files when they are not in the default location.

Value

The pathname of the SMC files.

Discussion

By default, the Method Definition Utility (MDU) and *xf*ServerPlus read the SMC data files from the DBLDIR directory. Use the XFPL_SMCPATH environment variable when these files are not in DBLDIR.

Setting location

On Windows, in the registry for use by *xf*ServerPlus and in the environment for use by the MDU. (You can use the Synergy Configuration Program to set *xf*ServerPlus environment variables on Windows.) On UNIX, in **synrc** or the environment for use by *xf*ServerPlus and in the environment for use by the MDU. On OpenVMS, in the **SERVER_INIT.COM** file for use by *xf*ServerPlus and in the environment for use by the MDU.

XFPL_SMCPATH can be set so that it applies to a specific instance of *xf*ServerPlus or to all instances of *xf*ServerPlus. See "Setting the XFPL_SMCPATH Environment Variable for xfServerPlus" in the "Defining Your Synergy Methods" chapter of the *Developing Distributed Synergy Applications* manual for details.

Used by

Method Definition Utility, *xf*ServerPlus.

See also

- "Specifying Which SMC to Update" in the "Defining Your Synergy Methods" chapter of the *Developing Distributed Synergy Applications* manual.
- "Setting the XFPL_SMCPATH Environment Variable for xfServerPlus" in the "Defining Your Synergy Methods" chapter of the *Developing Distributed Synergy Applications* manual.

Examples

As a system-level environment variable on Windows,

```
XFPL_SMCPATH = c:\temp
```

XSHOW – Keep application iconized

WIN

The XSHOW environment variable specifies that an application should remain as an icon until U_START is called.

Value

Any value.

Discussion

If XSHOW is not set, window size adjustments can be seen as the footer, information line, and toolbars are added when the applications starts up. To postpone the display of the application window until this sizing is complete, set XSHOW to any value.

A special value, HIDE, keeps the application window hidden. Setting XSHOW to HIDE has the same effect as setting APP_STATE to HIDDEN.

The XSHOW environment variable affects the current program and any programs spawned by the current program.

Setting location

The environment or the [synergy], [dbr], or [*myprog*] section of **synergy.ini** (where *myprog* is any **.dbr** file).

Used by

Runtime.

Examples

In the **synergy.ini** file,

```
[synergy]
XSHOW=1
```

2

System Options

System options #9, #14, #20, #25, and #26, which were available in a previous version of Synergy Language, are obsolete and ignored.

Setting an Option

By default, an option is not set. Specifying an option as an argument to the DBLOPT environment variable sets that option. If Synergy Language doesn't recognize an option on your operating system, that option is ignored.

WIN

Do *not* use quotation marks when defining multiple runtime options. For example:

```
set DBLOPT=1, 7, 16, 22
```

VMS

When you define multiple system options, make sure you enclose the options in quotation marks. For example:

```
define DBLOPT "1, 7, 16, 35"
```

If you don't use quotation marks, DCL interprets the logical as a search list, and the runtime will only process the first option specified.

See page 1-66 for more information about DBLOPT.

You can also set or reset a system option at runtime using the %OPTION function. See %OPTION in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual* for more information.

The following pages explain in detail all system options that are available on the various operating systems.

The online release notes file, **REL_DBL.TXT**, may contain additional information about system options.

#1 – Default SEND queue

System option #1 determines the default terminal number for the SEND statement.

If you set option #1, the default terminal number is 255, and the message being sent is inserted into the global message queue. System option #13 also sets the default terminal number, and it overrides option #1. If option #13 is set, option #1 is ignored.

If you don’t set option #1 or #13 and you don’t specify a terminal number in a SEND statement, the default terminal number is the number of the terminal running the program. The message is inserted into the terminal’s local message queue.

For more information about sending and receiving messages, see “Synergy Language messages and message queues” and “Receiving messages” in the “Welcome to Synergy Language” chapter of the *Synergy Language Reference Manual*.

#2 – Default file specification on STOP

System option #2 controls the default file specification for the STOP statement.

If you set option #2, the filename extension and path default to the extension and path of the program requesting the chaining. For example, on UNIX, if you run a program called **current** with the following command line:

```
dbr /usr/mine/current.app
```

the statement

```
STOP next
```

causes the program to chain to **/usr/mine/next.app**.

If you don't set this option, the filename extension defaults to **.dbr** on Windows and UNIX and to **.exe** on OpenVMS, and the path defaults to the current directory. For example, if you chain to a program called **next** from the program **/usr/mine/current.app** on UNIX, the Synergy runtime attempts to transfer execution control to the program **next.dbr** in the current directory.

#3 – ISAM file I/O caching

UNIX

System option #3 enables ISAM file caching on all OPENs.

When system option #3 is set, or when you use the **/cache** qualifier in the OPEN OPTIONS string (see OPTIONS in the "Synergy Language Statements" chapter of the *Synergy Language Reference Manual*), the runtime performs three types of caching to ISAM files depending on how the file was opened:

- Files opened with SHARE:Q_EXCL_RW get full caching. ISAM I/O is cached and is not written to disk until a CLOSE or FLUSH is issued. Blocks remain in read cache until that time.
- Files opened with SHARE:Q_EXCL_RO get a write-through form of caching. ISAM I/O is cached, but all output is written simultaneously to disk. Blocks remain in read cache until a CLOSE or FLUSH is issued.
- Files open without exclusive access get a write-through form of caching. ISAM I/O is cached, but all output is written simultaneously to disk. Blocks remain in read cache until either the file is updated by another user or a CLOSE or FLUSH is issued.

In all cases, if the cache becomes full before a CLOSE or FLUSH is issued, the oldest blocks will be flushed. (The NUMBUFS environment variable determines the size of the cache; see NUMBUFS on page 1-131 for more information.)

When you use ISAM file caching, an additional amount of memory and system resource is required for each file OPEN.

System option #3 only affects Revision 4 or higher ISAM files. If you want to enable caching on ISAM files created before Synergy Language 6.1, run **isutl** with the **-p** option to update your files.

TIP To determine if an ISAM file was created with version 6.1 or higher, run the **ipar** utility. If the generated parameter file descriptions include the text "Shared index cache allowed," your ISAM file was created with 6.1 or higher or was updated using **ismvfy -p** or **isutl -p**, and system option #3 enables file caching on all OPENs. See ipar in the "Synergy DBMS" chapter of *Synergy Language Tools* for more information.

WIN

ISAM file I/O caching is always enabled.

#5 – CRT mode

System option #5 determines whether terminal I/O should be interpreted as CRT I/O.

If you set option #5, Synergy Language assumes that terminal I/O is I/O to or from a CRT. This option acts the same as the FLAGS subroutine runtime option flag 4.

If you don't set this option, CRT mode is reset upon program start-up. Any terminal I/O is assumed to be non-CRT I/O.

Option #5 is automatically set if option #35 is set.

VMS

Synergy Language interprets the terminal setting to set the default for this option.

#7 – Message manager

UNIX, VMS

System option #7 determines whether the runtime uses the Synergy message manager or the local message facilities.

If you set option #7, Synergy Language uses the Synergy message manager (which is the Synergy Language daemon on UNIX) if it is available. If the message manager is not available, this option is ignored.

VMS

Option #7 is on by default.

If you don't set this option, Synergy Language uses the local message facilities for the SEND and RECV statements, which means that interprocess SENDs and RECVs are not possible.

#10 – Interrupt character(s)

System option #10 controls whether or not the interrupt character(s) are ignored at program start-up and between chains.

If option #10 is set, the interrupt character(s) are ignored at program start-up and between chained programs, and the state of the FLAGS subroutine runtime option flag 8 is automatically set on entry to each program. You can control when an interrupt character is processed (in other words, not ignored) throughout your program by unsetting and resetting flag 8.

If you don't set this option, Synergy Language processes any interrupt character that the user types unless flag 8 is set.

#11 – Rounding vs. truncation

System option #11 determines whether Synergy Language rounds or truncates expression results by default.

If option #11 is set, Synergy Language truncates results in the following situations at compile time:

- Stores from an implied-decimal type or an alpha type in implied-decimal format (for example, "1.3") to either an integer, a decimal, or an implied-decimal with a smaller fractional precision
- Intrinsic functions and system-supplied subroutines when an implied-decimal value is passed as an argument to a routine expecting a decimal or integer value
- The following data references when an implied-decimal value is used in place of a decimal or integer value (where *dexp* is specified):

  ```
  dim_var[dexp, dexp, ...] ^arg(dexp)
  var(dexp)          ^argn(dexp)
  var(dexp, dexp) ^d(exp, dexp)
  var(dexp:dexp)
  ```

- The GOTO statement when an implied-decimal value is used in place of a decimal or integer value (where *dexp* is specified):

  ```
  goto (lbl, lbl, ...) dexp
  ```

- I/O statements when an implied-decimal value is used as a channel argument or qualifier value
- Multiplication or division operations, if the result is an implied-decimal value with more than 10 digits of fractional precision
- READ and WRITE statements when you specify the record number as the third argument (where *dexp* is specified):

  ```
  read(channel, data_area, dexp)
  ```

- %IMPLIED with an alpha argument containing an implied-decimal value whose fractional precision is greater than 10

If option #11 is not set, Synergy Language defaults to rounding in each of the above situations. When rounding, if the leftmost digit (or the eleventh digit for multiplication or division) of the fractional precision is in the range 5 through 9, Synergy Language adds one to the absolute value of the whole number part (or the fractional part for multiplication or division). For example, if system option #11 is not set and you specify 3.5 as the record number to read in a READ statement, you get the fourth record. If system option #11 is set, you get the third record.

The state of this option is overridden in routines using the ROUND or TRUNCATE option on MAIN, SUBROUTINE, and FUNCTION.

This option is only used at compile time. It is ignored at runtime.

#12 – Buffered terminal output

UNIX

System option #12 affects buffered terminal output.

Option #12 determines whether or not you'll be able to customize the size of the terminal buffer. If you set it, you can customize the size of the terminal buffer for use with the TBUF environment variable. (For more information about TBUF, see TBUF on page 1-239.)

This option does not affect the Synergy windowing API.

#13 – Default SEND queue

System option #13 determines the default terminal number for the SEND statement.

If you set option #13, the default terminal number is 254, and the message being sent is inserted into the group message queue. Option #1, which also sets the default terminal number, is ignored.

If you don’t set this option or option #1 and you don’t specify a terminal number in a SEND statement, the default terminal number is the number of the terminal running the program. The message is inserted into the terminal’s local message queue.

For more information about message queues, see “Synergy Language Messages and Message Queues” in the “Welcome to Synergy Language” chapter of the *Synergy Language Reference Manual*.

#15 – File preallocation

UNIX

This feature is provided for compatibility with older software and should be considered obsolete.

System option #15 determines whether a file's initial allocation size as specified in the OPEN statement is used or ignored.

If you set option #15, Synergy Language uses the specified initial allocation size when it opens the file. When the file is closed, the size is *not* cut back to the written size of the file.

If you don't set this option, any initial allocation that you specify in the OPEN statement is ignored. For example:

```
open(15, o "test.ddf[500]")
```

is interpreted as

```
open(15, o, "test.ddf")
```

#16 – Quit character

UNIX, VMS

System option #16 maps the quit character to the interrupt character.

If you set option #16, pressing the quit character has the same effect as pressing the interrupt character. For example, on OpenVMS, CTRL+Y performs the same function as CTRL+C.

If you don't set this option, the quit character quits out of the program.

The quit characters and interrupt characters supported by Synergy Language are listed in Appendix C, in the *Synergy Language Reference Manual*.

Option #16 is automatically set if option #35 is set.

UNIX

If system option #16 is set, when possible, we trap the suspend signal (TSTP) and perform the same action as when we trap interrupt.

VMS

You can restart the program at the point of interruption by issuing a DCL CONTINUE command.

#17 – Terminal number returned by TNMBR

System option #17 controls how the TNMBR routine determines the terminal number.

VMS

If you set option #17, TNMBR uses the physical device to determine the terminal number for a virtual terminal.

If you don't set this option, TNMBR returns a number based on the VT device specification for a virtual terminal.

For more information about the TNMBR routine, see TNMBR in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual*.

#18 – In-place MERGE and logical end-of-file

UNIX

System option #18 controls how the in-place MERGE statement handles the logical end-of-file in the primary file. (An in-place MERGE merges two or more sorted files without creating a new output file; the secondary file is merged into the first input file. For more information, see MERGE in the “Synergy Language Statements” chapter of the *Synergy Language Reference Manual*.)

If you set option #18, the in-place MERGE extends the primary file by the number of records merged from the secondary file. All records past the logical end-of-file in the primary file are copied to the end of the primary file’s extension.

If you don’t set option #18, the in-place MERGE treats any logical end-of-file character or record as the end of usable data in the primary file. Any records that follow the logical end-of-file are overwritten.

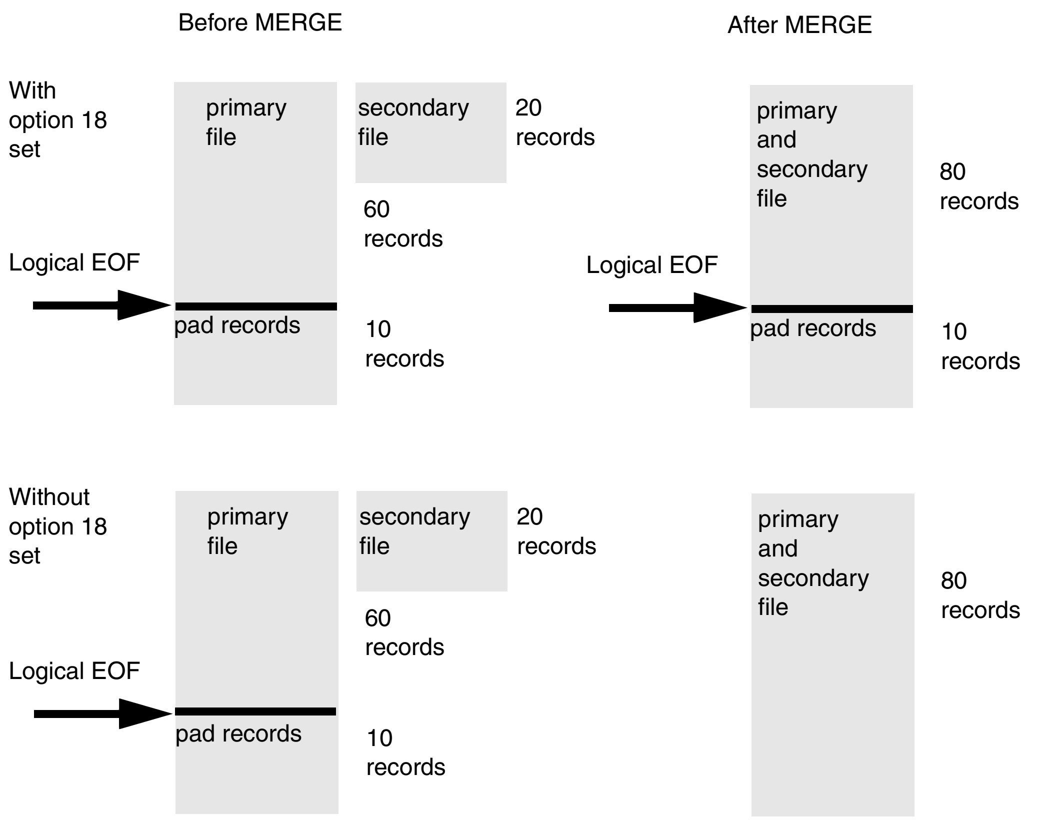

Figure 2-1. System Option #18 – In-place MERGE and logical end-of-file.

#21 – Interrupt trapping

System option #21 determines whether or not the interrupt character is trappable.

If you set option #21, the ONERROR statement cannot trap the interrupt character, which means that a user can press the interrupt character to interrupt your program.

If you don't set this option, the ONERROR statement traps the interrupt character, which may make it difficult to abort your program.

#22 – LPQUE statement

System option #22 determines how the LPQUE statement interfaces with the operating system.

WIN

If you set option #22, the runtime sends LPQUE arguments to the file **DBLDIR:dblpq.bat**, which can contain customized print commands. If you do not set this option, the runtime sends the print job directly to the Microsoft Windows print manager. Option #22 can be set in the environment or the **synergy.ini** file. See the *Migrating Your Application to Windows* document, available on the Online Manuals CD, for more information.

If you want more functionality than the default LPQUE statement provides, we recommend that you not set option #22 and that you use the Synergy Windows printing API instead. (See the "Synergy Windows Printing API" chapter in the *Synergy Language Reference Manual* for more information.)

UNIX

If you set option #22, the runtime sends LPQUE arguments to the script file **DBLDIR:dblpq**, which you can change to your own printing specifications. The LPQUE statement then executes the arguments in **dblpq** instead of those in the default printing program.

If you don't set this option, the LPQUE statement arguments executes the default printing program. Default UNIX printing programs are as follows:

System V	**lp**
BSD	**lpr**

We recommend that you examine the **dblpq** file to make sure it's set up appropriately when you install Synergy Language.

VMS

If you set option #22 and the program is running from an interactive session, the LPQUE statement spawns a PRINT statement to print a specified file, which enables you to add PRINT options after the filename.

If you don't set this option, or if you do set it and the program is *not* running from an interactive session, the LPQUE statement uses the **$sndjbc** system service to print a specified file. Any switches that follow the filename are ignored. This feature works within detached or batch processes. You cannot use wildcard characters in the file specification.

#23 – In-place MERGE and duplicate records

WIN, UNIX

System option #23 determines where the in-place MERGE statement places duplicate records.

If you set option #23, the in-place MERGE places duplicate records from the second input file *after* their matching records from the first input file.

If you don’t set this option, the in-place MERGE statement places duplicate records from the second file *before* their matching records from the first file.

For more information about the MERGE statement, see MERGE in the “Synergy Language Statements” chapter of the *Synergy Language Reference Manual*.

#27 – Suppression of record truncation I/O error

System option #27 allows storage of a larger record into a smaller destination buffer on a READ or READS and allows a write of a larger buffer to a smaller record on a STORE, WRITE, and WRITES.

If system option #27 is set, both the $ERR_IRCSIZ and $ERR_TOOBIG errors are disabled for ISAM files, and the $ERR_TOOBIG error is disabled for non-ISAM files. (Instead of generating an error, truncation occurs.) Disabling these errors enables programs that take advantage of incorrectly sized buffers to continue to function correctly.

If this option is not set, errors $ERR_IRCSIZ and $ERR_TOOBIG are enabled, and a READ or READS isn't allowed to store a larger buffer into a smaller destination, nor is a STORE, WRITE, or WRITES allowed to pass a larger buffer to a smaller record.

⚠ Use system option #27 with caution. An application that allows a record larger than the file's defined record size to be written may eventually cause a significant field of the record to be truncated without warning as new fields are added to the record.

#28 – Mapping decimal data type to numeric

System option #28 causes the compiler to map decimal subroutine arguments (**d**) to numeric (**n**).

Option #28 is a compiler option only. It has no effect at runtime. The compiler checks for this option when it compiles your modules. Once your modules are built, you cannot map the decimal data type to numeric.

If system option #28 is set, the compiler treats any subroutine arguments declared as type **d** as type **n**.

If this option is not set, no mapping of the decimal data type occurs.

#29 – Dimensioned variable

System option #29 determines whether the compiler requires dimension specifications on a dimensioned variable.

If you set option #29 and a dimensioned variable is referenced without dimension specifications (“[ ]”), Synergy Language generates a “Dimension specifications required for {var}” compiler error. Dimension specifications are *not* required on subroutine arguments and the argument to the ^PASSED data reference operation.

If you don’t set option #29, Synergy Language won’t require dimension specifications on dimensioned variables.

#30 – Default terminal definition

UNIX, VMS

System option #30 controls whether the runtime disregards the TERM environment variable in determining what type of terminal you are using.

If you set option #30, the runtime is forced to use the default, built-in terminal definition regardless of how the TERM environment variable is set. The default terminal definition includes all Compaq VT*xxx* series capabilities.

If you don't set this option, the runtime looks at TERM to determine which type of terminal you are using.

VT100 terminals may have a problem with the EC (erase character) **termcap** code in the Synergy windowing API. We suggest that you use VT102 or greater terminals if you want to use this option.

If you usually use TERM=vt100, setting system option #30 significantly reduces the character output to the terminal on screen clearing and attribute changes, which improves screen painting.

#31 – IF statement format

System option #31 determines which IF statement syntax the Synergy compiler recognizes.

If you set option #31, the compiler uses the alternate form of the IF statement during compilation, instead of the ANS DIBOL form. The alternate IF form is equivalent to using the alternate IF compiler option (for example, **-a** or **-qaltif** on UNIX).

Using the alternate IF syntax, an ELSE clause matches the previous IF clause. ANS DIBOL IF syntax, on the other hand, matches the ELSE clause with the previous THEN clause.

We'll use the following pseudocode as an example:

```
if (expr1)
  then if (expr2)
    stmt1
  else stmt2
```

With option #31 set, **stmt2** is executed only if **expr1** is true and **expr2** is false. If option #31 is not set, ANS DIBOL IF compilation is used, and **stmt2** is executed whenever **expr1** is false.

#32 – Uppercasing and lowercasing

System option #32 eliminates the uppercasing and lowercasing of all two-byte characters and characters whose ASCII code is greater than 127.

If you set system option #32, the uppercasing and lowercasing of the kanji language and all other two-byte characters is eliminated.

If system option #32 is not set, uppercasing and lowercasing occurs as normal.

#33 – SHARE qualifier disabling

UNIX

File locking on UNIX is implemented using the same system resource as record locking. For this reason, each file open requires at least one lock (for conventional file locking), in addition to any record locks applied by a READ operation. An application that opens a large number of files could exceed the maximum number of system-wide locks and generate the following error: “Internal runtime failure: Record locking problem.”

In the event this occurs, we recommend that you increase your system kernel parameter that controls the maximum number of locks allowed (typically FLCKREC or LOCKS, depending on your operating system). The parameter should be set to at least

number of running Synergy applications * *maximum open files per application* * 2

As a temporary solution, you can set system option #33, which disables this conventional file locking. As a result, however, all file SHARE access is also disabled.

Any file access that would normally have generated a “File in use by another user” error will now be allowed to continue and potentially cause data loss. This includes OPEN(O:mode), COPY, SORT, DELET, RENAM, ISCLR, and ISAMC.

On Sun Solaris systems, file locking is handled differently. Synergy uses a Sun-only locking facility that allows true (Windows- and VMS-style) file sharing. (This is always in effect and is not disabled by option #33.) In addition, however, the conventional file locking must also be performed to handle network file access (NFS) to non-Sun machines or when pre-8.1 Synergy runtimes are used to access the same files, where the Sun-only locking facility is not available. If neither of these situations can occur, we recommend using system option #33 to turn off the additional conventional file locking. This can have a dramatic impact on systems with a large number of concurrent Synergy users.

#34 – Command line syntax

WIN

Setting option #34 makes the runtime and utilities understand a non-UNIX-style command line.

By default, Synergy Language supports UNIX-style command line syntax on Windows with switches that follow a hyphen character (-). However, when system option #34 is set, Synergy Language recognizes command line switches that follow a slash character (/).

In addition, if this option is set, file specifications must contain backslash characters (\) between directories and subdirectories; you cannot optionally use a slash character. Also, you must use a hyphen (-) as the line continuation character instead of a backslash in command files specified on the command line.

The following table illustrates the effects of system option #34:

Option #34	Style	Command line switch	Continuation character
Not set	UNIX	–	\
Set	Windows	/	–

#35 – VAX DIBOL–compatible functionality

System option #35 provides VAX DIBOL–compatible functionality for certain Synergy Language features. If you set system option #35, the following changes to Synergy Language behavior occur:

- WAIT I/O statement qualifier: If you specify a value of 0 for the WAIT I/O statement qualifier, it indicates “wait forever” instead of “no wait.”
- Quit mapping: The quit character is mapped to the interrupt character.
- Appending to a nonexistent file: If you open a file that does not exist in append mode, a “File not found” error is generated.
- CRT mode: Synergy Language assumes that terminal I/O is I/O to or from a CRT.
- Invalid key checking: Invalid key checking is enabled (as if you had set system option #45). An error is generated if the key is not correctly specified.

WIN, UNIX

- BEGFL subroutine: The BEGFL subroutine resets the key of reference to zero for an ISAM file.
- PARSE subroutine: The PARSE subroutine stops at the semicolon (;) and doesn’t include it in the filename or file extension.

VMS

- TT: on the OPEN statement: If you open TT: while running from a command procedure, you only get input from the terminal, not the command procedure. Synergy Language no longer opens both SYS$INPUT: and SYS$OUTPUT:.
- **O:P** mode on the OPEN statement: If you open print files in **O:P** mode, Synergy Language uses sequential files with no carriage control, as opposed to sequential VFC. This implementation is more efficient if you’re outputting a large print file to a laser printer with escape sequences and graphics data, rather than standard carriage-return/line-feed records. (Each DISPLAY or WRITES statement for a VFC file incurs two bytes of overhead.)
- **O:S** mode on the OPEN statement: If you open a file in **O:S** mode with RECSIZ specified, Synergy Language creates a fixed-length file.
- Echoing control characters: Synergy Language echoes just the control symbol (^) by default, not control characters (for example, ^A and ^G).

- Line feeds on the FORMS statement: The FORMS statement on a channel opened to a file with carriage return carriage control outputs one extra line feed, advancing the paper one space more than the number specified in the FORMS statement. For example,

  ```
  open(chn, o, "file.ddf")
  forms(chn, 2)
  ```

  results in <CR><LF><LF> (three extra lines) if option #35 is set or <CR><LF> (two extra lines, as specified) if option #35 is not set.

- RENAM subroutine: The RENAM subroutine no longer works the same as the DCL command RENAME. Instead, it operates as in the following example:

  If you have three files named **FILE.A;1**, **FILE.A;2**, and **FILE.B;1**, and you use the RENAM subroutine as follows:

  ```
  xcall renam("FILE.A", "FILE.B")
  ```

  Synergy Language deletes **FILE.B;1** and renames **FILE.A;*** to **FILE.B;***. If flag 3 of the FLAGS subroutine is set, the existence of **FILE.B;1** causes a “Cannot supersede existing file” error ($ERR_REPLAC). (Normally, **FILE.A;2** is renamed to **FILE.B;2** by default.)

- RUNJB detached process: The process created by the RUNJB subroutine is detached. However, if the RUNJB *subprocess* argument is present, or if the *io_flag* argument has a nonzero value, option #35 is ignored.

- Message returned from ERTXT subroutine: The returned message includes the facility, severity, and mnemonic, and the *position* argument is loaded with the starting position of the message text. If option #35 is not set, the text does not include the facility, severity, or mnemonic, and *position* has a value of 1.

When you set option #35, options #5, #16, #38, #39, and #45 are also set. See #45 on page 2-39 for an important note.

If system option #35 is not set, Synergy Language handles the above functions as documented in the relevant sections of this manual.

#36 – File flushing

Synergy writes data directly to the operating system, but in many cases, the operating system caches data before writing it to the actual media. A flush operation is required to bypass this behavior. By default, the runtime does not flush WRITE, STORE, or DELETE operations on indexed, sequential, and relative files to disk. (The runtime does, however, flush ISAM files to disk after a CLOSE operation.) In addition, the runtime buffers WRITES to sequential files (open in output or append mode) in 8K buffers and only writes the cache buffer to the operating system when the buffer is full. This is done to improve sequential write performance, especially over mapped drives.

Setting system option #36 enables the flushing feature of the runtime for non-sequential files and uses the O_DSYNC file open flag. This ensures that all writes to the disk are flushed for data integrity; however, the file modification time is still cached. This is the best compromise between performance and guaranteed data integrity. Our current method of flushing ISAM files to disk flushes past any write caching that may be set up on the system. If system option #36 is set and you want to enable flushing for sequential files, add a FLUSH statement on the channel, immediately following the OPEN.

While this flushing procedure is more secure, flushing may decrease performance more in this version of Synergy Language than in past versions if write caching is enabled. However, if write caching is *disabled* on the drive, the new method of flushing affects performance less than before.

WIN, UNIX

On most Windows and UNIX systems, using system option #36 is faster than using the FLUSH statement. (On Windows it is significantly faster.)

VMS

If system option #36 is set, nonoutput files are not opened for deferred write or write behind caching.

Alternatively, you can use the FLUSH statement to flush files to disk programmatically. (See FLUSH in the “Synergy Language Statements” chapter of the *Synergy Language Reference Manual* for details.)

The **rpsutl** program disables system option #36. This improves performance for those who have system option #36 set in their environment.

#37 – VAX DIBOL–compatible store

System option #37 forces VAX DIBOL–compatible functionality on decimal-to-decimal and alpha-to-decimal stores. If option #37 is set, decimal-to-decimal assignments are translated. For example, assume you have the expression

```
dfld2=dfld1
```

where dfld1 contains “1 23”. With option #37 set, Synergy Language stores “1023” into **dfld2**. If the option is not set, no translation occurs.

On alpha-to-decimal stores, Synergy Language won’t ignore extra characters in the alpha source if option #37 is set. For example, the following statement causes a “Bad digit encountered” error ($ERR_DIGIT):

```
D5=A10 containing "XYZ--01234"
```

Alternatively, you can specify the /ALTSTORE compiler qualifier for each individual file for which you want to force an alternate store.

System option #37 is a compiler option. It has no effect at runtime. The compiler checks for this option when it compiles your modules. Once your modules are built, you cannot change their store functionality.

#38 – VAX DIBOL–compatible OPEN with O:P

System option #38 provides VAX DIBOL–compatible functionality for the OPEN statement with **O:P** mode.

If you open print files in **O:P** mode, Synergy Language uses sequential files with no carriage control, as opposed to the default sequential VFC format. This implementation is more efficient if you're outputting a large print file to a laser printer with escape sequences and graphics data, rather than standard carriage-return/line-feed records. (Each DISPLAY or WRITES statement for a VFC file incurs two bytes of overhead.)

Option #38 is automatically set when option #35 is set.

#39 – VAX DIBOL–compatible OPEN with TT: and echoing characters

System option #39 provides VAX DIBOL–compatible functionality for the following features:

- TT: on the OPEN statement
- Echoing control characters

If you set system option #35, the following changes to the Synergy runtime behavior occurs:

- If you open TT: while running from a command procedure, you only get input from the terminal, not the command procedure. The runtime no longer opens both SYS$INPUT: and SYS$OUTPUT:.
- The runtime echoes just the control symbol (^) by default, not control characters (for example, ^A and ^G) for characters entered by a READS statement. (Note that ^Z is an exception to this rule.)

Option #39 is automatically set when option #35 is set.

#40 – XCALL profiling

System option #40 enables XCALL profiling.

This option enables profiling for CPU time in the current routine only on all modules compiled with the profiling compiler option (**/profile** on OpenVMS and **-u** or **-qprofile** on all other systems).

See "The Synergy Language Profiler" in the "General Utilities" chapter of *Synergy Language Tools* for more information about routine profiling.

#41 – Cumulative XCALL profiling

System option #41 enables cumulative XCALL profiling.

If you set option #41, the profiling results for each subroutine include the CPU times for all subroutines XCALLed, rather than just the current subroutine, and any system or Synergy runtime routines.

See “The Synergy Language Profiler” in the “General Utilities” chapter of *Synergy Language Tools* for more information about routine profiling.

#42 – Profiling regardless of compiler options

System option #42 enables profiling of all routines, regardless of whether the profiling compiler option was set.

If you set option #42, all routines are profiled.

See "The Synergy Language Profiler" in the "General Utilities" chapter of *Synergy Language Tools* for more information about routine profiling.

#43 – Stop message

System option #43 controls whether a Synergy application produces a stop message when a STOP statement is executed.

If you set system option #43, no stop message is displayed, and the application terminates immediately.

If system option #43 is not set, the following message is displayed:

%DBR-S-STPMSG, STOP
%DBR-I-ATLINE, At line x in routine STOP (test.dbl)

WIN

If system option #43 is set, the message "%DBR-S-STPMSG, STOP, at line *x* in routine *xyz* (xyz.dbl)" is suppressed when a stop statement is reached. The application terminates immediately, and any text output of the program disappears from the screen.

If system option #43 is not set, a stop message is produced in one of the following two forms:

- If the application's output is not redirected, the stop message is displayed in a message box. The user must click the OK button to terminate the application. This setting allows the user to read text output from the program.
- If the application's output is redirected to a file, the stop message is printed to the file instead of being displayed in a message box.

#44 – No flush on CLOSE

UNIX

System option #44 “turns off” the **fsync()** system routine to disable flushing on a CLOSE.

If you set system option #44, the runtime won’t call **fsync()** when it closes an ISAM file. On some systems **fsync**() is quite slow, which is why you may want to turn it off.

If system option #44 is not set, the runtime calls **fsync()** each time it closes an ISAM file. Calling **fsync**() guarantees that data is written from the system cache back to disk.

#45 – Invalid key checking

System option #45 affects the way the implicit key of reference is determined for indexed READs. If the key argument to the READ statement is not a key within the specified buffer, an “Illegal key was specified” error ($ERR_BADKEY) is generated.

The checking done for system option #45 assumes that when a key field contained within the record buffer passed on the READ (or FIND) statement is used, the key is either nonsegmented or does not exceed the length of the first segment. If system option #45 is set and you change a file to use segmented keys, you must update your application to use a key buffer that is no longer contained within the record.

Option #45 is automatically set if option #35 is set.

#47 – Local message manager

UNIX, VMS

System option #47 turns off the Synergy message manager and tells Synergy Language to use the local message manager. It is the opposite of system option #7, which instructs Synergy Language to use the Synergy message manager rather than local message facilities.

On UNIX, the default is that system option #7 is *off*.

On OpenVMS, the default is that system option #7 is *on*. System option #47 turns option #7 off.

See system option #7 on page 2-8 for more information.

#48 – Initialize SQL Connection

UNIX, VMS

System option #48 signals the Synergy runtime to allocate the necessary memory for the SQL Connection. You can also set option #48 using %OPTION. When using %OPTION, SQL Connection is only initialized when %OPTION gets option #48, and SQL Connection is shutdown, releasing resources, when option #48 is not set. Using %OPTION allows only those programs using SQL Connection to use extra resources; using system option #48 allows all programs to use extra resources.

#49 – Disable debug on OpenVMS

VMS

System option #49 tells the runtime *not* to enter the debugger when you run Synergy Language programs built with the **/debug** compiler option.

#50 – Continue NOLOCK I/O when record locking error occurs

UNIX

When system option #50 is set and you get an internal runtime failure due to a record locking problem (for example, when opening a record on an NFS server that does not support record locking, such as OpenVMS), the OPEN is automatically retried using the NOLOCK qualifier.

#52 – Line profiler

System option #52 enables line-level profiling of Synergy Language programs.

If you compile a module with **/profile** (**-u** on UNIX and Windows) and set system option #52, a line execution count for the module is generated into a file called **lines.dat**. You can use the **DBLDIR:profline** program to decode this data file into a file (**lines.lst**) that contains the profile.

See "The Synergy Language Profiler" in the "General Utilities" chapter of *Synergy Language Tools* for more information about routine profiling.

#53 – RECORD defaults to LOCAL RECORD

When system option #53 is set, the default record type for routines declared with the REENTRANT qualifier is LOCAL rather than the usual default of STACK.

This option affects every invocation of the Synergy compiler.

#54 – Relax rules for compiling with -qcheck

When system option #54 is set, compiling with the **-qcheck** option (or **/CHECK=BOUNDS** on OpenVMS) only generates a subscript error for data outside the defining data area (common space, routine record space, global space, and literal space).

VMS

Only routine record space and literal space can be checked.

If system option #54 is not set, a subscript error occurs if the subscript is beyond the definition of the individual field. For example, given the following data division:

```
record
    var       ,2d4
    var1      ,d4
```

var(3) causes a subscript error if system option #54 is *not* set but not if it *is* set. However, var(4) causes an error in either case.

System option #54 will cause subscript errors if used in conjunction with arrays in class data. Therefore, it should not be used with object-oriented code.

#55 – Map Q_NO_LOCK to Q_NO_TLOCK

When system option #55 is set, READS with LOCK:Q_NO_LOCK is automatically mapped to READS with LOCK:Q_NO_TLOCK. This setting enables you to make use of the READS prefetch feature of *xf*Server (and thus improve performance when doing sequential reads) without modifying your code.

TIP Because Q_NO_TLOCK uses fewer resources than Q_NO_LOCK and provides similar performance to a READS on a file opened for input, system option #55 will provide performance gains, regardless of whether or not you use the prefetching feature.

Do not use this option if you want to see concurrent changes made by other users, as Q_NO_TLOCK does not guarantee changes will be seen. (See LOCK in the “Synergy Language Statements” chapter of the *Synergy Language Reference Manual* for more information.)

See READS in the “Synergy Language Statements” chapter of the *Synergy Language Reference Manual* for more information about the prefetch feature of *xf*Server.

#56 – Ignore GRFA on GETRFA and RFA qualifiers

WIN, UNIX

Prior to Synergy/DE 9.3.1, a GETRFA or RFA qualifier variable that exceeded 6 bytes was allowed. As of 9.3.1 and the introduction of GRFAs, however, if the size of a GETRFA or RFA argument is anything other than 6 or 10 bytes, an “Invalid record’s file address” error ($ERR_INVALRFA) is generated. Setting system option #56 turns off this behavior and causes all RFAs that exceed 6 bytes to be treated as 6-byte RFAs.

Index