

# **Synergy Tools**

**Version 10.3.3**



Printed: May 2016

The information contained in this document is subject to change without notice and should not be construed as a commitment by Synergex. Synergex assumes no responsibility for any errors that may appear in this document.

The software described in this document is the proprietary property of Synergex and is protected by copyright and trade secret. It is furnished only under license. This manual and the described software may be used only in accordance with the terms and conditions of said license. Use of the described software without proper licensing is illegal and subject to prosecution.

© Copyright 2006 – 2016 by Synergex

Synergex, Synergy, Synergy/DE, and all Synergy/DE product names are trademarks or registered trademarks of Synergex.

ActiveX, Visual Studio, Windows, Windows Vista, Windows Server, and Win32 are registered trademarks of Microsoft Corporation. Adobe, Acrobat, and Reader are registered trademarks of Adobe Systems Incorporated. All other product and company names mentioned in this document are trademarks of their respective holders.

DCN ST-01-10.3\_05

Synergex  
2330 Gold Meadow Way  
Gold River, CA 95670 USA

[www.synergex.com](http://www.synergex.com)

phone 916.635.7300

fax 916.635.6549

# Contents

## Preface

- About this manual ix
- Manual conventions ix
- Other resources x
- Product support information x
- Synergex Professional Services Group xi
- Comments and suggestions xi

## 1 Building and Running Synergy Applications

- Creating and Running Traditional Synergy Applications 1-2
  - Methods for invoking commands on Windows 1-2
  - Input/Output redirection 1-4
  - Filenames on Windows 1-4
  - Filenames on UNIX 1-5
- Compiling a Traditional Synergy Routine 1-6
  - Invoking the traditional compiler 1-6
  - Redirecting compiler commands from a file 1-23
  - Listing object file contents 1-24
  - The dbl8 compiler 1-25
- Creating and Using Libraries 1-26
  - About object and executable libraries 1-26
  - Creating executable libraries 1-28
  - Creating object libraries 1-29
  - Invoking the librarian 1-29
  - Redirecting librarian commands from a file 1-32
- Building Shared Images 1-33
- Linking Object Modules 1-37
  - Invoking the linker on Windows and UNIX 1-37
  - Redirecting linker commands from a file 1-44
  - Expanding the Synergy DBL stack through the linker 1-45

Listing executable programs	1-45
Listing ELBs	1-47
Invoking the linker on OpenVMS	1-48
Running Synergy DBL Programs	1-50
Running programs on Windows and UNIX	1-50
Running programs on OpenVMS	1-52
Non-interactive runtimes	1-52
Using dbr or dbs as a scheduled task	1-55

## 2 Debugging Synergy Programs

Introduction to the Synergy Debugger	2-3
Synergy Debugger Commands	2-12
Command recall and editing	2-12
BREAK – Set a program breakpoint	2-13
CANCEL – Cancel watchpoints and breakpoints	2-18
DELETE – Delete a program breakpoint	2-20
DEPOSIT – Assign a value to a variable	2-22
EXAMINE – Examine the contents of a variable or address	2-23
EXIT – Exit the current program with traceback	2-28
GO – Continue program execution	2-29
HELP – Provide command help information	2-31
LIST – Display lines of source code	2-32
LOGGING – Log the debugging session to a file	2-34
OPENELB – Make an ELB's subroutines available to debugger	2-35
QUIT – Quit the current program without traceback	2-36
SAVE – Save current debugger settings	2-37
SCREEN – Update the Synergy windowing system	2-38
SEARCH – Search the source for a string	2-39
SET – Set debugger options	2-40
SHOW – Examine debugger options and program state information	2-42
STEP – Step to the next Synergy DBL statement	2-46
TRACE – Display the current traceback	2-47
VIEW – Display lines around a debug entry	2-48
WATCH – Set a watchpoint	2-49
WINDBG – Invoke the UI Toolkit debugger	2-52
@ – Process an indirect command file	2-53
! – Execute system commands	2-54
Sample Debugging Session	2-55

### 3 Synergy DBMS

Synergy File Types 3-2

Synergy ISAM files 3-2

Synergy relative files 3-22

Synergy sequential files 3-25

Synergy stream files 3-27

Synergy block file I/O 3-31

Synergy counted files 3-32

Synergy DBMS Utilities 3-34

Parameter and XDL files 3-35

bldism – Create an ISAM file 3-38

chklock – Report file lock information 3-48

ctutl – Manipulate change tracking parameters 3-50

fcompare – Compare database files to system catalog or repository 3-56

fconvert – Convert database files to other file types 3-62

ipar – Generate parameter file descriptions 3-68

irecovr – Recover Revision 4 or higher ISAM files 3-71

isload – Load, unload, or clear an ISAM file 3-72

ismvfy – Verify structure of a Revision 4 or higher ISAM file 3-76

isutl – Verify, recover, and optimize Revision 4 and higher ISAM files 3-77

status – Report the status of an ISAM file 3-87

ISAM Definition Language 3-89

Rules for XDL keyword files 3-89

XDL keywords 3-90

XDL syntax checker utility 3-100

Moving Database Files to Other Systems 3-101

### 4 General Utilities

The Synergy UI Toolkit Control Panel 4-3

Before you begin 4-3

Making minor changes: Editing messages interactively 4-4

Making major changes: Unloading messages to an ASCII file 4-6

Creating your own text message files 4-7

Modifying messages at the command line 4-8

The Synergy DBL Profiler	4-11
Enabling profiling	4-11
Decoding the profile.dat or lines.dat file	4-12
Keep in mind...	4-12
The Synckini Utility	4-14
The Servstat Program	4-15
Function of the program	4-15
Running servstat with command line arguments	4-16
Servstat options	4-17
The Monitor Utility for Windows	4-21
The Monitor Utility for UNIX	4-23
When to use the Monitor	4-23
Running the Monitor	4-23
Displaying Monitor information	4-25
Sample output from the Monitor Utility	4-26
The ActiveX Diagnostic Utility	4-31
Registering an ActiveX control	4-31
Testing an ActiveX control	4-32
The Synbackup Utility	4-34
How is synbackup used?	4-34
Synbackup on Windows	4-36
Synbackup on UNIX	4-37
The Synergy Prototype Utility	4-39
The Variable Usage Utility	4-44
The Gennet40 Utility	4-46
The Dbl2xml Utility	4-51

## 5 Error Messages

About Synergy DBL Errors	5-2
Trapping runtime errors	5-2
Fatal errors	5-3
Using error literals instead of numbers	5-3
Customizing error messages	5-3

Runtime Errors	5-4
Runtime error messages	5-4
Informational error messages	5-36
Fatal error messages	5-45
Warning message	5-47
Success message	5-48
Debugging log messages	5-48
Window error messages	5-49
Compiler Errors	5-52
Fatal, nonfatal, and warning error messages	5-52
Informational error messages	5-115
Linker Errors	5-117
Fatal error messages	5-117
Informational error messages	5-122
Warning error messages	5-122
Librarian Errors	5-125
Fatal error messages	5-125
Warning error messages	5-127
Synergy DBMS Errors	5-128
List of Runtime Error Numbers	5-132

## **Appendix A: Traditional Synergy Compiler Listings**

Sample Compiler Listing	A-2
An explanation of the compiler listing	A-3
Compiler Listing Tables	A-5
Sample listing tables	A-5
An explanation of the compiler listing table	A-9

## **Index**





# Preface

## About this manual

This manual is your guide to the programming tools that are used with Synergy DBL. It documents the compiler, linker, librarian, and runtime; the debugger; the Synergy™ DBMS file management system; general utilities; and error messages.

## 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**).
- ▶ This grid indicates on which platforms and in which environments a routine, statement, etc., is supported: in traditional Synergy on Windows (WT), in Synergy .NET on Windows (WN), on UNIX (U), or on OpenVMS (V). By “supported” we mean that the item performs a useful function on that platform or environment. For example, an unsupported routine may cause a compiler error or it may just not do anything.

WT	WN	U	V
----	----	---	---

---

### 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 resources

- ▶ Synergy DBL release notes (**REL\_DBL.TXT**)
- ▶ *Synergy DBL Language Reference Manual*
- ▶ *Environment Variables & System Options*
- ▶ *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 reach the Synergy/DE™ Developer Support department at the following numbers:

800.366.3472 (in the U.S. and Canada)

916.635.7300 (in all other locations)

To learn about your Developer Support options, contact your Synergy/DE account manager at one of 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.

### Reporting Synergy .NET issues

If you are having any of the following problems, please send us the complete set of source files to re-create the issue, and send us the information in the **BuildVersion.txt** files in the \MSBuild\Synergex\dbl and \Synergex\SynergyDE\dbl directories in “Program Files” or “Program Files (x86)”.

- ▶ Visual Studio lock up or crash
- ▶ Compiler crash
- ▶ Unusual MSIL Assembler (**ilasm.exe**) issues

- ▶ “Invalid program” errors
- ▶ “JIT Compiler has encountered an internal limitation” error at runtime

For Visual Studio issues, zip the entire project.

Note that for untrapped errors, you won’t get a traceback, as you would with traditional Synergy. Instead, you’ll get the Windows Dr. Watson box. And if you click Debug, you’ll go into the debugger. If the program was not built with debug information, and you instead click Cancel, you’ll get a traceback.

## 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 the U.S. and Canada) 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](mailto:doc@synergex.com).



# 1

## Building and Running Synergy Applications

The following sections provide an overview and instructions for building and running traditional Synergy DBL programs from the command line or with Workbench. For information on creating and building traditional Synergy programs in Visual Studio, see the [“Traditional Synergy Development in Visual Studio”](#) topic in Synergy/DE WebDocs. For information on creating and building Synergy .NET programs, see [“Developing with Synergy .NET”](#) in *Getting Started with Synergy/DE*.

### **Creating and Running Traditional Synergy Applications 1-2**

Summarizes the commands used to create and run a Synergy program on Windows, UNIX, and OpenVMS.

### **Compiling a Traditional Synergy Routine 1-6**

Describes how to compile your Synergy DBL source code files into object files.

### **Creating and Using Libraries 1-26**

Gives instructions for creating object libraries using the Synergy librarian on Windows and UNIX. Also explains the difference between object libraries and executable libraries and how they can be used by your compiled Synergy DBL routines.

### **Building Shared Images 1-33**

Describes how to build shared executable images on OpenVMS.

### **Linking Object Modules 1-37**

Describes how to use the Synergy linker to combine object files into a single, executable program.

### **Running Synergy DBL Programs 1-50**

Describes how to use the Synergy runtime to execute your programs.

# Creating and Running Traditional Synergy Applications

To create and run a Synergy program, you must follow these basic steps:

Step	WIN or UNIX command	OpenVMS command
1. Compile the Synergy DBL source files into object files. (On OpenVMS this is user-definable at installation time.)	<b>dbl</b>	<b>DIBOL</b>
2. Link the generated objects into an executable program.	<b>dblink</b>	<b>LINK</b>
3. (optional) Create libraries.	<b>dblibr</b>	<b>LIBRARY</b>
4. Run the executable program.	<b>dbx</b>	<b>RUN</b>

See “[Compiling a Traditional Synergy Routine](#)” on page 1-6, “[Linking Object Modules](#)” on page 1-37, “[Creating and Using Libraries](#)” on page 1-26, “[Building Shared Images](#)” on page 1-33, and “[Running Synergy DBL Programs](#)” on page 1-50 for detailed instructions.

## Methods for invoking commands on Windows

In a Microsoft Windows environment, you can invoke the development tools and Synergy runtime from any of the following locations:

- ▶ The Command Prompt window
- ▶ The Run dialog box
- ▶ Professional Series Workbench

To create icons that run the development tools with previously specified options or to create an icon to run your application, refer to your Microsoft Windows documentation

## Invoking commands from the Command Prompt window

In the Command Prompt window, you can do any of the following:

- ▶ Enter the tool name and press ENTER. At the TOOLNAME prompt, you can enter the rest of the command line.
- ▶ Enter the full command lines. For example:  
`dblink -o util sub1 sub2 sub3 sub4 sub5`

- ▶ Enter a command line that redirects additional commands from a file. For example:

```
dblink <link.inp
```

- ▶ Enter the name of a batch file that contains the command lines. For example:

```
dblink> <link.inp
```

### Invoking commands from the Run dialog box

In the Run dialog box, you can do either of the following:

- ▶ Enter the full command line. For example:

```
dblink -o util sub1 sub2 sub3 sub4 sub5
```

- ▶ Enter a command line that redirects additional commands from a file. For example:

```
dblink <link.inp
```

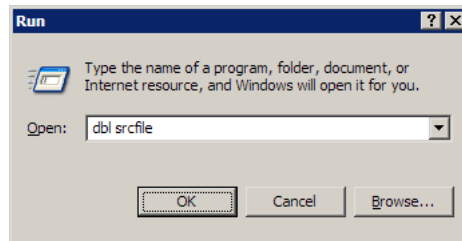


Figure 1-1. The Run dialog box.

### Invoking commands from Professional Series Workbench

If you are using Professional Series Workbench to develop your applications, you can compile, link, and run your project file directly from Workbench by selecting one of the following commands from the Project menu:

To invoke the	Select
Compiler	Compile
Linker	Build
Runtime	Execute

Refer to the [“Developing Your Application in Workbench”](#) chapter in *Getting Started with Synergy/DE* for more information about Workbench. See [“Compiling, Building, and Running”](#) in that same chapter for instructions on customizing the compile, link, and run commands.

## Input/Output redirection

Your Windows and UNIX runtimes and utilities redirect input and output as follows:

- ▶ The **UPPERCASE** option is used with input.
- ▶ **CTRL+G** is directed to the runtime as input.
- ▶ **%RDTRM** will return the proper character.
- ▶ When output is redirected to a file, traceback information and stop messages will go to the output file instead of appearing in a message box on the screen.

## Filenames on Windows

The Windows file system supports both upper and lowercase filenames, and the filenames are case insensitive. Thus, the filenames **test.dbl** and **TesT.Dbl** refer to the same file. However, the operating system stores each filename in the manner in which it was created. In a file listing of Explorer, you may see uppercase, lowercase, and mixed-case filenames. Synergy DBL lets the user determine the case of filenames. When a program creates a file, the **OPEN** statement specifies the filename case. Utility programs, such as the compiler (**dbl.exe**) and the linker (**dblink.exe**), create filenames spelled as they are on the command line. If not specified, the filename extensions default to lowercase.

You can control filename cases via the environment variable **DBLCASE**. With **DBLCASE =u**, all filenames are created in uppercase, regardless of the case used on the command line or in an **OPEN** statement. With **DBLCASE =l**, all filenames are created in lowercase. In Windows environments, **DBLCASE** has the same syntax as on UNIX. However, because environment variables are case insensitive under Windows, the “u” and “l” for logicals are ignored.

For example, if **DBLCASE** is not set:

- ▶ **dbl Test** will create the object file **Test.dbo**, regardless of what case is used for **test.dbl**.
- ▶ **dbl -o TEST Test** will create the object file **TEST.dbo**.
- ▶ **dbl -o TEST.Dbo Test** will create the object file **TEST.Dbo**.

If **DBLCASE =u**:

- ▶ **dbl Test**, **dbl -o TEST Test**, and **dbl -o TEST.Dbo Test** will create the object file **TEST.DBO**.

If **DBLCASE =l**:

- ▶ **dbl Test**, **dbl -o TEST Test**, and **dbl -o TEST.Dbo Test** will create the object file **test.dbo**.



## Filenames on UNIX

Filenames on UNIX are case sensitive. The case of the first letter of a filename is assumed to be the case for the extension if the extension is not specified. For example, if the first letter of a filename is uppercase, utility programs such as the compiler (**dbl.exe**) and the linker (**dblink.exe**) will look for that file with an uppercase extension. Thus, if a file is named **Test.dbl**, the only way to compile it is to specify the entire filename, including the extension, in the compile command. Otherwise, the compiler will assume that the file is named **Test.DBL**, and an error will be generated. Do not use the DBLCASE environment variable with this type of filename; if you do, you will not be able to open the file.

# Compiling a Traditional Synergy Routine

After creating your Synergy DBL source code files, your next step is to compile these files into object files. The object files are then used by the linker to create an executable program.



When compiling code that contains classes, do not compile classes that don't have methods, as there will be nothing to compile.

---

## Invoking the traditional compiler

### WIN, UNIX

---

```
dbl [options] [list_options] [--] source_1 [...source_n]
```

Additional ways to run the compiler in a Windows environment are explained in [“Methods for invoking commands on Windows” on page 1-2](#).

---

### VMS

---

The compiler verb is selected at install time. Here we are using “DIBOL” as an example.

```
DIBOL [options] source_1 [options]
```

or

```
DIBOL [options] source_1 [+...source_n][options]
```

---

## Arguments

*source\_1*

The first source file to be compiled. The default filename extension is **.dbl**.

*source\_n*

(optional) Represents additional source files to be compiled in the order listed. The default filename extension is **.dbl**. You can specify a maximum of 4096 files.

### WIN, UNIX

---

*options*

(optional) One or more compiler options and their arguments, shown in the [“Compiler Options” table on page 1-8](#). You can either precede a group of options with a minus sign (-), or precede each option with a minus sign. (For example, **-acdr**, **-ac -dr**, and **-a -c -d -r** are all valid.)

If an option requires an argument and that argument does not follow the option or option group immediately, Synergy DBL assumes that the option's argument is the next undefined element on the line. For example, the following **dbl** commands are equivalent:

```
dbl -ab bind_name -l list_file src_file
```

and

```
dbl -abl bind_name list_file src_file
```

In both cases, the *bind\_name* argument goes with the **b** option and the *list\_file* argument goes with the **l** option. The [Discussion on page 1-18](#) explains what happens if an argument is omitted.

#### *list\_options*

(optional) One or more of the following options, which control program listings and may override compilation flags set by the .START, .LIST, or .NOLIST compiler directives. (See the [Discussion on page 1-18](#).)

**+*[no]*list**

**+*[no]*cond**

**+*[no]*summary**

**+*[no]*offsets**

--

(optional) Separates *options* and *list\_options* from the source file list.

In Windows environments, if system option #34 is set, you must use a forward slash (/) instead of a minus sign (-) before each compiler option or group of options.

---

## VMS

### *options*

(optional) One or more compiler options and their arguments, shown in the “[Compiler Options](#)” table on page 1-8. You must precede each option with a slash (/). If an option requires an argument and that argument is not specified, Synergy DBL assumes that the option's argument is the same as the main filename.

---

## Building and Running Synergy Applications

### Compiling a Traditional Synergy Routine

## Compiler options

A complete list of the compiler options for Windows, UNIX, and OpenVMS follows.

Compiler Options			
Name	Description	WIN/UNIX option	OpenVMS option
Align data on system boundaries	Align integer types and alpha types greater than 64 bytes to a native int boundary in unnamed records. In addition, all other records and global commons are aligned to a native int boundary.	<b>-qalign</b>	<b>/ALIGN</b>
Alternate IF	Use the alternate, non-ANS DIBOL form of the IF statement, which specifies that the THEN is optional and the ELSE belongs to the last IF statement. (The ANS DIBOL form of the IF statement specifies that each ELSE belongs to the most recent THEN in the same lexical level.) For more information, see <a href="#">IF-THEN-ELSE</a> in the “Synergy DBL Statements” chapter of the <i>Synergy DBL Language Reference Manual</i> .	<b>-a</b> or <b>-q/no/altif</b>	<b>/ALTIF</b>
Alternate store	Support VAX aDIBOL-compatible zoned stores by translating spaces to zeros during alpha-to-numeric and decimal-to-decimal stores. By default, Synergy DBL ignores spaces and performs no translations during store operations. (The alternate store is significantly slower than the default.)	<b>-V</b> or <b>-q/no/altstore</b>	<b>/ALTSTORE</b>
Array size	<b>^SIZE</b> (or <b>%SIZE</b> ) of a real or pseudo array returns the size of the whole array. By default, Synergy DBL returns the size of one array element. See “ <a href="#">Array size</a> ” on <a href="#">page 1-22</a> for more information.	<b>-s</b> or <b>-q/no/decscope</b>	<b>/DECSCOPE</b>
Bind	Bind the specified name to an executable routine as a secondary main routine. If <i>bind_name</i> is not specified, the compiler uses the first source name without an extension.	<b>-b</b> [ <i>bind_name</i> ]	<b>/BIND[=SECONDARY]</b>
Bind primary	Compile <i>main</i> as the primary (entry) routine for a bound program. The default main routine name is the name of the first source name without an extension.	<b>-p</b> <i>main</i>	<b>/BIND=PRIMARY</b>

Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Bounds checking	Enable bounds checking to enforce array dimensions and string sizes to prevent subscripting off the end of an array or string. <b>-qcheck</b> converts all pseudo arrays to real arrays, converts all arguments to real arrays, checks all subscript ranging and dimension access to make sure it does not exceed the descriptor of the variable passed, and gives an error when memory used for %SSC_BIND, %SSC_DEFINE, %SSC_OPEN, %RCB_INSARG, %RCB_SETARG, or %RCB_SETARGS goes out of scope or is released. On OpenVMS, the default is <b>/CHECK=NOBOUNDS</b> .	<b>-B</b> or <b>-q/no/check</b>	<b>/CHECK=BOUNDS</b>
Common suffix	By default a dollar sign (\$) is appended to the end of common variables. This switch causes a dollar sign not to be appended to common variable names. On OpenVMS, the default is <b>/COMMON=SUFFIX</b> .	<b>-A</b> or <b>-q/no/suffix</b>	<b>/COMMON=NOSUFFIX</b>
Concatenate source files	Concatenate source files (treat them as one source) when compiling. Note that unresolved path problems due to imports may occur. The default is to treat each file as a separate source.	<b>-q/no/concat</b>	<b>/CONCAT</b>
Conditionals	Exclude conditional compiler directives and source code in false conditional blocks from the program listing. On OpenVMS, the default is <b>/SHOW= CONDITIONALS</b> .	<b>-C</b>	<b>/SHOW= NOCONDITIONALS</b>

## Building and Running Synergy Applications

### Compiling a Traditional Synergy Routine

Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Debug	<p>Manage debugging information. <i>Level</i> can be</p> <p><b>1</b> Reduce debug emission output. Field references are only forced for those in the scope of a referenced field's ancestry. Line numbers in the data division and its include files are not visible via the debugger LIST and VIEW commands. (default)</p> <p><b>2</b> Emit line number information, create a symbolic access table for all variables in the file, and provide source file relationship information for use by the debugger.</p> <p>If you don't compile and link with this option, or if you set <b>-qnodebug</b>, the compiler will not emit debugging information. On OpenVMS, the default is <b>/NODEBUG</b>.</p> <p>The <b>-d</b> option is equivalent to <b>-qdebug=1</b>.</p>	<b>-d</b> or <b>-q/no)debug[=<i>level</i>]</b>	<b>/DEBUG[=<i>level</i>]</b>
Defines at compile time	Set compile-time defines, where <i>id</i> is the name of the identifier being defined and <i>value</i> is the replacement text (if any).	<b>-qdefine:</b> <i>id1[=<i>value1</i>][,id2[=<i>value2</i>],...]</i>	<b>/DEFINE=<i>id1</i>[=<i>value1</i>] /DEFINE=(<i>id1</i>[=<i>value1</i>], <i>id2</i>[=<i>value2</i>],...)</b>
Expand macros	Include the expanded form of lines containing macros in the listing file (after the regular listing line).	<b>-qexpand</b>	<b>/EXPAND</b>
Export	Turn on the extended listing options used for Workbench.	<b>-qexport</b>	N/A
External common	Treat COMMON statements in the main routine that don't specify GLOBAL or EXTERNAL as external commons instead of global commons, which is the default.	<b>-c</b> or <b>-qexternal</b>	<b>/COMMON=EXTERNAL</b>
FIND lock	FIND statements default to locking found records. If this option is not specified, Synergy DBL does not lock the record unless the LOCK qualifier is specified. On OpenVMS, the default is <b>/NOFIND_LOCK</b> .	<b>-F</b>	<b>/FIND_LOCK</b>

Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Form feed	Form feed immediately after the data division in the program listing. (Also see the <b>-P</b> option, which performs the same task.) On OpenVMS, the default is <b>/SHOW=NEWPAGE</b> .	<b>-f</b>	<b>/SHOW=NEWPAGE</b>
Global common	Treat COMMON statements that don't specify GLOBAL or EXTERNAL modifiers as global commons instead of external commons.	<b>-G</b> or <b>-qglobal</b>	<b>/COMMON=GLOBAL</b> (default)
Global definitions	Don't make global definitions built into the compiler available to this Synergy program. (Refer to <a href="#">"Built-in compiler definitions" on page 1-21</a> for more information.) On OpenVMS, the default is <b>/GBLDEFS</b> .	<b>-g</b>	<b>/NOGBLDEFS</b>
Header	Exclude page headers and footers from the program listing. This is especially useful when directing program listings to the screen. On OpenVMS, the default is <b>/SHOW= HEADERS</b> .	<b>-h</b>	<b>/SHOW= NOHEADERS</b>
Import directories for prototyping	Specify the import directories that the IMPORT statement should search. On Windows and UNIX you can specify multiple directory locations (or logicals that contain a directory location), which will be searched in the order they appear on the command line. On OpenVMS, you can specify a single string containing a directory search path list (or logical list).	<b>-qimpdir=import_dir</b> <b>[,...]</b>	<b>/IMPDIR="import_dir</b> <b>[,...]"</b>
Include file	Specify a file to include when compiling. This option overrides the SYNUSERDEF environment variable.	<b>-quserdef=file</b>	<b>/USERDEF=file</b>
List	Generate a program listing named <i>list_file</i> . If <i>list_file</i> is not specified, the default filename is the first source file with the extension <b>.lis</b> . On OpenVMS, the default is <b>/NOLISTING</b> .	<b>-l [list_file]</b>	<b>/LISTING[=filename]</b>
Local record	Change the default behavior of unqualified RECORD statements to LOCAL.	<b>-qlocal</b>	<b>/LOCAL</b>

## Building and Running Synergy Applications

### Compiling a Traditional Synergy Routine

Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Namespaces to import	Specify namespaces to import automatically. This option overrides the SYNDEFNS environment variable. If your Synergy files do not have namespaces, they will be assigned to the default namespace, which is the first specified by <b>-qdefs</b> .	<b>-qdefs=namespace</b> [;namespace;...]	<b>/DEFNS=namespace</b> [;namespace;...]
.NET compiler warnings	Turn on .NET compiler warnings for items that are not supported in Synergy .NET. These include deprecated data types, syntax, APIs, compiler options, and alignment warnings. Warnings are output to standard error.	<b>-qnet</b>	<b>/NET</b>
No expression optimization	Don't reduce/optimize expressions.	<b>-R</b>	<b>/OPTIMIZE=value</b>
No <b>n</b> argument optimization	Relax integer optimization of <b>n</b> type arguments that are used in CASE and USING statements. Implied values passed to <b>n</b> arguments are treated as their full implied value, and using a string control variable in a USING statement causes a string comparison.	<b>-qnoargnopt</b>	<b>/NOARGNOPT</b>
No object file	Check syntax but do not create an object file. On OpenVMS, the default is <b>/OBJECT</b> .	<b>-n</b> or <b>-qnoobject</b>	<b>/NOOBJECT</b>
Numeric argument	Convert all decimal type arguments to numeric type. On OpenVMS, the default is <b>/NODECARGS</b> .	<b>-N</b> or <b>-q/no/decargs</b>	<b>/DECARGS</b>
Object	Name the object file <i>filename</i> . The default filename is the first source file with the extension <b>.dbo</b> . On OpenVMS, the default filename is the first source file with the default extension <b>.OBJ</b> . If you don't want to specify an object filename, you must specify <b>/NOOBJECT</b> .	<b>-o [filename]</b> or <b>-qobject</b> [= <i>filename</i> ]	<b>/OBJECT [=filename]</b> (default)
Offsets	Compiler-generated list of offsets into symbol table for each symbol referenced. (Compiling with <b>-d</b> or <b>/DEBUG</b> will include unreferenced symbols also.)	<b>-i</b>	<b>/OFFSETS</b>



Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Optimize	Turn optimizations on or off. <i>Level</i> can be <b>0</b> Optimizations will not occur. <b>1</b> Base optimizations will occur. (default) <b>-qnooptimize</b> is equivalent to <b>-qoptimize=0</b> . <b>/NOOPTIMIZE</b> is equivalent to <b>/OPTIMIZE=0</b> .	<b>-q/no/optimize</b> [= <i>level</i> ]	<b>/[NO/OPTIMIZE]=[<i>level</i>]</b>
Page break	Page break immediately after the data division in the program listing. (Also see the <b>-f</b> option, which performs the same task.) On OpenVMS, the default is <b>/SHOW=NONEWPAGE</b> .	<b>-P</b>	<b>/SHOW=NEWPAGE</b>
Page length	Set the length of each listing page equal to <i>length</i> . By default, the program listing will contain 60 lines.	<b>-L length</b>	<b>/PAGE_SIZE=length</b>
Platform	Specify 32-bit or 64-bit object file creation. <i>Type</i> can be <b>x86</b> 32-bit <b>x64</b> 64-bit	<b>-platform=type</b>	N/A
Profiling	Enable profiling of specific routines in the files being compiled. You must also set system option #40, #41, or #52, depending on what you want to profile. (You don't need to use this option if you want to profile all routines and have set system option #42.)	<b>-u</b> or <b>-q/no/profile</b>	<b>/PROFILE</b>
Recursion	All routines in the files being compiled can be re-entered. (You can also specify the REENTRANT modifier on the FUNCTION and SUBROUTINE statements for those routines.)	<b>-E</b> or <b>-q/no/reentrant</b>	<b>/REENTRANT</b>
Refresh	Refresh data from the disk between invocations of each routine. On OpenVMS, the default is <b>/NOREFRESH</b> .	<b>-r</b> or <b>-q/no/refresh</b>	<b>/REFRESH</b>

## Building and Running Synergy Applications

### Compiling a Traditional Synergy Routine

Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Relax strong prototyping and error checking	<p>Relax strong prototype validation. You can control which additional compiler checks are relaxed with one or more of these options:</p> <p><b>allowdup</b> Allow a duplicate structure in a namespace and duplicate routines imported from prototypes.</p> <p><b>deprecate</b> Allow deprecated syntax to pass: implied-decimal, implied-numeric, or implied-packed data types on a channel and function calls that begin with \$.</p> <p><b>end</b> Make END statement clear .DEFINES at the end of the routine instead of at the end of the file.</p> <p><b>extf</b> Don't check external function declarations against the return type of the function.</p> <p><b>interop</b> Compile classes generated by the <b>gennet40</b> utility. Identifiers longer than 30 characters are truncated instead of generating a warning.</p> <p><b>local</b> Relax error reporting on local routine prototype checking.</p> <p><b>param</b> Allow passing type <b>a</b> to an output <b>n</b> or type <b>d</b> to an input or unspecified direction <b>a</b>.</p> <p><b>paramad</b> Allow passing type <b>a</b> to an input or unspecified direction <b>d</b>.</p> <p><b>paramst</b> Allow passing a non-CLS structure (without objects) to a parameter whose type is another non-CLS structure of the same size.</p> <p><b>path</b> Help with ambiguous paths.</p> <p>With no options specified, <b>-qrelaxed</b> allows alphanumerics in unary plus operations, sizes larger than the maximum on decimal and implied-decimal fields, and EXITE to exist without a RETURN. As options are added, they add to this default relaxation.</p>	<p><b>-qrelaxed</b><i>[:option [...]]</i></p> <p>Each option must be preceded by a colon.</p>	<p><b>/RELAXED</b><i>[(=option,...)]</i></p> <p>Multiple options must be separated by commas. The parentheses are optional if only one option is specified.</p>

Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Runtime compatibility	Target an earlier runtime so code can be compiled to a previous version of the compiler. <i>Value</i> can be one of the following: <b>90501</b> Version 9.5.1 <b>90503</b> Version 9.5.3 <b>100101</b> Version 10.1.1 <b>10010101</b> Version 10.1.1a <b>10030100</b> Version 10.3.1 <b>10030101</b> Version 10.3.1a <b>10030102</b> Version 10.3.1b <b>10030103</b> Version 10.3.1c	<b>-qrntcompat=</b> <i>value</i>	
Show information	Generate extra information to the listing file. By default, the listing file contains source lines, conditional compiler directives, source lines in false conditional blocks, and page headers and footers. To exclude one or more of these items, see the “Conditionals” and “Form feed” options in this table.	<b>-C-f-h-m</b> or <b>-C-P-h-m</b>	<b>/SHOW</b> or <b>/SHOW=ALL</b> (default)
Stack record	Make STACK the default behavior of unqualified RECORD statements.	<b>-qstack</b>	<b>/STACK</b>
Static record	Make STATIC the default behavior of unqualified RECORD statements.	<b>-qstatic</b>	<b>/STATIC</b>
Stream file	When a file is opened for output with no submode, create a stream file instead of a sequential file.	N/A	<b>/STREAM</b>
Strict	Enforce strict bounds checking on real array access.	<b>-qstrict</b>	<b>/STRICT</b>
Trim	Trim trailing null arguments from a subroutine or function call.	<b>-T</b>	<b>/TRIM</b>
Truncate	Truncate subroutine, function, and variable names after the sixth character and ignore any remaining characters.	<b>-t</b>	N/A
Undefined functions	Automatically define undefined functions as ^VAL functions.	<b>-X</b> or <b>-qimplicit_functions</b>	<b>/IMPLICIT</b>

## Building and Running Synergy Applications

### Compiling a Traditional Synergy Routine

Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Variable usage	Generate a file (named <i>file</i> ) that reports unused variables. The default filename is the name of the primary source file with a <b>.unu</b> extension.	<b>-qvar_review</b> [= <i>file</i> ]	<b>/VAR_REVIEW</b> [= <i>file</i> ]
Variable usage level	<p>Specify the level of variable usage reporting. <i>Number</i> is the sum of the following bit flags that determine what is listed in the output file:</p> <ul style="list-style-type: none"><li><b>0</b> Unused local variables in each routine (default)</li><li><b>1</b> Unused global and local variables</li><li><b>2</b> Unused labels and local variables</li><li><b>4</b> Unused include files and local variables</li><li><b>8</b> Unused local variables in primary source file only</li><li><b>16</b> Referenced local variables in internal local routines, and unused local variables in each routine</li></ul> <p>You can add these bit flags together for additional combinations of reported information. For example, a value of 3 provides unused labels and unused global and local variables. See <a href="#">“The Variable Usage Utility” on page 4-44</a> for more detailed information.</p>	<b>-qreview_level</b> = <i>number</i>	<b>/REVIEW_LEVEL</b> = <i>number</i>
Variant	Define the value of the ^VARIANT data reference operation. The default variant value is 0.	<b>-v</b> <i>value</i> or <b>-qvariant</b> = <i>value</i>	<b>/VARIANT</b> =[ <i>value</i> ]
Vectors	Generate a vector list for the routines in the compiled source. For methods, the compressed mangled name is output along with the uncompressed method signature in the comment. The output can be used in the linker options file when creating an OpenVMS shared image.	N/A	<b>/VECTORS</b> = <i>filename</i>

Compiler Options (Continued)			
Name	Description	WIN/UNIX option	OpenVMS option
Warnings	<p>Control warnings. <i>Option</i> is one of the following:</p> <ul style="list-style-type: none"> <li><b>0</b> Don't generate any warning messages.</li> <li><b>1</b> Display severe warnings.</li> <li><b>2</b> Display level 1 warnings plus certain less severe warnings, such as warnings about hiding class members.</li> <li><b>3</b> Display level 2 warnings plus certain less severe warnings, such as warnings about expressions that always evaluate to true or false. (default)</li> <li><b>4</b> Display all level 3 warnings plus informational warnings. (Note that you may want to increase the value of DBLMAXERR.)</li> </ul> <p>The level of each compiler warning is specified in “Fatal, nonfatal, and warning error messages” on page 5-52.</p>	<b>-W</b> <i>[option]</i>	<b>/NOWARNINGS</b> or <b>/WARNINGS=option</b>
Warnings disabled	Disable the specified warnings. Multiple warning numbers must be separated by commas. (See “Fatal, nonfatal, and warning error messages” on page 5-52 for a list of warnings and their numbers.)	<b>-WD=error_num[,...]</b>	<b>/DISWARN=(error_num [...])</b> The parentheses are optional if only one option is specified.
Warnings to errors	Turn compiler warnings into errors.	<b>-qerrwarn</b>	<b>/ERRWARN</b>
Width	Set the width of the program listing equal to <i>list_width</i> , in columns. The default width is 132 columns.	<b>-w list_width</b>	<b>/WIDTH_SIZE =list_width</b>



The **-q** options are case insensitive and may be abbreviated to the shortest unambiguous string. For example, **-qalti** may be used for an Alternate IF or **-qalts** for Alternate store. Since profiling is the only **-q** switch starting with a p, you can use just **-qp**. The other options are case sensitive on Windows and UNIX but case insensitive on OpenVMS.

## Discussion

On Windows and UNIX, the Synergy compiler creates an object file that has the same name as the first source file with the extension **.dbo**, unless the **-o** option is used to specify a different name. All source files are included in a single object file in the order in which they are listed on the **dbl** command line.

Also on Windows and UNIX, filenames that are used as arguments to compiler options cannot begin with a minus sign (-).

On OpenVMS, if you specify more than one source file to be compiled, you must separate each filename with a plus sign (+), which causes the source files to be concatenated and compiled as one file. The result is a single object file that has the same name and location as the first source file listed, with the extension **.OBJ**, unless the **/OBJECT** compiler option is used to specify a different name. If you don't specify directories for the object and listing files, those files will be placed in the current directory, even if the source file is not in the current directory.

Also on OpenVMS, you can append one or more compiler options either to the DIBOL command or to individual source files. If you append compiler options to the DIBOL command, all of the source files listed in the command line will be affected. However, if you append compiler options to one or more source filenames, only the specified files will be affected.

## WIN, UNIX

---

### Omitted arguments

If a compiler option requires an argument and no argument immediately follows the option, the compiler will use the next undefined element on the line as the argument. (Compiler options or groups of options and the “--” separator are considered to be “defined” elements, whereas source filenames or arguments to the compiler options are considered to be “undefined.”) When the compiler encounters another compiler option group, it stops looking for the argument(s) to the previous option(s). The default argument then becomes the name of the first source file, with the appropriate extension. Once an element has been used as an argument to a compiler option, it cannot be used as an argument to another compiler option.

For example, in the following command:

```
dbl -l srcfile
```

the *list\_file* argument is omitted. The compiler will use **srcfile** as the list file and add a default extension of **.lis**. As a result, no primary source file will be found, and this command will generate a “No primary files specified” error (NULPR).

In the example below, the *list\_file* argument is omitted. Since the next element on the line is another compiler option, which is a defined element, the compiler will use **srcfile.lis** as the list file and use **srcfile.dbl** as the primary source file.

```
dbl -l -c srcfile
```

In the following example, because the “--” separator indicates that there are no more compiler options and anything that follows is a source file, **srcfile.lis** again will be the list file, and **srcfile.dbl** will be the primary source file.

```
dbl -l -- srcfile
```

In the command below, both **-l** and **-o** require arguments. Since **-o** immediately follows the **-l** specification, the compiler will use the first source filename as the list filename argument (**srcfile1.lis**). Since the **-o** option has an undefined element immediately following it, the compiler assumes this undefined element is the argument for **-o** and will name the object file **object.dbo**. **Srcfile1.dbl** will be the primary source file, and **srcfile2.dbl** the second source file.

```
dbl -l -o object -- srcfile1 srcfile2
```

### List options

On Windows and UNIX, if you specify more than one *list\_option*, separate each option with a blank space. Type out the entire *list\_option* name. If no *list\_options* are included on the **dbl** command line, all compilation flags set by the **.START**, **.LIST**, and **.NOLIST** compiler directives will be processed. For an explanation of each list option, see **.START** in the “Preprocessor and Compiler Directives” chapter of the *Synergy DBL Language Reference Manual*.

### Binding

On Windows and UNIX, the **-b** and **-p** compiler options enable you to create bound programs. Binding is a method of grouping more than one main routine into a single executable program. (See “**Bound programs**” in the “Welcome to Synergy DBL” chapter of the *Synergy DBL Language Reference Manual*.) You can create a bound application by following the steps below:

1. Compile the main routine at which the application will be entered with the **-p** compiler option.
2. Compile the other main routines with the **-b** compiler option.
3. Link all of the main routines together with their subroutines and any libraries that they use.

The example below creates a bound program named **main1.dbr**, which consists of a main routine from **main1.dbl**, two other main routines (from **main2.dbl** and **main3.dbl**) that are treated as subroutines, and whatever utility subroutines are linked into **util.elb**.

```
dbl -p main1 main1
dbl -b main2 main2
dbl -b main3 main3
dblink main1 main2 main3 util.elb
```

If you convert nonbound programs to bound programs, you might need to use the **-r** compiler option. Otherwise, your record data won’t be refreshed upon re-entry.

### Refreshing data from the disk

Note that the **-r** (or **/REFRESH**) option adds significant overhead to a routine because it has to reread the original file from the disk as it refreshes the variables.

#### Targeting a specific runtime version

The **-qrntcompat** option enables the compiler to target an earlier runtime version, back to 9.5.1. For example, you can build with Synergy/DE 10.3 but target a 9.5.3 runtime, which allows you to take advantage of the latest and greatest Synergy features for development while still supporting older runtime versions for customers that have not yet upgraded. A corresponding linker option (also called **-qrntcompat**) enables you to verify at link time that the version of an object file is not greater than the specified version. Only object files created in 10.3.3 or higher can be verified.

---

#### Bounds checking

Bounds checking can help you find subscripting errors in your code. To turn bounds checking on, specify **-qcheck** (Windows and UNIX) or **/CHECK=BOUNDS** (OpenVMS) on your compiler command line. When you run your application, the runtime will report errors if your program subscripts outside the bounds of a field. When bounds checking is specified, Synergy DBL converts all pseudo arrays (for example, **10d2**) to real arrays (for example, **[10]d2**). If you subscript array arguments, be sure to build the calling routine with bounds checking.

Keep in mind that bounds checking may also report “legal” subscripting errors. For example, the following code samples will generate subscript errors if bounds checking is turned on, even though the code is valid:

```
record
    var          , [10] d2
    .
    .
    .
    var (1,20) = "abcdef"
```

The above code is actually referencing var[1](1,20), the first element of the array, which has a length of 2. Because you are trying to write 20 characters to the two-character field, the runtime will report an error.

Suggestion: Put the array inside a group and reference the group name, or name the record and reference the record.

```
record
    var          , 10d2
    .
    .
    .
    xcall subr(var)

subroutine subr
    arg          , d2
    .
    .
    .
    arg (2) = 10
```



The reference above will report an error if the subroutine is compiled with bounds checking and the calling module is not.

```
record name
    ivar      ,i4
    i2var     ,i4
    .
    .
    .
    clear ivar(1,8)
```

Suggestion: Use `clear ^i(name)`.

```
record
    var      ,d2
    var2     ,d2
    .
    .
    .
    var(1,4) = 1010
or
    var(2) = 10
```

Suggestion: Use a group or name the record.

If you don't want to modify your code, you can also use system option #54 to relax the bounds checking rules (for Windows and UNIX) to only report an error if you subscript off the end of the defined data space for a routine. Because this incurs a lot of overhead on each subscript, we do not recommend using system option #54 for production code.

If you encounter "segmentation fault errors," we expect you to run your application with bounds checking on and then inspect the output to see if any reported problems are really problems. You may even want to modify your code so it does not try to subscript past the end of your fields. Then, when you run the bounds checking, all reported errors will be valid.

We suggest you turn bounds checking off before going to production.

### Built-in compiler definitions

The compiler defines various symbols you may want to use in your programs. These built-in system defines include operating system or environment symbols, such as `D_GUI`, `OS_VMS`, and `OS_UNIX` (as well as `DBLNET`, `D_MONO`, and `D_PORTABLE` for Synergy .NET), so you can conditionally compile your code based on the operating system on which you're running. There are also numerous values for I/O qualifiers, such as `Q_NO_LOCK` and `Q_FIRST`, and a `_DEBUG` define that enables you to include or exclude code depending on whether a program is compiled in debug mode. (Most of these definitions are listed in the **dbl.def** file, which is included in your Synergy/DE distribution for reference purposes, although it is no longer used by the compiler.) If for some reason you don't want these built-in global definitions to be available to your program, specify the **-g** compiler option (**/NOGBLDEFS** on OpenVMS).

Array size

The **-s** or **-qdecscope** options (**/DECSCOPE** on OpenVMS) affect the result of the **^SIZE** operation when used on an array. For example, let's assume you have the following array declarations:

```
array      , [10] d4
pseudo    , 10d4
```

The effect of these options is as follows:

^SIZE reference	No -s or no /decscope	-s or /decscope
^size(array)	4	40
^size(array[ ])	40	40
^size(array[1])	4	4
^size(pseudo)	4	40
^size(pseudo[ ])	40	40
^size(pseudo(1))	4	4

VMS

Debugging

Due to limitations of the DECC READ statement, specifying the **/DEBUG** option on certain combinations of stream file types produces incorrect line number information. If you're using **/DEBUG** and your line numbers for blank lines are incorrect, try converting the file to a sequential file with carriage control set to CR/LF.

Examples

WIN, UNIX

In the example below, the **main.dbl** and **util.dbl** source files are compiled, and the resulting object code is stored in **main.dbo**. A program listing is created and stored in the file **listfile.lis**. The **+list** option causes the compiler to override any **nolist** compiler options in **.START** and **.NOLIST** compiler directives in the source files. Therefore, all lines in the source files will be listed.

```
dbl -ld listfile +list main util
```

The following three examples do exactly the same thing: compile **msmenu.dbl** and generate a program listing called **msmenu.lis**. Note that in the third example, the “--” indicates the end of the option string and list options, so the compiler knows to use **msmenu** as the default argument to the **-l** option, in addition to using it as the source file.

```
dbl -l msmenu msmenu
dbl -l msmenu --msmenu
dbl -l --msmenu
```

---

### VMS

In the example below, the **MAIN.DBL** and **UTIL.DBL** source files are compiled and the resulting object code is stored in **MAIN.OBJ**. A program listing is created and stored in the file **LISTFILE.LIS**. A symbolic access table is created.

```
$ DIBOL /LISTING=LISTFILE/DEBUG MAIN+UTIL
```

---

## Redirecting compiler commands from a file

---

### WIN, UNIX

To redirect compiler commands from a file, use the following format:

```
dbl [-T] <file
```

### Arguments

#### -T

(optional) Specifies that the command line(s) should be traced, or displayed, as they are executed. If you don't specify **-T**, the command lines will not be displayed.

*file*

The ASCII file that contains one or more command lines to be input to the compiler.

### Discussion

The Synergy compiler supports continuation lines, which can make the files containing your compiler commands easier to read. If you need to continue a line to a new physical line, place the appropriate continuation character at the end of the line to be continued. The standard continuation line character is the backslash (\). On Windows, if you set system option #34, use a minus sign (-) as the continuation character.

The aggregate command line, including all continuation lines in the redirected input file, can be a maximum of 128K and 4000 files for **dbl** or 125 files for **dblink** and **dblibr**.

Examples

Assume the file *main* contains the following lines:

```
-d main util\  
sub1\  
sub2
```

If system option #34 is set on Windows, you would use the minus sign (-) as the continuation character and the forward slash (/) as the switch character. The file would look like this:

```
/d main util-  
sub1-  
sub2
```

To input the required information into the Synergy compiler from **main.com** with tracing set, type

```
dbl -T <main.cmd
```

---

Listing object file contents

WIN, UNIX

---

Once your source code is compiled, you can use the **listdbo** utility to see the internal organization of your object files. Object files are organized into object records, and **listdbo** displays the information for each object record.

This utility has the following syntax:

```
listdbo [option] obj_file [...]
```

Arguments

*option*

(optional) One of the following options:

Name	Description	Option
Dump contents	List the object record display plus the contents of each record.	<b>-d</b>
No verbose	Turn off verbose mode and provide an abbreviated listing of the object file's contents.	<b>-v</b>

*obj\_file*

One or more object files for which you want to display information.

If you run **listdbo** without any options or filenames, a dialog box will display that allows you to enter options and filenames.

---

#### VMS

---

The ANALYZE/OBJECT utility is equivalent to the **listdbo** utility.

---

## The dbl8 compiler

The dbl8 compiler was released with Synergy/DE 9 to handle specific instances in which the new compiler could not be used. Despite the name, the compiler distributed as dbl8 is not the same as the version 8 compiler. In fact, code compiled with dbl8 can be run only on Synergy/DE 9 and higher. The dbl8 compiler does not support Synergy objects. It supports all features and functions in the language as of version 8, as well as all version 8 compiler options.

You will rarely need to use dbl8, but it might be necessary on OpenVMS if your development machine has limited memory. Because the Synergy/DE compiler is designed to take advantage of memory to increase performance, if your system has less than 64 MB, we recommend that you use the dbl8 compiler instead.

# Creating and Using Libraries

This section is for Windows and UNIX only. For information on creating and using libraries on OpenVMS, see [“Building Shared Images” on page 1-33](#).

## About object and executable libraries

Your compiled Synergy DBL routines can be stored in one of the following:

- ▶ Object libraries (OLBs)
- ▶ Executable libraries (ELBs)

### Object libraries (OLBs)

An object library is a collection of object modules. You use the Synergy librarian to create object libraries. (See [“Invoking the librarian” on page 1-29](#) for details about using the librarian.) Each object library is a single file with an **.olb** extension.

Object libraries are linked into a Synergy DBL executable file with the Synergy linker. If any routine calls are unresolved, the linker looks for them in the libraries listed in the link command line. When the linker builds the executable file, it puts a copy of the referenced OLB routines in the executable file.

In the diagram below, **util<sub>2</sub>** is unreferenced; therefore, it is not included in the executable.

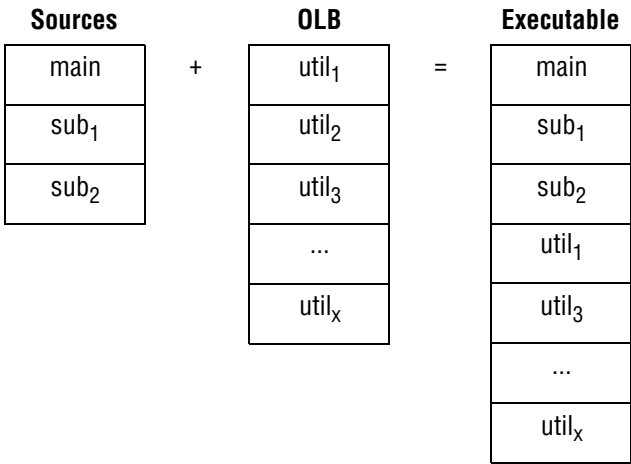


Figure 1-2. Object library.

## Executable libraries (ELBs)

An executable library is a collection of executable modules. You use the Synergy linker to create executable libraries. (See [“Linking Object Modules”](#) on page 1-37 for details on using the linker.) Each executable library is a single file with an **.elb** extension.

Executable libraries are linked against a Synergy DBL executable file with the Synergy linker. If any routine calls are unresolved, the linker looks for them in the libraries listed in the link command line. Only the executable library reference for the routines is put in the Synergy DBL executable file, not the code itself. During program execution (at runtime), the executable library is opened and the routines are accessed directly from the library file.

In the diagram below, the arrows show references between routines.

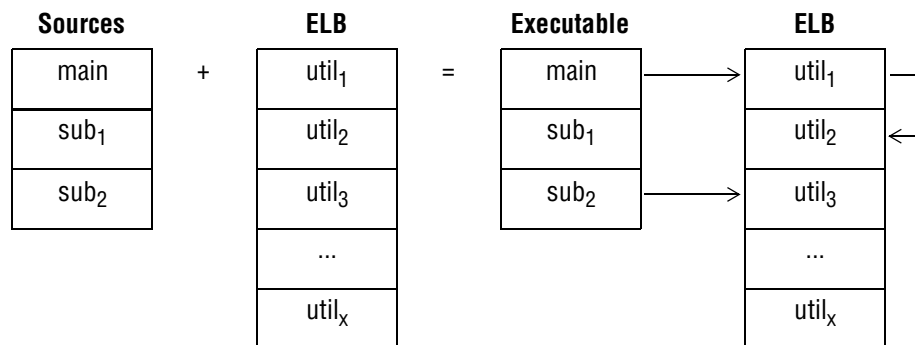


Figure 1-3. Executable library.

An executable library can also include references to executable modules in other executable libraries. When linking a Synergy DBL executable file, if any routines are unresolved, the linker looks for them in the libraries listed in the link command line and in the libraries linked against those libraries. The diagram below assumes ELB1 is linked against ELB2. During program execution (at runtime), both ELB1 and ELB2 are opened and routines are accessed from them.

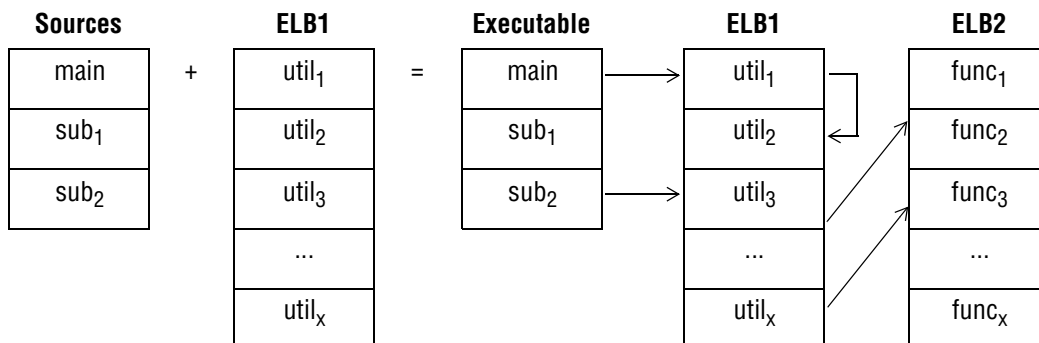


Figure 1-4. Executable library with references to other executable libraries.

## OLBs vs. ELBs

When you use executable libraries you generate smaller executables than when you use object libraries because there's only one copy of the subroutines in the ELB instead of a copy of each subroutine in every **.dbr** file. This feature has a multiplying effect on applications comprised of many programs.

An ELB subroutine is independent of the executable files that access it. You may change ELB subroutines without modifying the executable file and without relinking any programs that access the subroutines in the ELB. If you change a subroutine in an object library, you must relink all programs that use the object library to ensure they have the most recent code.

We recommend that you create ELBs when you have the choice.

## Creating executable libraries

To create an executable library (ELB), use the Synergy linker, **dblink**, with the **-I** option. You can link either object files (**.dbo** files), object libraries (**.olb** files), or other executable libraries (**.elb** files) to create your ELB.

See the **-I** option in [“Linking Object Modules” on page 1-37](#) if you want to create an ELB from object files. Also see [“Creating an ELB from an OLB” on page 1-43](#) if you want to create an ELB from an OLB.



You cannot rebuild an ELB if it is in use by another program.

Also, be careful when changing the size of a global data section: If an ELB subroutine increases the defined size of a global data section that's owned externally, and no programs are relinked against the ELB, executing those programs could cause memory access violations. If you change the size of a global data section, all programs that use it must be relinked.



## Creating object libraries

The **dblibr** command starts the librarian.

The librarian creates and maintains object libraries (OLBs). If you want to create an executable library, see [“Creating executable libraries” on page 1-28](#).

---

### VMS

Use the LIBRARIAN commands to create and maintain object libraries. See your OpenVMS documentation set for more information.

---

The various methods you can use in a Windows environment to invoke the **dblibr** command are explained in [“Methods for invoking commands on Windows” on page 1-2](#).

## Invoking the librarian

The librarian command (**dblibr**) uses the following format:

```
dblibr [options] [--] library [object_1 ... object_n]
```

### Arguments

#### *options*

(optional) One or more of the following librarian options. You can either precede a group of options with a minus sign (-) or precede each option with a minus sign. (For example, **-acdr** and **-a -c -d** are both valid.)

On Windows, if system option #34 is set, you must use a forward slash (/) instead of a minus sign (-) before each librarian option or group of options.

Librarian Options		
Name	Description	Option
Add	Add the object modules from the specified object files to the library file. If the modules already exist in the library, a warning message is generated, and the new modules will not be added to the library.	<b>-a</b>
Create	Create an object library named <i>library</i> . No warning message will be generated if the file already exists.	<b>-c</b>
Delete	Delete the specified object modules from the library file.	<b>-d</b>

Librarian Options (Continued)		
Name	Description	Option
Extract	Extract the specified object modules from the library. A separate object file is created for each extracted module. It is assigned the name of the module plus a <b>.dbo</b> extension.	<b>-x</b>
Information	Provide additional system-specific information on some fatal librarian errors.	<b>-l</b>
Replace	Replace the object modules from the specified object files in the library file. If a specified module is not found, a warning message will be generated, and the object modules will be added to the library.	<b>-r</b>
Table	Generate a list of the library's contents to your screen in alphabetical order. If you specify one or more object modules on the command line, only those modules will be listed in the table of contents. If you don't specify any object modules on the command line, all modules in the library will be listed.	<b>-t</b>
Verbose	Provide a detailed description of the object library during processing. Used in conjunction with the <b>-t</b> option, this option provides all information about the files in the library.	<b>-v</b>
Warnings	Don't print warning messages.	<b>-w</b>

--

(optional) Included for consistency with other Synergy DBL command lines but serves no function here.

*library*

The name of the object library. The default extension is **.olb**.

*object\_1*

*object\_n*

(optional) One or more object files or object modules, depending on whether you're adding to, deleting from, replacing in, or extracting from the object library. The default extension is **.dbo**. You can specify a maximum of 8192 files.

## Discussion

Libraries contain object modules, not object files. The object module name is the name of the routine loaded in the object library. (This is the name defined with the SUBROUTINE or FUNCTION statements.) When you're using the add or replace librarian option (**-a** or **-r**), specify the object file. When you're using the extract or delete librarian option (**-x** or **-d**), specify the object module.



You must specify the **-a**, **-d**, **-r**, or **-x** options on the command line unless the only other options that you specify are **-tv**. The **-c** option will default to **-a** if neither **-a** nor **-r** is specified.

---

When using the **-tv** option combination, each line of output will look something like this:

```
SXC_GLOBAL_DATA  218723  Thu Sep 06 09:06:38 2007
```

The number following the routine name (218723 in this example) is the size of the routine. (Note that this number cannot be used to approximate the runtime memory size due to data compression and other factors.)

The librarian detects and prevents duplicate routines from being added from another object library and issues a warning. It also detects methods being added to an object library and removes all existing methods from the object library that are from the same class.

## Examples

The example below creates an object library file named **screens.olb**. It contains the object code for all subroutines in the files **scr1.dbo**, **scr2.dbo**, and **scr3.dbo**.

```
dblibr -ca screens scr1 scr2 scr3
```

The following example displays a detailed table of contents of the file **screens.olb** to the screen.

```
dblibr -tv screens
```

The following example adds the object modules in **scr4.dbo** to the object library file, **screens.olb**, and displays the modules added. Any duplicate modules will be replaced without a warning message.

```
dblibr -awv screens scr4
```

The example below extracts the subroutine named **clear** from the object library **screens.olb**. The Synergy librarian creates the file **clear.dbo**, which contains the extracted object module.

```
dblibr -x screens clear
```

The following example deletes the routine **clear** from the object library named **screens.olb**.

```
dblibr -d screens clear
```

# Redirecting librarian commands from a file

To redirect librarian commands from a file, use the following format:

```
dblibr [-T] <file
```

## Arguments

**-T**

(optional) Specifies that the command line(s) should be traced, or displayed, as they are executed. If you don't specify **-T**, the command lines will not be displayed.

*file*

The ASCII file that contains one or more command lines to be input to the librarian. It cannot contain more than 2000 characters.

## Discussion

If system option #34 is set, you must use a forward slash (/) instead of a minus sign (–) before the trace flag option.

The Synergy librarian on Windows supports continuation lines in the input command file, which can make this file easier to read. If you need to continue a line to a new physical line, place the appropriate continuation character at the end of the line to be continued. The standard continuation line character on Windows and UNIX is the backslash (\). On Windows, if you set system option #34, use a minus sign (–) as the continuation character.

## Examples

```
-ca screens scr1 scr2 scr3 util1 util2 util3 \  
sutil1 sutil2
```

On Windows, if system option #34 is set, you would use the minus sign (–) as the continuation character and the forward slash (/) as the switch character. The file would look like this:

```
/ca screens scr1 scr2 scr3 util1 util2 util3 -  
sutil1 sutil2
```

To input the required information into the Synergy librarian from **libr** with tracing set, type

```
dblibr -T <libr
```

## Building Shared Images

This section describes how to build shared executable images on OpenVMS. For additional information, see your OpenVMS systems documentation.

Shared images are useful for storing commonly used subroutines or data in applications and environments. Typically, the common subroutines used for all applications are compiled into a shared image and then applications are linked to these images. This method provides two advantages: the subroutines are “shared,” and they can be maintained without rebuilding the applications. This concept is similar to the ELBs used in Synergy DBL on other platforms.

Consider the following modules of example code, which include a sample Synergy DBL main routine that writes to a global data section and a common variable, and a Synergy DBL subroutine that owns the global data section (because of the ,INIT) and the common (because it is declared GLOBAL).



Only the filename component of the *elb\_spec* passed to the OPENELB subroutine is significant. The device, directory, and file type field in the *elb\_spec* argument are ignored. By default, OpenVMS attempts to locate the ELB in SYS\$SHARE: unless the *elb\_spec* is a logical. The only way to use an ELB that is not in SYS\$SHARE is by assigning a logical to refer to it. You can then use that logical name in a call to OPENELB (or %XADDR).

---

Note that the subroutine in the shared image, **sharesub**, is called two ways: by direct reference (XCALL), which necessitates linking the main routine to the shared image using the options file, and by XSUBR, which causes the image to be loaded and the name resolved at runtime and therefore does not need the image to be linked to the shared image.

### SHAREMAIN.DBL

```
.main sharemain
global data section general_data
record
    name          ,a10
                  ,a19990                ;Spare space!
endglobal
external common
    chan          ,d2

.proc
    xcall flags(7000000, 1)
    open(1, o, 'tt:')
    chan = 1
    name = "Test!!!!!!"
    xcall sharesub
    xcall openelb("nigel")
    xcall xsubr("sharesub")
```

```
        stop
    .end
```

## SHARESUB.DBL

```
.subroutine sharesub
global data section general_data, init
record
    name      ,a10
              ,a19990          ;Spare space!
endglobal
global common
    chan      ,d2

.proc
    writes(chan, name)
    xreturn
.end
```

Compilers group object code into different program sections called psects. Each object module will contain a contribution to several psects. These psects are then collected by the linker, and the contributions from each included module are (depending on their compiler-defined characteristics) either concatenated together, or overlaid to create a program section in the final image. The following table shows the psects that Synergy DBL generates:

Psect	Contents	Read only	Read/write	Shared
\$_MDB_0	Module name information	Y	N	N
\$_MDB_1	Module name information	Y	N	N
\$_MDB_2	Module name information	Y	N	N
\$ABSS\$	External linker constants	N/A	N/A	N/A
\$CODE\$	Alpha or I64 start-up code	Y	N	Y
\$DBG\$	Debug information	Y	N	Y
\$DBL_ADDR	LABEL information	Y	N	Y
\$DBL_CODE	Interpretive code	Y	N	Y
\$DBL_COMMON	Common data	N	Y	N
\$DBL_DATA	Data division	N	Y	N
\$DBL_DESCR	Variable descriptors	Y	N	N

Psect	Contents	Read only	Read/write	Shared
\$DBL_FXD4CTL	Control information	Y	N	N (Alpha) Y (I64)
\$DBL_FXDCTL	Control information	Y	N	N
\$DBL_LINCTL	Line number information	Y	N	Y
\$DBL_LITERAL	Literals	Y	N	Y
\$DBLTRNSF_CODE	Transfer vectors	Y	N	Y
\$DBLTRNSF_LINK	Transfer vectors	Y	N	N
\$EXT\$	LINK information for external literals	Y	N	N
\$LINK\$	Linkage psect	Y	N	N
\$SYMVECT	Linker symbol vectors	N	Y	N



We recommend that you update the linker options files to include the following line:

```
PSECT_ATTR=$DBL_FXD4CTL, SHR
```

Currently on OpenVMS Alpha, the \$DBL\_FXD4CTL psect is not created sharable, but there is no reason that it cannot be sharable. Adding this line makes the attributes on the psect sharable, thereby improving overall application sharability.

The values of externally visible symbols (universal symbols that may be read by the linker, or the image loader) whether definitions of data, or executable code, must not change when a shared image is updated, to allow programs that were linked to previous versions to continue to function without relinking.

For this reason, subroutines in a shared image should be presented in the form of a transfer vector table in which the address of the routine is “aliased” to a fixed position in the image which will never move, and a jump takes place from there to the real subroutine address which is then free to move anywhere as subroutines are revised or added.

The location of the transfer vector table and shared data psects must not change in an OpenVMS shared image. For this reason, the vector table must always go first, with extra space set aside for new entries in the table, and the data psects must always follow the vector table. The transfer vector is created by instructions to the linker. See [“Invoking the linker on OpenVMS” on page 1-48](#) for details.

COMMON data and GLOBAL DATA sections are implemented as universal symbols; therefore, you cannot add a field to a common record (except in spare space at the end) or change the length of a global data section without relinking every image that is linked with the shared image.

## Building a Synergy DBL shared image

We use the linker `SYMBOL_VECTOR` command to create a vector table containing universal symbol definitions. There is no initial `CLUSTER` command. Note the spare added to the symbol vector definition to allow for future addition of modules.

Remember that data lines drawn from a command file such as those input to the linker must not begin with a dollar sign. You must indent with a space if you are wrapping an element onto the next line that begins with a dollar. (See the third `COLLECT` command and the `SYMBOL_VECTOR` command.)

```
$ DBL SHARESUB
$ LINK/NOTRACE/SHARE/EXE=SYS$SHARE:NIGEL SYS$INPUT/OPT
COLLECT = SHR_DATA,$DBL_COMMON,$DBL_DATA,$$GENERAL_DATA,CHAN$
! Above is data that the shared image shares with the outside world
! so the sizes of these psects must never change. The psects below only
! define symbols which are referenced internally to the image, so they
may
! change size and move at any time.
COLLECT = SHR_ADDRS,$DBL_DESCR,$DBL_FXDCTL,$EXT$, $LINK$
COLLECT = SHR_SHARE,$DBL_CODE,$DBL_LITERAL,$CODE, -
    $DBLTRNSF_CODE,$DBL_LINCTL,$DBG$, $DBL_ADDR
SHARESUB
SYS$SHARE:DBLTLIB/LIB
SYS$SHARE:SYNRTL/SHARE
SYMBOL_VECTOR = ( -
    SHARESUB = PROCEDURE, -
    SPARE, -
    SPARE, -
    SPARE, -
    SPARE, -
    SPARE, -
    SPARE, -
    $$GENERAL_DATA = DATA, -
    CHAN$ = DATA)
GSMATCH = LEQUAL, 1, 0
$ EOD
$
$ DBL SHAREMAIN
$ LINK/NOTRACE SHAREMAIN,SYS$INPUT/OPT
SYS$SHARE:NIGEL/SHARE
SYS$SHARE:DBLTLIB/LIB
SYS$SHARE:SYNRTL/SHARE
$ EOD
```



# Linking Object Modules

After you've compiled the source files for your program, you must combine the resulting object files into a single module that can be executed by the Synergy runtime. This step is accomplished using the Synergy linker.

---

## WIN, UNIX

The **dblink** command starts the linker.

Additional methods you can use in a Windows environment to invoke the **dblink** command are explained in [“Methods for invoking commands on Windows” on page 1-2](#).

---



---

## VMS

Use the OpenVMS LINK command to link your object files. [“Invoking the linker on OpenVMS” on page 1-48](#) explains how to use this command in greater detail.

---

## Invoking the linker on Windows and UNIX

```
dblink [options] [--] input_1 [...input_n]
```

### Arguments

*options*

(optional) One or more linker options and their arguments, shown in the [“Linker Options” table on page 1-38](#). You can either precede a group of options with a minus sign (-) or precede each option with a minus sign. (For example, **-eos** and **-e -o -s** are both valid.) If an option requires an argument and that argument does not follow the option or option group immediately, Synergy DBL assumes that the option's argument is the next undefined element on the line. For example, the following **dblink** commands are equivalent:

```
dblink -e lib_mod -s 8192 input_file
dblink -es lib_mod 8192 input_file
```

In both cases, the **lib\_mod** argument goes with the **-e** option and the **8192** argument goes with the **-s** option. The [Discussion on page 1-39](#) explains what happens if a required argument is omitted.

On Windows, if system option #34 is set, you must use a forward slash (/) instead of a minus sign (-) before each linker option or group of options.

## Building and Running Synergy Applications

### Linking Object Modules

Linker Options		
Name	Description	Option
Debug	Incorporate symbolic access table information in the <b>.dbo</b> file into the <b>.dbr</b> file for use by the debugger. If you don't compile and link with this option, symbolic information won't be available to the debugger at runtime.	<b>-d</b>
Extract	Extract the object module <i>mod</i> from object library <i>lib</i> for use as a main routine in the current program. The default library name extension is <b>.olb</b> .	<b>-e lib mod</b>
Information	Provide additional system-specific information to resolve the link problem if an internal error occurs.	<b>-l</b> (uppercase l)
Library file	Create an executable subroutine library named <i>library_file</i> . The default filename extension is <b>.elb</b> .	<b>-l library_file</b> (lowercase l)
Map file	Create an allocation map file named <i>map_file</i> . If <i>map_file</i> is not specified, the linker uses the name of the first input file plus the extension <b>.map</b> .	<b>-m[map_file]</b>
No output	Do not create an output file.	<b>-n</b>
Output file	Name the output file <i>output_file</i> . If <i>output_file</i> is not specified, the linker uses the name of the first input file plus the extension <b>.dbr</b> .	<b>-o [output_file]</b>
Unresolved references only	Only include the routines from the specified object library ( <i>olb_file</i> ) that are necessary to resolve the routines in the executable library being created. This option is used in conjunction with <b>-l</b> (lowercase l).	<b>-R olb_file</b>
Reference check	Do not allow unresolved references to subroutines in an executable library. This option is used in conjunction with <b>-l</b> (lowercase l).	<b>-r</b>
Runtime compatibility	Verify that the object file version is not greater than the version specified by <i>value</i> . (Note that object files created prior to 10.3.3 cannot have their version verified.) <i>Value</i> can be one of the following: <b>90501</b> Version 9.5.1 <b>90503</b> Version 9.5.3 <b>100101</b> Version 10.1.1 <b>10010101</b> Version 10.1.1a <b>10030100</b> Version 10.3.1 <b>10030101</b> Version 10.3.1a <b>10030102</b> Version 10.3.1b <b>10030103</b> Version 10.3.1c	<b>-qrntcompat=</b> <i>value</i>

Linker Options		
Name	Description	Option
Stack size	Set the internal operations area equal to <i>stack_size</i> . The default stack size is 256K. For more information about increasing the size of the internal operations area, see <a href="#">“Expanding the Synergy DBL stack through the linker” on page 1-45</a> .	<b>-s</b> <i>stack_size</i>
Warning	Allow unresolved references to subroutines in an executable program. If you don't use the warning option when linking a Synergy program and there are unresolved XCALLs, the linker aborts and won't create an executable file. If you do use this option, the linker creates an executable file and maps all unresolved XCALLs to an internal XCALL, which generates a “Referenced undefined XCALL” error (\$ERR_NOXCAL) when accessed at runtime.	<b>-W</b>
Warning disabled	Disable linker warnings.	<b>-wd</b> or <b>-w 0</b>

--

(optional) Separates *options* from the input file list.

*input\_1*

The first input file to be linked. The default filename extension is **.dbo**.

*input\_n*

(optional) Represents additional input files to be linked. The default filename extension is **.dbo**. You can specify a maximum of 8192 files.

## Discussion

The **dblink** command can be used to create an executable program file or an executable subroutine library.

If you are creating an executable program file, the linker creates the executable file with the same name as the first input file listed with the extension **.dbr**, unless the **-o** option (output file) is used to specify a different name.

To create an executable subroutine library, use the **-l** *library\_file* option. The linker will create the library with the name you specify. The default extension for an executable subroutine library is **.elb**. Note that **-l** can be used to link additional, existing ELBs against the ELB being created (the primary ELB), so they can be opened automatically when the primary ELB is opened. Once dependent ELBs have been linked against the primary ELB, you only need to list the primary ELB on the command line when creating an executable program. To link ELBs, list the ELBs as input files according to the specifications in [“Input files” on page 1-40](#).

#### Omitted arguments

If a linker option requires an argument and no argument immediately follows the option, the linker will use the next undefined element on the line as the argument. (Linker options and the “--” separator are considered to be “defined” elements, whereas filenames or arguments to the compiler options are considered to be “undefined.”) When the linker encounters another linker option group, it stops looking for the argument(s) to the previous option(s). The default argument then becomes the name of the first input file, with the appropriate extension. Once an element has been used as an argument to a linker option, it cannot be used as an argument to another linker option.

For example, in the following command:

```
dblink -el libA modA libB fileA
```

the arguments **libA** (with a default extension of **.olb**) and **modA** are used by the **-e** option. The argument **libB** (with an extension of **.elb**) is used by the **-l** option. The linker uses **fileA** (with an extension of **.dbo**) as the first input file.

#### Input files

The input file list can include any combination of the following types of files:

- ▶ Synergy DBL object files (**.dbo**)
- ▶ Synergy DBL object libraries (**.olb**)
- ▶ Synergy DBL executable subroutine libraries (**.elb**)

Input filename extensions always default to **.dbo**. If you’re specifying object or executable libraries, you must specify the **.olb** or **.elb** extension.



All ELBs specified on the **dblink** command line (as well as all ELBs linked to those ELBs) are automatically opened when the program is started. A maximum of 256 ELBs can be open at any one time.

The maximum length of an ELB file specification on the **dblink** command line is 31 characters.

---

Synergy DBL will use the ELB filename exactly as specified on the **dblink** command line when attempting to open the ELB at runtime. For example, let’s assume you build a program called **script** with the following **dblink** command:

```
dblink -o script script misc utils.elb
```

At runtime, Synergy DBL will expect **utils.elb** to be in the current directory. Therefore, you should include a logical that defines the location of **utils.elb**. For example:

```
dblink -o script script misc UTL:utils.elb
```



We recommend that you always include a logical when specifying ELBs.

Using logicals will help avoid two potential problems:

- ▶ Your user moves the ELB to a different directory. If you use logicals, your users can move their libraries to other directories and assign their own search paths to the new locations.
- ▶ You have a long path specification that causes the entire file specification (path and filename) to exceed 31 characters. For example, Synergy DBL will not allow you to specify the following:

```
/usr/fredrina/toolkit/common/utils.elb
```

The above file specification has 38 characters, which is too long. If the path is defined as a logical (TKUTL, for example), a shorter file specification can be used (**TKUTL:utils.elb**).

### Link procedure for .dbrs

The Synergy linker follows the steps below when linking executable programs (**.dbr**):

1. Link all object files. All object files are included in the output file in the order in which they are listed on the **dblink** command line.
2. Process all ELB files in the order in which they are listed on the **dblink** command line. If an ELB references other ELBs, those ELBs are processed and added to the list of ELB files in the header of the output file as they are referenced.
3. Add all ELB routines to the list of resolved routine names. The linker allows duplicate routine names across linked ELBs; the routines are linked in the order they are specified in the ELB.
4. Look at each OLB in order of reference. Do not start processing the next OLB until no more routine names can be resolved in the current OLB.



The linker gives ELB routines precedence over OLB routines: if a subroutine is contained in both an ELB and an OLB and both libraries are linked with a program, the subroutine will be taken from the ELB.

5. Resolve any remaining unresolved routine names from the system-supplied subroutine library, **dlib.lib**.
6. If any unresolved routine names still exist, generate an “Undefined XCALL references” (XUNDEF) error. If **-W** is specified, the undefined references are listed as warnings instead of fatal errors.

#### Link procedure for .elbs

The Synergy linker follows the steps below when linking executable libraries (**.elb**):

1. Link all object files. All object files are included in the output file in the order in which they are listed on the **dblink** command line.
2. Link all object libraries. All routines from each object library are included in the output file in the order in which they are listed on the **dblink** command line.
3. Process all ELB files in the order in which they are listed on the **dblink** command line. If an ELB references other ELBs, those ELBs are processed and added to the list of ELB files in the header of the output file as they are referenced.
4. Resolve referenced routine names from the OBJ and OLB files.
5. Add all ELB routines to the list of resolved routine names. The linker allows duplicate routine names across linked ELBs; the routines are linked in the order they are specified in the ELB.
6. Resolve any remaining unresolved routine names from the ELB files in the order in which they were processed.
7. Resolve any remaining unresolved routine names from the system-supplied subroutine library, **dlib.lib**.
8. When **-r** is specified, if any unresolved routine names still exist, generate an “Undefined XCALL references” (XUNDEF) error.

#### Unresolved references

When creating an executable program file (**.dbr**), by default the linker does not allow unresolved external references (routines, commons, or globals) and forces all references to be resolved. To override this default and allow unresolved references, use the **-W** option.

When creating an executable subroutine library (**.elb**), by default the linker allows unresolved external references. To override this default and require all references in the ELB to be resolved, use the **-r** option. Then, if there are unresolved routines, commons, or globals, the linker will abort with an error and won't create the executable library. This option can be useful if you are planning to use the ELB with xfServerPlus, because it ensures that all references have been resolved.

When **.dbo** and **.olb** files are processed, the size of any global data section is verified to ensure that the size of the referenced global does not exceed the size of the definition. If the referenced size exceeds the definition, a REFBIG warning is issued.

When references are resolved in other ELBs, those ELBs should be added to the command line so they can be searched to resolve data references and then placed in a list in the DBR or ELB file. When the program is started, those ELBs are opened automatically, as if an explicit OPENELB had been performed on them. If an ELB references other ELBs, they are opened as well (if they weren't open already).

The **-W** and **-r** options are mutually exclusive.

For debugging purposes,

- ▶ ELBs should be linked once (using **-r**) with all ELBs that the program opens with OPENELB. This ensures that all references are resolved, duplicate references don't exist, there are no duplicate globals and commons, and references to globals don't exceed the size of the defining global data section when creating **.dbr** or **.elb** files.
- ▶ DBRs should be linked once with all ELBs that the program opens with OPENELB. This ensures that duplicate routines do not exist. If the same routine exists in multiple ELBs, the one in the first ELB in the list of open ELBs will be called.

### Creating an ELB from an OLB

On Windows and UNIX, **dblink** can be used to create an ELB from an OLB. For example, the following command will link all routines in the object library **mylib.olb** into the executable subroutine library **mylib.elb**:

```
dblink -l mylib.elb mylib.olb
```

All routines in the specified object library are included in the ELB regardless of whether the references are resolved or unresolved. To include *only* routines that resolve an unresolved reference when creating an ELB, link with the **-R** option. This option is especially useful if the object library contains many routines, because it allows you to include only the routines that are necessary to resolve references, instead of including all routines from the object library.

## Examples

In the example below, the **dblink** command line links the object files **main.dbo** and **util.dbo**, and any referenced subroutines in the object library **ulib.olb** and the executable subroutine library **elib.elb**. An allocation map named **file.map** is created, and the resulting executable program is named **main.dbr**.

```
dblink -m file main util ulib.olb MYUTIL:elib.elb
```

In the example below, the command line creates an executable library named **util.elb**, which contains all of the subroutines in the files **sub1.dbo**, **sub2.dbo**, and **sub3.dbo**.

```
dblink -l util sub1 sub2 sub3
```

The example below creates an executable library named **mylib.elb** that contains the routines **sub1** and **sub2** and is linked to the ELBs **lib1.elb**, **lib2.elb**, **lib3.elb**, and **lib4.elb**, and it checks for (and disallows) any unresolved external references. (We recommend using the **-r** option if the ELBs are being linked for use with xfServerPlus.)

```
dblink -l mylib -r sub1 sub2 lib1.elb lib2.elb lib3.elb lib4.elb
```

The following is true for the example below:

- ▶ **Test1.dbl** has routines **sub1**, **sub2**, **sub3**, **sub4**, and **sub5**. Routine **sub4** has an external reference to **sub4a**. Routine **sub5** has an external reference to **sub5a**. **Test1.dbo** contains the routines from **test1.dbl**.

- ▶ **Test2.olb** contains the routines from **test2.dbl**, which has routines **sub4a**, **sub4b**, **sub4c**, and **sub4d**.
- ▶ **Test3.olb** contains the routines from **test3.dbl**, which has routines **sub5a**, **sub5b**, **sub5c**, and **sub5d**.

The example below creates the ELB containing all of the routines from **test1.dbo**, **sub4a** from **test2.olb**, and **sub5a** from **test3.olb**. The rest of the routines from **test2.olb** and **test3.olb** are not included in the ELB.

```
dblink -l test1.elb -R test2.olb -R test3.olb test1.dbo
```

## Redirecting linker commands from a file

### WIN, UNIX

---

To redirect linker commands from a file, use the following format:

```
dblink [-T] <file
```

### Arguments

#### -T

(optional) Specifies that the command line(s) should be traced, or displayed, as they are executed. If you don't specify **-T**, the command lines will not be displayed.

#### *file*

The ASCII file that contains one or more command lines to be input to the linker. It cannot contain more than 2,000 characters.

### Discussion

Synergy DBL on Windows and UNIX supports continuation lines in the input command file, which can make this file easier to read. If you need to continue a line to a new physical line, place the appropriate continuation character at the end of the line to be continued. The standard continuation line character on Windows and UNIX is the backslash (\). In Windows environments, if you set system option #34, use a minus sign (-) as the continuation character.

### Examples

```
-o EXE:main sub1 sub2 sub3 sub4 sub5 sub6 sub7 \  
sub8 ulib.olb elib.elb
```

If system option #34 is set in a Windows environment, you would use the minus sign (-) as the continuation character and the file would look like this:

```
/o EXE:main sub1 sub2 sub3 sub4 sub5 sub6 sub7 -  
sub8 ulib.olb elib.elb
```



To input the required information into the Synergy linker from **link.cmd** (without tracing), type

```
dblink <link.cmd
```

---

## Expanding the Synergy DBL stack through the linker

### WIN, UNIX

---

The Synergy DBL stack is an allocated area used by the runtime to process arithmetic expressions, subroutine arguments, and invocation controls. By default, the size of the Synergy DBL stack is 256K, which should be sufficient for most programs. The minimum stack size on UNIX is 8,192 bytes. We do not recommend lowering the stack size below its default. If the Synergy DBL stack is not large enough, the runtime will generate a “Runtime stack overflow” error (STKOVF).

You can increase the size of the Synergy DBL stack on Windows and UNIX when you link your Synergy program using the **-s** linker option. For example, the following command will set the size of the internal operations area equal to 40,000 bytes and link **recipe.dbo**, **index.dbo**, and the Synergy DBL object library **ulib.olb** to create the executable Synergy DBL program **recipe.dbr**:

```
dblink -s 40000 recipe index ulib.olb
```

---

## Listing executable programs

### WIN, UNIX

---

The **listdbr** utility displays information about the routines and global data (including initial contents) in the specified executable program(s). It also displays the names of external routines and global data referenced by each of those routines. **Listdbr** also displays the same information for each of the ELBs linked to the specified executable program(s) and any ELBs linked to those ELBs. **Listdbr** has the following syntax:

```
listdbr [option] exec_file [...]
```

### Arguments

*option*

(optional) One or more of the following options:

Name	Description	Option
ELBs only	Display the list of ELB names only.	<b>-e</b>
Extra ELBs	Include list of additional ELBs on the command line. <b>listdbr</b> loads every module in the main routine and the specified ELBS (including any ELBs linked to those ELBs) to see if any modules are undefined, which would cause an error at runtime if the module were dynamically loaded.	<b>-x</b>
Global data	Dump global data. This option is the default and is the same as not specifying any options. It will be removed in Synergy/DE 9.	<b>-g</b>
Module	Show module descriptor block of each routine. (The module descriptor block contains descriptive information about the routine, such as the routine name, program section offsets, and program section lengths.)	<b>-m</b>
Verbose	Turn off verbose mode. Provide an abbreviated listing of the executable file's contents.	<b>-v</b>
Version	Display the version of the linker used to create the <b>.dbr</b> file in the file's header.	<b>-i</b>

*exec\_file*

One or more executable programs for which you want to display information. If the **-x** option is specified, you can specify one executable program followed by one or more ELBs.

The **listdbr** utility returns an exit status of **0** if there are no undefined subroutines or global data in the executable or in any ELBs accessed by that executable. It returns a nonzero exit status if some routine or global data item is undefined.

You can use the **-x** option to find out if your ELB would have runtime failures with xfServerPlus. For example:

```
listdbr -v -x xfp1.dbr xfp1_api.elb myelb.elb
```

---

## Listing ELBs

### WIN, UNIX

The **listelb** utility displays information about the routines and global data (including initial contents) contained in the specified executable library or libraries. It also displays the names of external routines and global data referenced by each of those routines. **Listelb** can also display the same information for each of the ELBs linked to the specified ELB by using the **-l** option. **Listelb** has the following syntax:

```
listelb [option] elb_file [...]
```

### Arguments

*option*

(optional) One or more of the following options:

Name	Description	Option
ELBs only	Display the list of ELB names only.	<b>-e</b>
Linked ELBs	Include information for each of the ELBs linked to the specified ELB.	<b>-l</b>
Verbose	Turn off verbose mode. Provide an abbreviated listing of the ELB's contents.	<b>-v</b>
Version	Display the version of the linker used to create the ELB file in the file's header.	<b>-i</b>

*elb\_file*

One or more **.elb** files for which you want to display information.

### VMS

The ANALYZE/IMAGE utility is equivalent to the **listelb** utility.

## Invoking the linker on OpenVMS

To link your object files, use the OpenVMS LINK command. To link a Synergy program, use the following syntax:

```
LINK [/EXE=exe_name]object_1[,object_n,...], option_file/OPT
```

### Arguments

*exe\_name*

(optional) The name you want to use for the resulting executable file. If you don't specify *exe\_name*, the linker creates an executable file with the same name as the first object file listed, with the extension **.EXE**.

*object\_1*

The first object file or object library to be linked. If you specify an object library, you must append **/LIB** to the library name. The default filename extension for object files is **.OBJ**. The default extension for object libraries is **.OLB**.

*object\_n*

(optional) Represents additional object files or object libraries to be linked. If you specify an object library, you must append **/LIB** to the library name. The default filename extension for object files is **.OBJ**. The default extension for object libraries is **.OLB**.

*option\_file*

A file that contains special directions to the linker. A template option file for the Synergy runtime, **SYNRTL.OPT**, is located in **SYSS\$SHARE**.

### Discussion

The **SYSS\$SHARE:SYNRTL.OPT** file is a template linker options file that enables you to link programs against the Synergy runtime library. (You must use a linker options file when linking against shared images.)

#### Input files

The input file list can include any combination of the following types of files:

- ▶ Synergy DBL object files (**.OBJ**)
- ▶ Synergy DBL object libraries (**.OLB**)
- ▶ Other language object files or object libraries
- ▶ Option files, which can contain other libraries, shared images, and symbol tables

Input object filename extensions always default to **.OBJ**. Input object library extensions always default to **.OLB**. **/LIB** must be appended to each library file specification.

## Link procedures

Refer to your OpenVMS linker manual for a description of the linker algorithms.

## Break points

You can set debugger break points in modules inside shared images. Before you do so, add the following line to the linker options file:

```
$ELB_DBG=data
```

In addition, when you link the shared image, include the module **DBLDIR:ELB.OBJ**.

See [SET on page 2-40](#) for more information about setting break points.

Note that you can only have one shared image in any link command using **ELB.OBJ**. For example, if you link a shared image with this module, you can't reference additional shared images that are also linked with this module, or you'll get a "Duplicate symbol" warning.

## Examples

The following example links the object file **FRED** and the object library **WND:APLIB** into an executable file named **FRED.EXE**.

```
$ LINK FRED,WND:APLIB/LIB,SYS$SHARE:SYNRTL/OPT
```

The next example links the object file **DBLSORT** into an executable file named **SYS\$SYSTEM:DBLSORT**, with the library **DBLTLIB** and shared images **SORTSHR** and **SYNRTL**.

```
$ LINK /EXE=SYS$SYSTEM:DBLSORT DBLSORT,SYS$INPUT:/OPT
SYS$SHARE:DBLTLIB/LIB
SYS$SHARE: SORTSHR/SHARE
SYS$SHARE: SYNRTL/SHARE
```

# Running Synergy DBL Programs

## Running programs on Windows and UNIX

After you’ve compiled the sources and linked them into one executable file, use the Synergy runtime to execute your program. The **dbr** command starts the Synergy runtime. (Additional methods you can use in a Windows environment to invoke the **dbr** command are explained in [“Methods for invoking commands on Windows” on page 1-2.](#))

The **dbr** command has the following format:

```
dbr [options] [--] program [<input_file>] [>output_file]
```

### Arguments

*options*

(optional) One or more of the following runtime options. You must precede each individual option with a minus sign (for example, **-d -n**). In Windows environments, if system option #34 is set, you must use a slash (/) instead of a minus sign before each runtime option or group of options.

Name	Description	Option
Debug	Debug program while running.	<b>-d</b>
Debug after error	Launch debugger immediately following an untrapped error.	<b>-dz</b>
Remote debug	Debug program in a client/server configuration, where <i>port</i> is the port number on which the debug server will listen as a Telnet server for the debug client. Valid values are 1024 to 65535. <i>Timeout</i> is the number of seconds that the debug server will wait for a connection from the debug client. The default is 120.	<b>-rd</b> <i>port[:timeout]</i>
Terminal settings	(UNIX only) Do not change terminal (stty) settings.	<b>-r</b>
Terminal settings (program startup)	(UNIX only) Do not change terminal (stty) settings on program startup only.	<b>-x</b>

--

(optional) Included for consistency with other Synergy DBL command lines but serves no function here.

*program*

The name of the compiled and linked Synergy program. The default filename extension is **.dbr**.

*input\_file*

The name of the file from which input will be redirected.

*output\_file*

The name of the file to which output, including any error messages, will be redirected.

## Discussion

The runtime executes the program you specify.

See [chapter 2, “Debugging Synergy Programs,”](#) for more information about the debugger.

---

### WIN

Input from *input\_file* is not available using the Synergy windowing API (W\_) or Toolkit routines. READS and ACCEPT do work. If both input and output are redirected, the application runs as if **XSHOW=hide** is specified.

---

## Examples

The example below tells the runtime to get licenses from a server named **comet** if licensing is not initialized on the machine.

```
dbr -lcomet fred.dbr
```

The following example debugs the program while running in a client/server configuration. The debug server listens on port 49151 and waits for a connection from the debug client for 60 seconds.

```
dbr -rd 49151:60 fred.dbr
```

## Running applications that require elevated privileges (Windows)

If you have Synergy programs that require elevated privileges because they write to protected locations, such as Program Files or the registry, you must use the **dbrpriv** runtime. (If you're using a non-interactive runtime, use **dbspriv**; see [“The dbspriv runtime” on page 1-54.](#)) This version of the runtime has the correct embedded manifest, which will prompt for UAC elevation as required.

The **dbrpriv** command has the same format as **dbr**. See [“Running programs on Windows and UNIX” on page 1-50](#) for syntax.

## Running programs on OpenVMS

After compiling the sources, and linking them into one executable file, there are several ways to execute the file.

1. Use the DCL RUN command:

```
$ RUN filename
```

2. On OpenVMS 6.2 and above, include the location of the program in the DCL\$PATH logical. For example, if the program executable is in your home directory (SYS\$LOGIN), you could run the program as follows:

```
$ DEFINE DCL$PATH DBLDIR: , SYS$LOGIN: , SYS$DISK: [ ] , UTILS:
$ filename
```

Note that SYS\$DISK:[ ] always refers to the current working directory.

3. Define a DCL global symbol to be a foreign command to invoke the program. You *must* prefix the program name with a dollar sign (\$), even if the disk name used as part of the full program specification already begins with a dollar sign.

```
$ MYCOM*MAND: == " $SYS$LOGIN: filename "
$ mycom
```

Note that an asterisk in the definition of a foreign command indicates the shortest possible abbreviation of that command.

4. Use a command definition file to define a DCL verb to invoke the program. See your OpenVMS documentation set for information about command definition language.



Utility programs that take their parameters from the command line must be executed without the RUN command, which means that method 1 cannot be used.

---

## Non-interactive runtimes

Synergy DBL provides reduced-size, non-interactive runtimes that start up faster than the regular (**dbr**) runtime because the functions required to control an application's display and debugging capabilities — which by nature are only available to an interactive session — are minimized. A non-interactive runtime can be used to run cron jobs on UNIX or scheduled tasks on Windows, or to spawn background tasks from a **dbr**.

There are three non-interactive runtimes:

- ▶ **dbb** (Windows and UNIX)
- ▶ **dbssvc**, which is the service form of **dbb** (Windows only)
- ▶ **dbspriv**, which is the privileged form of **dbb** (Windows only)



## The dbs runtime

### WIN, UNIX

The **dbs** runtime is intended for detached programs, including xfServerPlus services. The **dbs** command has the following format:

```
dbs [options] [--] program [<input_file> [>output_file]
```

See [Arguments on page 1-50](#) for a description of the arguments. The **-n** option (detached status) is set by default. The **-rd** option is not available to the **dbs** runtime.

See “[Functionality limitations of the non-interactive runtimes](#)” on page 1-55 for a list of functions that are unavailable or limited in the **dbs** runtime.

To display version information, press ENTER at the **dbs>** prompt.

## The dbssvc runtime

### WIN

The **dbssvc** runtime is intended for Synergy programs run as services. The **dbssvc** command has the following format:

```
dbssvc option
```

*option*

One of the following options:

Name	Description	Option
Help	Print the help screen.	<b>-h</b>
Register service	Register a new service named <i>service_name</i> , where <i>display_name</i> is the display name associated with the service name, <i>program</i> is the path and filename of the Synergy program to be run when the service is started, and <i>program_args</i> are any arguments to <i>program</i> . (If any of these variables contain spaces, they must be enclosed in quotation marks.) If the optional <b>s</b> option is specified, the new service will be started after it is registered.	<b>-r[s] -c service_name -d display_name program [program_args]</b>

## Building and Running Synergy Applications

### Running Synergy DBL Programs

Name	Description	Option
Remove service	Remove the specified service.	<b>-x -c</b> <i>service_name</i>
Stop service	Stop the specified service.	<b>-q -c</b> <i>service_name</i>
Version	Display the current version.	<b>-v</b>

The service name appears as a registry key in the Windows registry. The display name will appear in the Services dialog box. The Synergy program to be run must reside on a local drive (not a mapped drive), a UNC path, or a drive that has been mapped via **subst**.

See “[Functionality limitations of the non-interactive runtimes](#)” on page 1-55 for a list of functions that are unavailable or limited in the **dbssvc** service runtime.

If you want to register a routine to be called when the program executed by **dbssvc** is stopped, use the %SYN\_ATEXIT function. (See [%SYN\\_ATEXIT](#) in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual* for details.)



We recommend that you include an outer TRY/CATCH or ONERROR with an \$ERR\_CATCH literal in programs that use the service runtime so you can trap any unhandled error, write it to the event log, and then issue a STOP statement.

---



**Dbssvc** checks for the existence of the Synergy License Manager service. If it exists, the services running under **dbssvc** will be made dependent on the License Manager service. If the machine is no longer a license server or if the License Manager service is not started, the services must be re-registered or they won't start.

---

## The dbspriv runtime

### WIN

---

The dbspriv runtime is intended for applications that require elevated privileges. **Dbsspriv** has the same format as **dbbs**. (See “[The dbs runtime](#)” on page 1-53 for syntax.) This version of the runtime has the correct embedded manifest, which will prompt for UAC elevation as required.

---

## Functionality limitations of the non-interactive runtimes

The functions below are unavailable or limited in the **dbs**, **dbssvc**, and **dbspriv** runtimes, which implement limited I/O. If you need to use these functions with xfServerPlus, set the **XFPL\_DBR** environment variable, which causes xfServerPlus to use **dbr** instead of **dbs**. See [XFPL\\_DBR](#) in the “Environment Variables” chapter of *Environment Variables & System Options* for more details.

Function	Additional information
MASK qualifier	MASK will not work on a detached program.
Message SEND/RECV	
Synergy DBL Profiler	Profiling of code is not available.
Synergy debugger	
<b>synergy.ini</b> file	Read only when SFWINIPATH is set.
<b>synuser.ini</b> file	Read only when both SFWINIPATH and SFWUSRINIPATH are set.
Terminal I/O	Only very minimal terminal I/O is available (ACCEPT, DISPLAY, READS, and WRITES to stdin/stdout for limited debugging).
TNMBR routine	The terminal number is always -1.
TTSTS routine	TTSTS on TT: is not available.
W_ routines	
WAIT routine	WAIT will not work on a detached program.
UI Toolkit	Use of Toolkit is limited to U_START, U_OPEN, U_FINISH, and similar routines that perform no terminal I/O and do not create or use windows.

## Using dbr or dbs as a scheduled task

### WIN

---

You can use **dbr** or **dbs** as a scheduled task to emulate a batch job.

Scheduled tasks using **dbr** that are run while a user is logged in will operate and display windows as if run from a command prompt. To disable this behavior, set **XSHOW=hide** in your batch file.

Scheduled tasks that are run while no user is logged in will operate as if the **TNMBR** environment variable is set to **-1** (detached). In this mode there is no user interface, and UI Toolkit user interface calls should not be made. You can test for **%TNMBR.It.0** to detect this condition.

## Building and Running Synergy Applications

### Running Synergy DBL Programs

If you use a scheduled task and want to review any error log that is output when the user is not logged in, redirect **stdout** to a file. For example:

```
dbr my_prog > my_out.log
```

---

# 2

## Debugging Synergy Programs

The information in the following sections applies to traditional Synergy only. You will use the Visual Studio .NET debugger when debugging Synergy .NET applications. (Note that you do not need to build your application in Visual Studio in order to debug it in Visual Studio.)

### Introduction to the Synergy Debugger 2-3

Describes how to invoke the debugger, create a symbolic access table, control the debugger indirectly using a command file, specify variables within the debugger, and use the debugger remotely.

### Synergy Debugger Commands 2-12

Specifies recall and editing commands and documents the following debugger commands:

BREAK – Set a program breakpoint .....	2-13
CANCEL – Cancel watchpoints and breakpoints .....	2-18
DELETE – Delete a program breakpoint .....	2-20
DEPOSIT – Assign a value to a variable .....	2-22
EXAMINE – Examine the contents of a variable or address .....	2-23
EXIT – Exit the current program with traceback .....	2-28
GO – Continue program execution .....	2-29
HELP – Provide command help information .....	2-31
LIST – Display lines of source code .....	2-32
LOGGING – Log the debugging session to a file .....	2-34
OPENELB – Make an ELB's subroutines available to debugger .....	2-35
QUIT – Quit the current program without traceback .....	2-36
SAVE – Save current debugger settings .....	2-37
SCREEN – Update the Synergy windowing system .....	2-38
SEARCH – Search the source for a string .....	2-39
SET – Set debugger options .....	2-40
SHOW – Examine debugger options and program state information .....	2-42
STEP – Step to the next Synergy DBL statement .....	2-46

TRACE – Display the current traceback..... 2-47

VIEW – Display lines around a debug entry ..... 2-48

WATCH – Set a watchpoint ..... 2-49

WINDBG – Invoke the UI Toolkit debugger ..... 2-52

@ – Process an indirect command file..... 2-53

! – Execute system commands ..... 2-54

**Sample Debugging Session 2-55**

Provides an example scenario using the debugger.

## Introduction to the Synergy Debugger

The Synergy DBL source-level debugger enables you to run your traditional Synergy programs in debug mode so you can control and examine the execution environment. The debugger supports line numbers, source display, breakpoints, watchpoints, examination by offset, `.INCLUDED` routines, dimensioned variables, and named access to fields, including complete variable path specifications.

If you want to be able to perform source displays and look up variables by their names, you must create a symbolic access table. If you don't, you will not have access to the symbolic information when debugging. To create a symbolic access table,

On	Do this
Windows and UNIX	Use the <b>-d</b> or <b>-qdebug</b> option when you compile and the <b>-d</b> option when you link. For example: <pre>dbl -d source_file dblink -d input_file</pre>
OpenVMS	Use the <b>/DEBUG</b> option when you compile, and then link normally. For example: <pre>dbl /debug source_file or dibol /debug source_file dblink source_file</pre> On OpenVMS, the main routine must also be compiled with the <b>/DEBUG</b> switch for the program to start up in debug mode.

To invoke the debugger, enter the appropriate command for your operating system, where *program* is the name of your compiled and linked Synergy program:

On	Enter
Windows and UNIX	<pre>dbr -d program</pre> or <pre>dbr -rd port[:timeout] program</pre> (See “ <a href="#">Debugging remotely</a> ” on page 2-8 for more information about the second format.)
OpenVMS	<pre>run program</pre>

Your command line prompt is

**DBG>**

(or **DblDbg>** on OpenVMS systems) and you can enter any of the debugger commands described in [“Synergy Debugger Commands”](#) on page 2-12.

---

#### VMS

If system option #49 is set, the runtime does *not* enter the debugger when you run programs built with the **/DEBUG** compiler option.

---

If you’re running your program in the debugger and a fatal error is encountered, the debugger generates the fatal error message and its traceback and break at the line that caused the fatal error. This feature enables you to investigate the circumstances that surround the error.

Online help for the debugger is available by typing

`help [command]`

where *command* is the command for which you want more information.



A system define called `_DEBUG` is defined when compiling with **-d** or **-qdebug:1**. By testing for `_DEBUG` using `.IFDEF` and `.IFNDEF`, you can conditionally include or exclude code based on whether or not a program is compiled in debug mode.

---

## Using the debugger on Windows

When you are debugging a Synergy application on Windows, the debugger output appears in a separate window. Debugger commands can be entered in this window at the prompt.

You can specify the initial size and placement of the Synergy debugger window in the **synergy.ini** file using the initialization settings `DBG_HEIGHT`, `DBG_WIDTH`, `DBG_X`, and `DBG_Y`. If the window fits on the desktop, it appears without scroll bars. If the window is resized to be smaller than the originally created size, it displays scroll bars on the window borders, which you can then move with the mouse to view the rest of the screen.

You can specify the font used in the debugger window using the `FONT_DEBUG` initialization setting. Refer to [“Using Fonts on Windows”](#) in the “Customizing UI Toolkit” chapter of the *UI Toolkit Reference Manual* for the defaulting hierarchy used when `FONT_DEBUG` is not specified. We recommend using a fixed font for the debugger.



See the “[Environment Variables](#)” chapter of *Environment Variables & System Options* for more information about the above settings.



If you move a source file and a **.dbr** file from Windows to UNIX or vice versa, you must move the files in binary mode if you want to view the source code correctly in the debugger. If you move the source file in ASCII mode (via FTP), the LF or CR-LF line terminators will not be preserved.

---

## Indirect command file processing

You can control the debugger indirectly using a command file. If an “at” sign (@) is the first character on an input line, the remainder of the line is assumed to be the name of a text file that contains debug commands to be executed.

When you specify a command file, a new command file level is activated until the last line in the file is executed. If one of the lines executed is another indirect command file specification, another level is activated until the lines in that file are executed. Up to eight levels can be activated in the debugger.

The default filename extension for a command file is **.cmd**, and full Synergy DBL-style logical name translation occurs.

---

### VMS

You can use the **DBG\$INPUT** and **DBG\$OUTPUT** logical names to redirect debugger input and output. See [DBG\\$INPUT](#) and [DBG\\$OUTPUT](#) in the “[Environment Variables](#)” chapter of *Environment Variables & System Options* for more information.

---

## Initialization file processing

You can also control the debugger using an initialization file. If you set the environment variable **DBG\_INIT** to the name of your initialization file, the debugger reads the file and executes the debugger commands within it.

The default filename extension for an initialization file is **.cmd** (**.com** on OpenVMS).

## Specifying variables

A variable specification can be a simple variable, a variable path, or a field belonging to an object instance. In fact, any variable specification that is valid during compilation is valid in the debugger, except that you can only specify a maximum of 12 elements within a given path specification. Like the compiler, the debugger requires that each variable path be unique.

A variable specification has any of the following formats:

- ▶ *term*
- ▶ *routine:term*

- ▶ *object.field*
- ▶ *record.field*
- ▶ *group.field*

where *term* is any simple variable or path specification, *routine* is the name of a routine in the current calling chain, *object* is an object instance variable name, *field* is a field name, *record* is a record name, and *group* is a group name.

For example, the following are all valid variable specifications:

```
v(1)
v(1:3)
rout:v(2,5)
grp1.fld
test:grp1[2].grp2[8].fld
myclass.myfield
(myclass)x.myfield
ns1.ns2.myclass.myfield
```

If you want, you can specify variables in symbol table offset form (as opposed to symbolic, or name-oriented, form). To use the offset form, substitute

*@offset\_code*

for each variable name in any form of the variable specification syntax above, where *offset\_code* is a decimal literal index code. (Look at a compiler listing that was created using the symbol table offsets or list compiler options; see “[Sample listing tables](#)” on [page A-5](#) for a sample compiler listing with symbol table offsets.)

For example, if you have the following variable specification in the **test** routine:

```
a[12].b[17]
```

and **a** has a symbol table offset of 5 and **b** has an offset of 10, you can reference the specification like this:

```
@5[12].@10[17]
```

You cannot mix symbolic and offset entries; entries at the same level of a variable specification must be either all symbolic or all offset. For example, let’s assume we have the following data division:

```
record
  group grp,[5]a
    fld1    ,a3
    fld2    ,a3
  endgroup
```

Assuming that the symbol table offsets are as follows:

```
grp      2
fld1     1
```

you can access **fld1** as

```
grp[3].fld1 or @2[3].@1
```

You *cannot* access it as

```
grp[3].@1 or @2[3].fld1
```

If you use the offset form, the debugger assumes that your path specification is valid. If an element is invalid, the result is undefined. You cannot specify a variable that is not in the calling chain.

To reference arguments, use

*@-index*

where *index* is the argument number.

To reference an object instance's field value, specify the object instance variable name and field name. For example,

```
x.myfield
```

To reference an object instance's field value from an ancestor class, you can cast the object instance variable to any of the ancestors of the created class. Only one cast is allowed per EXAMINE command, but the object path can be enclosed in parentheses to clarify the object being cast. Use one of the following syntaxes:

```
(class_path) handle
(class_path) handle .field
(class_path) (handle_path) .field
(class_path) (array_list[entry]) .field
```

where *class\_path* is *namespace.class* and *handle\_path* is *record.group.handle*. For example,

```
(myclass)x.myfield
```

or

```
(myclass)(x.y).myfield
```

or

```
(myclass)(myarray[0]).myfield
```

To cast a boxed object, use one of the following syntaxes:

```
(@structure_path) handle
(@structure_path) handle .field
(@structure_path) (handle_path)
```

```
(@structure_path) (handle_path) .field  
(@boxed_type) handle  
(@boxed_type) (handle_path)
```

where *structure\_path* is *namespace.structure*; *handle\_path* is *record.group.handle*; and *boxed\_type* is **a**, **d**, or **i**.

You can reference a field value for the current object instance from within an instance method by supplying the field name, or you can specify the **this** keyword. For example,

```
this.myfield
```

The path for a static field in a class is made up of two parts: the class path and the field path. The class path is made up of namespace and class identifiers which may be abbreviated on the left side as long as the path is unique, but the specified identifiers must be an exact path without any missing identifiers. The field path must be a path to a static field in the specified class but may have unspecified identifiers as long as it is unique. When in a method, the static field paths may be specified without the class path as long as the static field is a member of the same class as the method. For example,

```
myclass.myfield
```

or

```
ns1.ns2.myclass.myfield
```

You cannot reference a complex path that includes an indexer, method call, or property.

## Debugging remotely

### WIN, UNIX

---

The Synergy DBL debugger can also run in a client/server configuration, where **dbd** acts as the debug server and the debug client is any program that is capable of acting as a Telnet client.

Running the debugger remotely has several benefits:

- ▶ It is useful when the runtime is running non-interactively—for example, as a service or scheduled task on Windows, as a detached process on UNIX, under xfServerPlus or with an HTTP server application. (If you are debugging a service or other program that is normally started with **dbd** or **dbssvc**, you will need to start it with **dbd** instead. However, it is still running non-interactively, in that it doesn't interact with the desktop and it runs under the same user profile as it does in normal [nondebug] mode.)
- ▶ It can also be useful in instances where the application is highly user-interactive and using the debugger causes the program you are trying to debug to behave differently. If an application has a problem with the way a field receives focus or if an application is run with input redirected, you probably don't want the debugger window to pop up and receive focus. Let's say, for example, that you have an ActiveX event procedure hooked to the OnFocus event for the container. If you break in that routine in the regular debugger, the debugger will steal focus,

so that when you continue program execution with **GO**, the **OnFocus** event will be invoked again and you will get another break. Running the debugger remotely from a separate workstation will alleviate this problem.

- ▶ It can be helpful for cell-based debugging. When debugging a Toolkit application, the window-system displays and the debugger displays get mixed together, and you end up doing a lot of “screen redraw” commands to see what is going on. By running the debugger remotely, you can run the application on one terminal (or terminal emulator) and debug it on another.

Running the debugger remotely requires the following:

- ▶ The machines on which the debug client and debug server are running must be capable of communicating via Telnet. (**Dbr** acts as the Telnet server.) Both debug client and debug server can be on the same machine, or they can be on separate machines.



Telnet is a TCP/IP protocol for accessing remote computers. You can use whatever Telnet application you prefer (for example, the basic **telnet.exe** program that comes with Windows, the shareware QVT/Term application, or something else).

---

- ▶ TCP/IP must be configured.
- ▶ If there is a firewall between the debug client and the debug server, the firewall must be configured to allow Telnet access on the debug port number. (Most firewalls are configured to prohibit Telnet access.)
- ▶ You must have access to the machine running the debug server (**dbr**) as well as access to the external trigger that initiates events within the program (if the program is non-interactive).

To run the debugger remotely, do the following:

1. (Recommended) Compile and link with the **-d** option.
2. Start the program to be debugged with **dbr -rd** on the command line:

```
dbr -rd port[:timeout] program
```

where *port* is the port number on which the debug server will listen as a Telnet server for the debug client (1024 to 65535, inclusive), *timeout* is the number of seconds the debug server will wait for a connection from the debug client (the default is 100), and *program* is the name of your compiled and linked Synergy program. If you include the *timeout*, there cannot be a space on either side of the colon. Make sure *timeout* is lower than your client connection timeout value.



(Windows) If your program is a detached program or a service that is normally started with **db**s or **dbssvc**, your environment may change, because **dbr** always reads the **synergy.ini** file, whereas **db**s and **dbssvc** read it only when **SFWINIPATH** is set. We recommend that you use **SFWINIPATH** to point to the location of your **synergy.ini** file and thereby avoid potential problems. For more information on **db**s and **dbssvc**, see “[Non-interactive runtimes](#)” on page 1-52.

---

3. Start a Telnet session and connect to the debug server. (The Telnet application may be on the same machine as the debug server or on a separate machine.) Specify the IP address or host name of the debug server (or localhost if you are on the same machine as the debug server) and the port number you specified with the **-rd** option.

Once a connection is established, the debug session displays in the Telnet session window on the debug client machine.

4. Debug your program. (Remember that your source files must be accessible by the debug server machine if you want to view source code within the debugger.)

The debug commands WINDBG (invoke the Toolkit window debugger) and ! (invoke a shell command) are not supported by the debug server. If either command is used, an error is generated.



Most Telnet applications support paging and scrolling in the window. This provides a scrollable debug display and enables you to see more of what you are working on than in the normal Synergy debugger window.

---

5. Once debugging is complete, let the program finish running; the runtime will exit, and the Telnet session will close automatically. Optionally, you can close the session in one of the following ways:
  - ▶ Shut down the Telnet session. All breakpoints and watchpoints will be canceled, and the program will continue running in normal mode.
  - ▶ Use the debugger QUIT or EXIT command to stop the runtime and exit the program. The Telnet session will close automatically.

For more information and specific instructions for debugging when xfServerPlus is involved, see [“Debugging Your Remote Synergy Routines”](#) in the “Configuring and Running xfServerPlus” chapter of the *xfNetLink & xfServerPlus User’s Guide*.

Timeouts or other failures are logged to a file named **rd.log**, which is created in the TEMP directory when the first entry in the file is logged. This file contains the process ID of the instance of the runtime that logged entries, the date and time entries were logged, and specific messages. If you are having a problem debugging remotely, check this file first.



When using remote debugging with xfServerPlus, we recommend that you explicitly set TEMP in the Synrc node in the Windows registry, or else **rsynd** will put the log file in a system-determined location (most likely somewhere in the C:\Users path).

---

### VMS

---

To set remote debugging,

1. Compile with the /DEBUG option.
  2. Link the program as usual
  3. Define the DBG\_RMT logical. See [DBG\\_RMT](#) in the “Environment Variables” chapter of *Environment Variables & System Options* for more information.
-

# Synergy Debugger Commands

You can abbreviate any of the debugger commands or their options to one or more unique characters (for example, B for BREAK, DEL for DELETE, and SH for SHOW). There are also some exceptions (D and DE for DEPOSIT, S for STEP, SE for SET, and W and WA for WATCH), due to the evolution of the command set. If the first token on a debug command line is a semicolon (;), the rest of the line is ignored.

## Command recall and editing

You can recall and edit debugger commands using the following control characters:

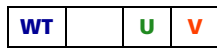
CTRL+P	Recall previous command.
CTRL+N	Recall next command.
CTRL+B	Move backward within the line.
CTRL+F	Move forward within the line.
CTRL+H	Delete previous character within the line.
CTRL+K	Delete current character within the line.
CTRL+U	Delete to the beginning of the line.
CTRL+E	Delete to the end of the line.

On a PC or VTxxx terminal, the UP ARROW and DOWN ARROW keys recall the previous command and the next command, respectively. The LEFT ARROW and RIGHT ARROW keys and the REMOVE key are automatically mapped to the backward, forward, and delete current character functions, respectively.

The command recall buffer handles 240 characters. The length of your commands determines how many commands the recall buffer can hold.



## BREAK – Set a program breakpoint



```

BREAK routine
BREAK line
BREAK routine : line
BREAK label [/LABEL]
BREAK .
BREAK method
BREAK method : line
BREAK method#id
BREAK method#id : line
BREAK method#ALL
BREAK method (signature)
BREAK image/routine

```

### Arguments

*routine*

Sets a breakpoint on entry to the specified routine.

*line*

Sets a breakpoint at the specified source line in the current routine.

*routine:line*

Sets a breakpoint at the specified source line in the specified routine.

*label* [/LABEL]

Sets a breakpoint at the specified label in the current routine.

.

(period) Sets a breakpoint at the current line in the current routine.

*method*

Sets a breakpoint on entry to the specified method.

*method:line*

Sets a breakpoint at the specified source line in the specified method.

*method#id*

Sets a breakpoint on entry to the specified implementation of the specified method.

*method#id:line*

Sets a breakpoint at the specified source line in the specified implementation of the specified method.

*method#ALL*

Sets a breakpoint on entry to all implementations of the specified method.

*method(signature)*

Sets a breakpoint on entry to an explicit method.

*image/routine*

Sets a breakpoint on entry to the specified routine that is inside the specified shared image. (OpenVMS only)

## Discussion

The **BREAK** command sets a program breakpoint, which is a point at which your program stops and goes into the debugger.

You can specify two kinds of breakpoints: entry breakpoints and specific breakpoints. An entry breakpoint causes the program to break upon entering a routine. You can set a break at the entry to a routine using the **BREAK *routine*** syntax, and to a method using the **BREAK *method*** syntax. A specific breakpoint causes the program to break at a specific line in a routine or method. You can set a specific breakpoint using the **BREAK *line***, **BREAK *routine:line***, **BREAK *label***, **BREAK .**, **BREAK *method:line***, or **BREAK *method#id:line*** syntax.

When a breakpoint occurs, the break line has not yet been executed.

When specifying a line number with the **BREAK *routine*** or **BREAK *method*** syntax, the colon can be replaced with a space.

You can specify more than one breakpoint, separated by commas. If *routine* is not specified, the break specification is assumed to be for the current routine.

You can set breakpoints in routines that are **.INCLUDEd** into another routine. To do so, specify each one in the form

*source\_file#.line#*

You can determine the source file number by viewing a listing file.

The maximum number of breakpoints is 32.

If you try to set a breakpoint in a method whose name is overloaded within the class or whose specified name matches methods in more than one class, the debugger presents a numbered selection list that includes the method name and its parameter types to allow you to select which method should have the breakpoint.

For example,

```
DBG> break testdrive
Found multiple matches:
1: BREAKMETH.CAR.TESTDRIVE()
2: BREAKMETH.CAR.TESTDRIVE(A)
3: BREAKMETH.CAR.TESTDRIVE(A,A,I)
*** Choose which breakpoint to set
DBG> break testdrive #2
DBG> show break
BREAKMETH.CAR.TESTDRIVE(A) on entry
```

You can either set the breakpoint using one of the unique identifiers, as shown above in the line

```
DBG> break testdrive #2
```

or you can set the breakpoint for all of them, like this:

```
DBG> break testdrive #all
```

You can use the *method#id* syntax at any time to set a breakpoint to a particular method, even without a set break attempt generating the list of overloaded methods.

You can alternatively specify an overloaded method by specifying the signature, or parameter list, enclosed in parentheses. (A method does not have to be overloaded to use this syntax, although the signature is not required for a nonoverloaded method.) For example,

```
BREAK mymethod(i, i)
```

or

```
BREAK myclass.mymethod(a, @Class1)
```

The parameter list is a comma-delimited list of one or more of the following parameter specifications:

Parameter specification	Description
A	Parameter is of type alpha
D	Parameter is of type decimal
I	Parameter is of type integer
\$struct	Parameter is a structure
@class	Parameter is a class handle
^VAL	Parameter is passed by value
^REF	Parameter is passed by reference

Parameter specification	Description
@\$struct	Parameter is a boxed structure
@A	Parameter is a boxed alpha
@D	Parameter is a boxed decimal
@ID	Parameter is a boxed implied-decimal
@P	Parameter is a boxed packed
@IP	Parameter is a boxed implied-packed
@I	Parameter is a boxed integer

Except for ^VAL and ^REF, each parameter specification can optionally be preceded by a dimension specification.

A method signature that has a real array parameter must specify it by a left square bracket ([]) followed by the number of dimensions. A method signature that has a dynamic array parameter must specify it by a left curly brace ({} followed by the number of dimensions. In either case, if the number of dimensions is one, the number of dimensions may be omitted.

For example,

```
method mthd
arg1, [*]a
arg2, [*,*]d
arg3, [#]@cls
arg4, [#,#] [#]@cls
proc
    mreturn
end
```

```
BREAK ns.cls.mthd([A, [2D, {1@cls, {2{ @cls)
```

If the signature doesn't match a single method implementation exactly, the debugger displays a list of one or more choices, all having the same number of arguments as the signature you specified.

## VMS

You can also set breakpoints in routines inside a shared image using the **BREAK *image/routine*** syntax. To do so, you must have done the following when linking the shared image file:

- ▶ Included \$ELB\_DBG=DATA within the "SYMBOL\_VECTOR=" line of the options file used
- ▶ Included DBLDIR:elb.doj

Only one shared image at a time can be built in this manner.

The logical used to reference the shared image must be used as the *image* part of the *image/routine* break specification.

---

## Examples

The following example sets a breakpoint at lines 7 and 10 in the current routine and also at line 5 in routine ABC.

```
BREAK 7, abc 5, 10
```

The following OpenVMS example sets a breakpoint at the entry of the **post\_data** routine in the shared image referenced by the MCBA\_LIB logical.

```
BREAK MCBA_LIB/post_data
```

The example below shows a breakpoint being set in a .INCLUDEd routine.

```
Break at 4 in MYFILE (myfile.dbl) on entry
  4>          xcall flags(1001010, 1)
DBG> set break 2.2
DBG> go
Break at 2.2 in MYFILE (MYFILEA.DBL)
  2.2>          nop
DBG> step
Step to 2.3 in MYFILE (MYFILEA.DBL)
  2.3>          writes(1, "Exiting include file")
```

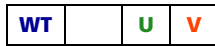
The example below breaks at line 14 within the method **mynamespace.myclass.mymethod**.

```
BREAK mynamespace.myclass.mymethod:14
```

The following example breaks in the third method of **myclass** at line 53.

```
BREAK myclass#3:1.53
```

## CANCEL – Cancel watchpoints and breakpoints



CANCEL/ALL

CANCEL WATCH [/ALL | *variable* | *address* | *index*]

CANCEL BREAK [/ALL | *line* | *routine* | . | *routine line*]

### Arguments

#### /ALL

(optional) Cancels all watchpoints and breakpoints (in the CANCEL/ALL syntax), all watchpoints (in the CANCEL WATCH/ALL syntax), or all breakpoints (in the CANCEL BREAK/ALL syntax).

#### *variable*

(optional) Cancels a watchpoint for the specified variable. See [“Specifying variables” on page 2-5](#) for more information about variable specifications.

#### *address*

(optional) Cancels a watchpoint for the specified address.

#### *index*

(optional) Cancels a watchpoint for the specified index code (or offset code). (For example, if you set three watchpoints, you can cancel the third using 3 as the index.)

#### *line*

Cancels a breakpoint at the specified source line in the current routine.

#### *routine*

Cancels a breakpoint on entry to the specified routine.

.

(period) Cancels a breakpoint at the current line in the current routine.

#### *routine:line*

Cancels a breakpoint at the specified source line in the specified routine.

### Discussion

The CANCEL command cancels one or more existing program watchpoints or breakpoints.

The watchpoint that you specify must already exist. Use the SHOW WATCH command to get a list of existing watchpoints.

## Examples

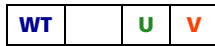
The following example cancels a watchpoint on the **dvar** variable.

```
CANCEL WATCH dvar
```

The example below cancels all breakpoints.

```
CANCEL BREAK /all
```

## DELETE – Delete a program breakpoint



```
DELETE /ALL
DELETE routine
DELETE line
DELETE routine line
DELETE label [/LABEL]
DELETE .
DELETE image/routine
```

### Arguments

#### **/ALL**

Deletes all breakpoints.

#### *routine*

Deletes a breakpoint at the entry to the specified routine.

#### *line*

Deletes a breakpoint at the specified source line in the current routine.

#### *routine line*

Deletes a breakpoint at the specified source line in the specified routine.

#### *label* [/LABEL]

Deletes a breakpoint at the specified label in the current routine.

.

(period) Deletes a breakpoint at the current line in the current routine.

#### *image/routine*

Deletes a breakpoint on entry to the specified routine that is inside the specified shared image.  
(OpenVMS only)

### Discussion

The DELETE command deletes one or more existing program breakpoints.

The breakpoint that you specify must already exist. Use the SHOW BREAK command to get a list of existing breakpoints.

You can specify more than one breakpoint for deletion, separated by commas. If *routine* is not specified, the break specification is assumed to be for the routine most recently specified on the command line.



## Examples

The following example deletes the breakpoint at the entry to the MAINT routine as well as the breakpoints at lines 12 and 19 of the ADD routine.

```
DELETE maint, add 12, 19
```

## DEPOSIT – Assign a value to a variable



DEPOSIT *variable*=*value*

### Arguments

*variable*

A variable specification to which a value is assigned. See [“Specifying variables” on page 2-5](#) for more information about variable specifications.

*value*

Either a literal or variable to assign to *variable*.

### Discussion

The DEPOSIT command assigns a value to a variable.

You must be able to read or write to a variable specification, but a data specification can be read-only (in other words, a literal).

You can specify more than one *variable=value* assignment, separated by commas. When you specify more than one assignment, they’ll be processed from left to right.



^VAL routine arguments cannot be deposited.

---

If *variable* is a field in an object instance, the field must be accessible and writable.

### Examples

The following example assigns the value **15** to the expression **a(b)** in the **xyz** routine. (Remember, **xyz** must be in the current calling chain.)

```
DEPOSIT xyz:a(b)=15
```

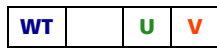
In the example below, **i** is set to 15 and **x(15)** is set to 12.

```
DEPOSIT i=15, x(i)=12
```

The example below deposits a nonobject value into a static field by specifying a fully qualified name.

```
deposit ns1.ns2.myclass.myfield = value
```

## EXAMINE – Examine the contents of a variable or address



EXAMINE [/PAGE] *variable* [/X] [*display\_option*] [*object\_option*] [*variable ...*]

EXAMINE [/PAGE] *address* [*display\_option*] [*address ...*]

EXAMINE [/PAGE] /OBJECT\_ID *object\_id* [*object\_option*] [*object\_id ...*]

EXAMINE [/PAGE] /STATIC *class\_name* [*class\_name ...*]

### Arguments

#### /PAGE

(optional) Stops the output every 24 lines and waits for input (CR to get the next page and <EOF> to terminate input). On Windows, the output will vary based on the values of DBG\_HEIGHT and DBG\_WIDTH. (/PAGE can alternatively be placed at the end of the line.)

#### *variable*

A variable specification whose contents will be displayed. See “[Specifying variables](#)” on [page 2-5](#) for more information about variable specifications. *Variable* can also be a ^M(*struct\_fld, handle*) specification.

#### /X

(optional) Displays the variable’s address.

#### *display\_option*

(optional) These qualifiers display the contents of the variable (or the address) as the specified data type.

**/I[n]I[size]** Integer or integer of length *size*, where *size* is 1, 2, 4, or 8 and *n* is the number of I[size] fields.

**/Dsize[.prec]** Decimal or implied-decimal of *size* bytes and the specified precision.

**/Asize** Alphanumeric of *size* bytes.

**/HEX or /H** Hexadecimal. (This does not apply to handles.)

#### *object\_option*

(optional) One or more of the following options:

**/INFORMATION** Display detailed information.

**/LINES** Display the first referencing source line.

**/REFERENCES** Display all other references to the object.

**/SCOPE** Display all accessed (active) handles within the object’s scope. (This does not apply to object IDs, only handles.)

*address*

An address specification whose contents will be displayed.

**/OBJECT\_ID**

Indicates that an object identifier follows.

*object\_id*

An object identifier obtained from the SHOW CLASSES /OBJECTS command.

**/STATIC**

Display all static fields within the specified class.

*class\_name*

(optional) A class name path.

## Discussion

The EXAMINE command displays the contents of a variable, address, object, or class, depending on which syntax is used. You can specify more than one variable, address, object, or class by separating them (along with their options) with commas or spaces.

Depending on what is being examined, the debugger displays the name of each “outer” field in the record, group, or structure, along with its data type, size, and contents. For example, if **rec** is examined below, the fields **rflld1**, **grp**, and **rflld2** will be displayed. None of the fields inside **grp** (**fld1**, **fld2**, or **fld3**) will be displayed.

```
record rec
    rflld1    ,a4
    group grp
        fld1  ,a4
        fld2  ,a4
        fld3  ,a4
    endgroup
    rflld2    ,a4
```

For records, groups, structures, and object instances, any arrayed fields display the data for each element of the array. Any group fields display the data in the format of the data type of the group. If *variable* is an object instance, the debugger displays each field in the object instance with its data. The class name is displayed first, followed by the data in alpha format if an object has been instantiated, or the word “^NULL” if an object has not been instantiated.

Unnamed fields will not be displayed.

If you examine a variable whose contents are longer than one line (80 characters), the variable’s contents are displayed on multiple lines. Any nonprinting characters are displayed as periods (.). On Windows, the output will vary based on the values of DBG\_HEIGHT and DBG\_WIDTH.

An arraylist is an array of System.Object, and since every object inherits from System.Object, an arraylist can hold any type of object. When an arraylist variable is examined, the count and capacity are displayed. When an arraylist cell is examined, the class of the object is displayed along with the

object value(s). To examine the elements of the contents of an arraylist, you must cast the arraylist variable to the correct class of the arraylist cell contents, because the object's fields are not members of the arraylist. Only one cast is permitted per EXAMINE command, but the object path can be enclosed in parentheses to clarify the object being cast. (See [Examples](#) for an example of examining an arraylist variable.)

To examine an argument, use

*@-index*

where *index* is the argument number. Arguments are displayed as their passed data type unless overridden.

When specified in conjunction with an object *variable*, the /INFORMATION option displays the following information about the specified object variable:

- ▶ The handle ID and class of the object variable
- ▶ The object ID and class of the instantiated object (if instantiated)
- ▶ The number of references to the object (if instantiated)
- ▶ Any circular references

When specified in conjunction with an *object\_id*, the /INFORMATION option displays the following information about the specified object:

- ▶ The class of the object
- ▶ The number of references to the object
- ▶ Any circular references

## Examples

The following example displays the contents of the variable **ab**.

```
EXAMINE ab
```

Given the following definition:

```
class c1
    c1_fld1    , a4
    c1_fld2    , a4
    c1_fld3    , a4
endclass
class c2
    record crec
        c2_fld1    , a4
        c2_fld2    , a4
        c2_fld3    , a4
    endclass
endclass
```

```
record
    hnd1 ,@c1
    hnd2 ,@c2
```

The command

```
DBG> examine hnd1
```

displays the fields **c1\_fld1**, **c1\_fld2**, and **c1\_fld3**, while

```
DBG> examine hnd2
```

displays only **crec**.

The example below examines the contents of an address.

```
DBG> examine fld1
1234
DBG> examine fld1/X
12542980
DBG> examine 12542980
1234
DBG> examine 12542980/I
875770417
DBG> examine 12542980/I/H
^x(34333231)
```

The example below examines an object ID.

```
DBG> examine /object_id 1
Object id 1, class objid.car, refs 1
DBG> examine /object_id 1 /REFERENCES
Object id 1, class objid.car, refs 1
Other referencing handles:
Handle id 2, class objid.car
DBG> examine /object_id 1 /LINES
Object id 1, class objid.car, refs 1, def at line 45 in CD_MAIN
(exam_objid.dbl)
DBG> examine /object_id 1 /INFORMATION
Object id 1, class objid.car, refs 1
```

The example below examines an object variable.

```
DBG> examine ford
cartyp, a22, "Mustang b...0400000000"
price, d6.2, 8800.00
limit, i4, 600
DBG> examine ford /lines
Handle id 2, class objid.car, set at line 45 in CD_MAIN
(exam_objid.dbl)
Object id 1, class objid.car, refs 1, def at line 45 in CD_MAIN
(exam_objid.dbl)
```

```
DBG> examine ford /ref
    Handle id 2, class objid.car
    Object id 1, class objid.car, refs 1
DBG> examine ford /scope
    Handle id 2, class objid.car
    Object id 1, class objid.car, refs 1
    No active handles within object's scope
DBG> examine ford /info
    Handle id 2, class objid.car
    Object id 1, class objid.car, refs 1
```

The example below examines an arraylist variable.

```
DBG> examine m_collection
    <system.collections.arraylist>
    Count = 2, Capacity = 16
DBG> examine m_collection[0]
    <testing.leaf>
    m_value, @system.string, "Value 1"
DBG> examine (testing.leaf)(m_collection[0]).m_value
    "Value 1"
```

## EXIT – Exit the current program with traceback

WT		U	V
----	--	---	---

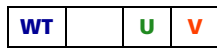
EXIT

### Discussion

The EXIT command exits the current program with traceback information and returns you to the operating system prompt. On exit, all FINALLY blocks and object destructors are called before the runtime exits. Any breakpoints in these code sections will operate normally.



## GO – Continue program execution



GO  
GO /*option*  
GO *routine*  
GO *line*  
GO *routine line*  
GO .

### Arguments

*option*

One of the following options:

<b>COUNT</b>	Continue execution through <i>count</i> breaks, displaying each.
<b>DEBUG</b>	Continue execution until the next routine that is compiled in debug mode is entered.
<b>EXIT</b>	Continue execution until the current function or subroutine is exited.
<b>NEXT</b>	Continue execution until the next function or subroutine is entered.
<b>NODEBUG</b>	Cancel debugging and continue execution as if the program had never entered the debugger. This option cancels all breakpoints and watchpoints.
<b>RETURN</b>	Continue execution until a RETURN is executed for the current CALL.

*routine*

Continues execution until the specified routine is entered.

*line*

Continues execution until the specified source line in the current routine is reached.

*routine line*

Continues execution until the specified source line in the specified routine is reached.

.

(period) Continues execution until the current source line in the current routine is reached again.

## Discussion

Except for the GO/NODEBUG form, all of the forms of GO automatically continue execution until either the specified condition or one of the following circumstances occurs:

- ▶ The program reaches a breakpoint or watchpoint.
- ▶ The program chains to another program.
- ▶ The program returns control to the monitor.

If you reach a breakpoint before the specified condition is met, the specified condition is cancelled.

GO without any arguments merely continues program execution.

## Examples

The example below continues program execution until the program enters the next function or subroutine.

```
GO/NEXT
```

## HELP – Provide command help information



HELP [*command*] [/PAGE]

### Arguments

*command*

(optional) Any valid debugger command or unique command abbreviation.

**/PAGE**

(optional) Stops the output every 24 lines and waits for input (CR to get the next page and <EOF> to terminate input). On Windows, the output will vary based on the values of DBG\_HEIGHT and DBG\_WIDTH.

### Discussion

The HELP command gives more detailed information about the specified debugger command. If no arguments are present, general help is displayed.

### Examples

The example below displays help information about the GO command.

DBG> HELP GO

The following appears:

Continue program execution:

```
GO          - Continue execution
GO /DEBUG   - Continue execution until entering routine compiled
              with debug
GO /NODEBUG  - Cancel debugging and continue execution
GO /NEXT     - Continue until the next function or subroutine entry
GO /EXIT     - Continue until the current function or subroutine
              exits
GO /RETURN   - Continue until a RETURN from the current CALL
GO /nnn      - Continue through nnn breaks, each break will be
              displayed
GO rtn       - Continue until the routine <rtn> is entered
GO ln        - Continue to line <ln> in the current routine
GO rtn ln    - Continue to line <ln> in routine <rtn>
GO .         - Continue until current line reached again
```

Note that, with the exception of GO/NODEBUG, all of the qualified forms will also suspend execution whenever a breakpoint or watchpoint is encountered.

## LIST – Display lines of source code

WT		U	V
----	--	---	---

LIST

LIST *line#*

LIST *line# count*

LIST/ALL

### Arguments

*line#*

Sets the current source line to the specified line number and displays 12 lines of code. *Line#* can either be the number of a source line or a period (.), which indicates the current line.

*line# count*

Sets the current source line to the specified line number and displays the number of lines of code specified by count. *Line#* can either be the number of a source line or a period (.), which indicates the current line.

/ALL

Displays all source lines within the current routine.

### Discussion

The LIST command displays lines of code beginning at the current source line. If you don't specify any arguments, LIST displays 12 lines.

The current source line is the line at which the debugger was entered. If you specify a line number rather than a period for the *line#* argument, the specified line becomes the current line. The current debug line is marked with a right angle bracket (>) in the display.

All nonprinting alpha field data is displayed as periods (.), except BS, HT, LF, VT, FF, and CR.

You cannot list any lines that come before the start of a routine or after the end of the routine. Therefore, the command

```
LIST 1
```

always lists from the beginning of the routine, even if the first line in the routine is line number 1255, for example. Likewise, the following command always lists the last line of the routine, even if 99999 is outside of the possible range of line numbers:

```
LIST 99999
```

Program routines that are .INCLUDEd display the line number in the form *source\_file# . line#*.

## Examples

The following example lists five lines beginning at the current line.

```
LIST . 5
```

## LOGGING – Log the debugging session to a file



LOGGING START[/APPEND] *filename*

LOGGING STOP

### Arguments

#### **/APPEND**

Appends the logging output to the end of the specified file.

#### *filename*

The name of the file to log to.

### Discussion

The LOGGING START command writes all debugger commands and command output to the specified log file. By default, a new log file is created unless the /APPEND option is present. If /APPEND is specified, the debugger opens the file in append mode and starts writing at the end of the file.

The LOGGING STOP command stops logging and closes the file. If the debugger is exited before this command is issued, an implied LOGGING STOP is performed to close the log file.

## OPENELB – Make an ELB’s subroutines available to debugger



OPENELB *elb\_spec*

### Arguments

*elb\_spec*

The file specification of the ELB to attach to.

### Discussion

The OPENELB command attaches to the specified ELB. The ELB’s subroutines are made available to the debugger and the executing program exactly as if the OPENELB system-supplied subroutine had been called prior to the line number of the break at which the command was entered. You can then set breakpoints in routines within that ELB.

Each OPENELB command opens the specified ELB and any ELBs that are linked to it. See [OPENELB](#) in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual* for restrictions on *elb\_spec*. Any *elb\_spec* that is valid for the OPENELB subroutine is valid for this command as well.

The OPENELB command can be used for client/server or regular debugging. When running the debugger in a client/server configuration with xfServerPlus, you will need to use OPENELB to open the ELBs containing your Synergy routines before setting a breakpoint in one of those routines.

If the specified ELB cannot be located or successfully attached, an error message is generated.

### Examples

The example below opens the ELB **WND:tklib.elb**.

```
OPENELB WND:tklib
```

## QUIT – Quit the current program without traceback

WT		U	V
----	--	---	---

QUIT

### Discussion

The QUIT command exits the current program without displaying any traceback information and returns to the operating system prompt. The program stops immediately; no FINALLY blocks or destructors are executed.



## SAVE – Save current debugger settings



SAVE *filename*

### Arguments

*filename*

The name of the file that will contain the debugger settings.

### Discussion

The SAVE command enables you to save the current debugger state to a file. The debugger state includes breakpoints, watchpoints, option settings, and ELB names. The name of each ELB that is opened via the OPENELB debugger command is written to the file before any other debugger commands.

Once you've saved your debugger settings to a file, you can specify the name of this file as the initialization file for the debugger, which enables you to associate a set of debugger commands with a project and invoke those commands every time you restart the debugging session.

If you don't specify a filename extension, the default extension is **.cmd** on Windows and UNIX or **.com** on OpenVMS. The saved file contains all debugger commands for the current setting state in the debugger, including the WATCH, BREAK, and SET commands.

You can restore the saved debugger commands by executing the *@filename* debugger command or setting the DBG\_INIT environment variable to the name of the file before invoking the debugger.

## SCREEN – Update the Synergy windowing system



SCREEN *option*

### Arguments

*option*

One or more of the following options:

<b>CLEAR</b>	Clear the screen.
<b>REDRAW</b>	Redraw the current windowing system screen.
<b>ROW</b> <i>row</i>	Reposition the cursor to the first column of the specified row.
<b>TOP</b>	Reposition the cursor to the first row on the screen.
<b>UPDATE</b>	Update a windowing system screen to its current state.
<b>WAIT</b>	Wait for a character to be typed before continuing.

### Discussion

If you are using the Synergy windowing API or UI Toolkit, the SCREEN command updates the screen and/or windowing system.

The SCREEN command serves as an interface to the windowing API. It enables you to establish small macros (using a combination of the SAVE and @ commands) so you can look at intermediate states of the screen that you wouldn't ordinarily get to see.

You can place more than one option, separated by spaces and/or commas, on the same line. Specifying WAIT as the last option gives you a chance to examine the current state of the screen before the next debug prompt covers it up.

### Examples

The following example redraws the windowing system screen, leaves the screen displayed until you enter a character, and then clears the screen before accepting any more debugger commands.

```
SCREEN REDRAW, WAIT, CLEAR
```

## SEARCH – Search the source for a string

WT		U	V
----	--	---	---

```
SEARCH string
SEARCH string line# count
SEARCH/ALL string
```

### Arguments

*string*

Searches from the current line for *string*.

*string line# count*

Searches *count* lines for *string*, starting with source line *line#*.

/ALL *string*

Searches all lines within the module for *string*.

### Discussion

The SEARCH command searches the current source module for the specified string.

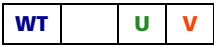
If *line#* is “>”, the current debug entry line is used. If *line#* is “.”, the current source line is used. Only lines within the current module are searched, so a *line#* of 1 searches from the module’s first line and a large *line#* searches the last line. After listing, the current source line is set to the next line to be listed. The current source line is set to the entry line on each debug entry.

### Examples

The following example searches the current module for the string “var1.” All source lines that contain “var1” are displayed.

```
SEARCH var1
```

## SET – Set debugger options



SET *option*

### Arguments

*option*

One of the following options:

<b>BREAK</b> <i>breakpoint</i>	See <a href="#">BREAK on page 2-13</a> .
<b>DBGSRC</b> <i>path</i>	Set the DBGSRC environment variable to the specified path.
<b>STEP OVER</b>	Set the default mode for the STEP command to step over a routine.
<b>STEP INTO</b>	Set the default mode for the STEP command to step into an external subroutine. (default)
<b>STOP ON</b>	Break when a STOP statement is executed (after any destructors have been executed).
<b>STOP OFF</b>	Do not break when a STOP statement is executed. (default)
<b>TRAP ON</b>	Break whenever a program traps an error.
<b>TRAP OFF</b>	Do not break when an error is trapped. (default)
<b>TYPEAHEAD ON</b>	Allow the debugger to use characters that are typed ahead in the application. (default)
<b>TYPEAHEAD OFF</b>	Prevent the debugger from using characters that are typed ahead in the application.
<b>UNINITIALIZED BREAK</b>	Set a program breakpoint when uninitialized stack records and ^M memory is accessed.
<b>UNINITIALIZED ON</b>	Turn on debugger messages when uninitialized stack records and ^M memory is accessed.
<b>UNINITIALIZED OFF</b>	Turn off uninitialized memory debugger messages and program breaking.
<b>VIEW</b> <i>count</i>	Set the default number of source lines that will be displayed immediately preceding and following the current line when the VIEW debugger command is specified without the optional <i>count</i> value. The default <i>count</i> for the VIEW command is 4.
<b>WATCH</b>	See <a href="#">WATCH on page 2-49</a> .

## Discussion

The SET command lets you customize debugger behavior by setting various debugger options.

The DBGSRV environment variable is appended to the beginning of source filenames before source files are opened in the debugger if the file cannot be found through the path to the file defined at compile time. It tells the debugger where the source file is located. You can either set DBGSRV in the environment, before you begin your debugging session, or you can set it with the SET command. If you set DBGSRV with the SET command, it overrides any DBGSRV path that you set previously at the environment level. If you are debugging remotely, make sure *path* is accessible from the server.

Detection for INITIALIZED ON or INITIALIZED BREAK occurs on assignment and IF tests.

STEP INTO only applies to external subroutines, not functions. (To step into a function, you need to either set a breakpoint or use the GO command to go to that function name.)



If the default step mode is STEP OVER, the debugger steps *into* any external subroutine that has a function as one of its arguments.

---

If you don't set TRAP ON, when an error is trapped, the program begins executing at the ONERROR label, without any warning that program control has changed. Setting TRAP ON causes the program to break when an error is trapped, and you get a message that tells you where the error occurred and what line will be executed next.

The SET UNINITIALIZED debugger feature is designed to help track down random problems at runtime due to variables in ^M and stack records not being initialized before use. Both %MEM\_PROC(DM\_ALLOC...) and stack records are random value unless preinitialized. When SET UNINITIALIZED debugger options are set, %MEM\_PROC memory is initialized to a series of 0xFA bytes, and stack memory is set to a series of 0xFB bytes. When an IF test or assignment statement is encountered, the data is scanned for a series of four or more FA or FB bytes, and the appropriate action is taken if found. It is possible for false positives to occur if integer fields that happen to have exactly the same values are used.

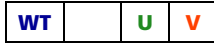
When the break occurs, the line reported is the next line that would be executed after the error statement, and this is the module where the uninitialized data is detected. In many cases, this is not the module defining the data, but the module using the data, which was most likely passed in via arguments. For example, many times a UI Toolkit list processing routine uses stack data passed in several levels up the call stack via the user data argument. However, this is a bug in the user code, not a bug in Toolkit.

## Examples

The example below sets the default STEP mode to STEP OVER. If no arguments are specified on the STEP command, STEP steps over a routine.

```
SET STEP OVER
```

## SHOW – Examine debugger options and program state information



SHOW [/PAGE] [option, ...]

### Arguments

#### /PAGE

(optional) Stops the output every 24 lines and waits for input (CR to get the next page and <EOF> to terminate input). On Windows, the output will vary based on the values of DBG\_HEIGHT and DBG\_WIDTH. (/PAGE can alternatively be placed at the end of the line.)

#### option

(optional) One or more of the following options, separated by a comma and/or a space:

<b>BREAK</b>	Display all current breakpoints.
<b>CHANNELS [/FULL/]</b>	Display all Synergy DBL channels that are currently open, along with their open mode, submode, and filename. If /FULL is specified, the filename is the full Windows filename, including physical path. If no channels are opened, a message to that effect is displayed.
<b>CLASSES [opts] [name] [...]</b>	Display information about the currently instantiated classes. If specified, CLASSES must be the last option on the command line. See <a href="#">“Showing instantiated classes” on page 2-44</a> for more detailed information.
<b>DBGSRC</b>	Display the path to which the DBGSRC environment variable is currently set.
<b>DLL</b>	Display all open DLL handles and the complete filename of the associated DLL, in the order in which they were opened. The same DLL is listed multiple times if it was opened multiple times. (Windows only)
<b>DYNMEM</b>	Display all dynamic memory segments that are currently in use.
<b>ELB</b>	Display all ELBs that are currently open.
<b>ERROR</b>	Display the error that caused the current error trap.
<b>MEMORY</b>	Display current Synergy DBL memory usage as well as the number of segment reclamations that occurred during program execution.

<b>OPTIONS</b>	Display Synergy compiler/runtime options and flags.
<b>STACK</b>	Display the current Synergy DBL stack parameters: the size of the stack, how much of the stack is currently in use, and the maximum number of bytes that have been used.
<b>STEP</b>	Display the current mode for the STEP command (STEP OVER or STEP INTO).
<b>STOP</b>	Display the current STOP mode (STOP ON or STOP OFF) as set by the SET command.
<b>TRACE</b>	Display the current CALL or XCALL traceback. (This option displays the same information as the TRACE command.)
<b>TRAP</b>	Display the current error trap mode (TRAP ON or TRAP OFF) as set by the SET command.
<b>UNINITIALIZED</b>	Display the current UNINITIALIZED state as set by the SET command.
<b>VARIABLE</b> <i>var_list</i>	Display the type and size of one or more variables. If present, VARIABLE must be the last keyword on the SHOW command line, followed by one or more variable names separated by spaces and/or commas.
<b>WATCH</b>	Display any watchpoints.

## Discussion

The SHOW command displays the current values for the debugger options set with the SET command, as well as a variety of program state information.

If you don't specify any options, the SHOW command displays all of the current debugger options and program state information. You can place more than one option, separated by spaces and/or commas, on the same line.

If the amount of output is large, you can use the /PAGE option to page the output.



If you want to use the SHOW DYNMEM command and you are using UI Toolkit, we recommend that you use SHOW DYNMEM after calling the U\_FINISH routine, since Toolkit also uses dynamic memory.

---

#### Showing instantiated classes

The CLASSES option has the following syntax:

```
CLASSES [opts] [name] [...]
```

*opts*

(optional) One or more of the following class options:

<b>/ALL</b>	Display class information for all classes that have (or at one time had) instantiation.
<b>/OBJECTS</b>	Display object information.
<b>/LINES</b>	Display the first referencing source line or creation line.
<b>/WARNINGS</b>	Display only objects with circular references.

*name*

(optional) A class name path.

If no class names are specified, all classes that have current instantiations will be included. This information will include the name and the number of instantiated objects for each class.

If CLASSES is specified in conjunction with other SHOW options, CLASSES must be the last option specified on the line. If CLASSES is not the last option on the line, any option that follows CLASSES will be interpreted as a *name*.

You can specify more than one class by separating them (along with their class options) with commas or spaces.

#### Examples

The following example shows the current breakpoints, STEP mode, error trap mode, and Synergy DBL stack parameters.

```
SHOW BREAK, STEP, TRAP, STACK
```

The example below displays information about each class and the number of instantiated objects for each class.

```
DBG> show classes /all
```

```
Class cmdtest.b : 0 instances
Class cmdtest.d : 1 instance
Class cmdtest.e : 1 instance
Class cmdtest.loon : 1 instance
```



The following example displays the object ID, the number of references, and any circular references.

```
DBG> show classes /obj cmdtest.d
```

```
Class cmdtest.d : 1 instance  
  Object id 1, refs 2 (circular!)
```

The example below displays the creation line for the object.

```
DBG> show classes /lines cmdtest.d
```

```
Class cmdtest.d : 1 instance : 1st ref at line 92 in TD2 (objcmds.dbl)
```

The example below indicates that a circular reference exists.

```
DBG> show classes /warn cmdtest.d
```

```
Class cmdtest.d : 1 instance  
  Object id 1, refs 2 (circular!)
```

## STEP – Step to the next Synergy DBL statement



STEP *[count]*

STEP *option*

### Arguments

*count*

(optional) Steps through *count* statements in the current step mode.

*option*

One of the following options:

**INTO**                      Step into, or enter, a routine.

**OVER**                      Step over, or skip, a routine.

### Discussion

The STEP command steps to the next Synergy DBL statement.

If no arguments are specified, STEP steps in the current step mode, as set by the SET command.

You can check the current step mode with the SHOW command.

You can only STEP into an XCALL, not a function; to step into a function, use the GO/NEXT command.

### Examples

The following example steps to the next Synergy DBL statement. If the next statement is an XCALL statement, the debugger either steps over or into the routine, depending on the mode that was set by the SET command. If no STEP mode was set, this command will STEP INTO the routine by default.

STEP

## TRACE – Display the current traceback

WT		U	V
----	--	---	---

TRACE

### Discussion

The TRACE command displays the CALL, XCALL, and function traceback to the line at which the debugger currently has control.

The TRACE command is equivalent to the SHOW TRACE command.

## VIEW – Display lines around a debug entry

WT		U	V
----	--	---	---

VIEW *[count]*

### Arguments

*count*

(optional) The number of lines to display around the current breakpoint.

### Discussion

The VIEW command displays the current debug entry and the lines that immediately precede and follow it.

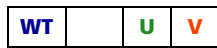
If you don't specify any arguments, the four lines that precede the current debug line, the current debug line, and the four lines that follow it are displayed.

### Examples

The following example displays the six lines that precede the current debug line, the current debug line, and the six lines that follow the current debug line.

```
VIEW 6
```

## WATCH – Set a watchpoint



```
WATCH variable [option]
WATCH variable rel_operator value
WATCH address [option]
```

### Arguments

#### *variable*

Sets a watchpoint on the specified variable. See “[Specifying variables](#)” on page 2-5 for more information about variable specifications. *Variable* can also be a `^M(struct_fld, handle)` specification.

#### *option*

(optional) The type and size of the variable or address to watch. It must be one of the following:

**/An**      Alpha of size *n*

**/In**      Integer of size *n*

#### *rel\_operator*

One of the following relational operators:

**.GT.**      Greater than

**.LT.**      Less than

**.GE.**      Greater than or equal to

**.LE.**      Less than or equal to

**.EQ.**      Equal to

**.NE.**      Not equal to

#### *value*

The numeric value to which *variable* will be compared.

#### *address*

Sets an **i4** watchpoint at the specified address.

### Discussion

The WATCH command sets a watchpoint on the specified variable or address. This means that when the contents of the variable have changed or the expression *variable rel\_operator value* is true, the debugger is invoked and the contents are displayed. If you watch a variable that is longer

than one line (80 characters), the variable's contents are displayed on multiple lines. Any nonprinting characters are displayed as periods (.). The debugger breaks on the line that changes the variable.

The maximum number of watchpoints is 32.

*Variable* is a variable specification identical to data division variable specifications. Any variable or address being watched can be cast by the *option* qualifier. For example, you can watch the center three bytes of a **d5** variable as alpha using the following command:

```
SET WATCH dvar(2:3)/a3
```

You can also watch two successive records by spanning across both. For example, given the following data division:

```
record a
    avar1      ,a10
    avar2      ,a10

record b
    bvar1      ,d10
    bvar2      ,d10
```

you could specify the following SET WATCH commands:

```
SET WATCH a/a40
```

or

```
SET WATCH a    and    SET WATCH b/d20
```

The relational operators below can be used to watch a variable in relation to another value or in relation to another variable. In the second case, the value of the second variable will be saved at the time the watchpoint is set, and the value of the first variable will be compared against that value. A relational watchpoint will be deleted after the watchpoint has triggered.

.GT. or >  
.LT. or <  
.GE. or >=  
.LE. or <=  
.EQ. or ==  
.NE. or !=

For example, the following breaks when **var1** becomes less than the value of **var2** (that is, the value of **var2** when the watchpoint was set):

```
WATCH var1 .LT. var2
```

## Watching object handles

A simple, nonrelational watchpoint on a handle saves the current handle reference and breaks when that reference changes. For relational watchpoints on handles, only .EQ. and .NE. are permitted. These are interpreted as “Do these handles now reference the same object?” and “Do these handles now reference different objects?” respectively. Handles can be compared against each other or against the value ^NULL, which indicates that no object is referenced.



Custom implementation of the op\_Equality or op\_Inequality operator methods will not be recognized by the debugger.

---

## Watching string objects

Watchpoints on string objects behave the same as watchpoints on alpha variables, with the additional option of comparing the string handle to ^NULL using the .EQ. or .NE. operator.

## Examples

The following example breaks when the first four characters of the variable **ab** have changed.

```
WATCH ab
```

The next example breaks when the contents of **x** are greater than or equal to 0.

```
WATCH x .ge. 0
```

The example below breaks when the contents of the alpha variable **id** equals “A327”.

```
WATCH id/A4 .eq. "A327"
```

The following example breaks when the contents of the class field **myfld** is less than 12.

```
WATCH myclass.myfld .lt. 12
```

The example below breaks when the object handle **objhnd** changes to reference a different object or changes to or from ^NULL.

```
WATCH objhnd
```

This example breaks when **objhnd1** no longer references the same object as **objhnd2**.

```
WATCH objhnd1 .NE. objhnd2
```

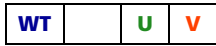
This example breaks when the string handle becomes ^NULL.

```
WATCH stringvar .EQ. ^NULL
```

This example breaks when the content of the string variable exceeds “ABC”.

```
WATCH stringvar .GT. "ABC"
```

## WINDBG – Invoke the UI Toolkit debugger



WINDBG

### Discussion

The WINDBG command invokes the UI Toolkit window debugger.

The WINDBG command is not supported by the debug server if you are debugging remotely.



## @ – Process an indirect command file



*@ filename*

### Arguments

*filename*

The file from which to process debugger commands.

### Discussion

When you specify a command file, a new command file level is activated until the last line in the file is executed. If one of the lines executed is another indirect command file specification, another level is activated until the lines in that file are executed. Up to eight levels can be activated within the debugger.

The default filename extension for a command file is **.cmd** (**.com** on OpenVMS), and full Synergy DBL-style logical name translation occurs.

## ! – Execute system commands



`! [command]`

### Arguments

*command*

(optional) The command in a shell to be executed.

### Discussion

The `!` command takes you to the operating system prompt, which enables you to execute a system command without exiting the debugger.

If no arguments are specified, `!` places you at the operating system's command level prompt. On UNIX, this means executing a UNIX shell without executing any commands. On OpenVMS, `!` spawns a subprocess.

The `!` command is not supported by the debug server if you are debugging remotely.

### Examples

For example, in a UNIX environment whose shell prompt is a `$`, the following sequence of commands exits to a shell, renames **a.b** to **x.y**, deletes **test**, and returns to the debugger:

```
DBG> !  
$ mv a.b x.y  
$ rm test  
$ exit
```

## Sample Debugging Session

We've used a Synergy program called **badship.dbl** in our sample debugging session. **Badship.dbl** creates a report of bad shipments with the specified starting and ending dates. The user can also specify the customer's ID number and the output device that should be used to report the results.

We know that the **badship.dbl** program has three problems:

- ▶ When a customer's ID number is specified, **badship.dbl** doesn't find anything, even if it should.
- ▶ The starting date for the search doesn't work correctly.
- ▶ The "No bad records..." message is not sent to the screen even if no bad records exist.

*First, we'll compile, link, and run the program with the debug option. (Note that these commands do not apply to OpenVMS systems. See [chapter 1](#), "Building and Running Synergy Applications," for the correct compile, link, and run commands for OpenVMS.)*

```
$ dbl -d badship
$ dblink -d badship
$ dbr -d badship
*** DEBUG 8.3.1 ***
Break at 121 in BADSHIP (badship.dbl)
*121:          xcall flags(7004020, 1)
```

*Now that we're in the debugger, we'll use the **HELP** command to list all available debugger commands.*

```
DBG> help
```

Help is available on:

```
BREAK      - Set a program breakpoint
CANCEL     - Cancel program breakpoints and watchpoints
DEPOSIT    - Modify variables
DELETE     - Delete program breakpoints
EXAMINE    - Examine variables
GO         - Continue program execution
LIST       - List source lines
OPENELB    - Open the specified ELB or shared image
SAVE       - Save the current debugger context
SCREEN     - Do screen manipulation
SEARCH     - Search source
SET        - Set debug options
SHOW       - Display debug options and program state information
STEP      - Step to the next Synergy DBL instruction
TRACE     - Display current traceback
VIEW      - View Synergy DBL source around the debug entry
WATCH     - Set a watchpoint
WINDBG    - Invoke the Toolkit window debugger
```

## Debugging Synergy Programs

### Sample Debugging Session

```
Cmd_Inp  - Command recall and editing
@        - Process an indirect command file
!        - Execute a shell command
```

*We'll view the source lines that surround the debug entry and then set the default mode for the STEP command, set the debugger to break whenever the program traps an error, and examine all debugger parameters.*

```
DBG> view
117:                                     ,a2
118:          rep                      ,a5
119:
120: proc
121>    xcall flags(7004020, 1)
122:    open(TT_CH, i, "tt:")
123:    display(TT_CH, $scr_clr(SCREEN), "Create bad
shiplist.",10,13)
124:    do
125:    begin
DBG> set step into
DBG> set trap on
DBG> show
```

The debugger wasn't entered in an error state.

No breakpoints are set.

Default step mode is "INTO"

%DBR-I-ATLINE, at line 120 in routine MAIN\$BADSHIP (badship.dbl)

A trapped DBL error will cause a break.

DBL stack size 4096 bytes; now using 32; maximum used 0.

No DBL channels opened.

DBGSRC not set

*Now we'll list 140 lines beginning at line 120 so we can determine possible problem spots.*

```
DBG> list 120 140
120: proc
121>    xcall flags(7004020, 1)
122:    open(TT_CH, i, "tt:")
123:    display(TT_CH, $scr_clr(SCREEN), "Create bad ship list.",10,13)
124:    do
125:    begin
```

```
126:         display(TT_CH, $scr_pos(2,0),"For a specific customer? ")
127:         reads(TT_CH, ans)
128:     end
129: until ((ans.eq."y").or.(ans.eq."n"))
130:     if ((ans.eq."Y").or.(ans.eq."y"))
131:         begin
132:             display(TT_CH, $scr_pos(3,0),"Enter Customer ID: ")
133:             reads(TT_CH, cust_id)
134:             cmp_id(1,4) = cust_id
135:             do
136:                 begin
137:                     display(TT_CH, $scr_pos(4,0),"Want a total history?")
138:                     reads(TT_CH, history)
139:                 end
140:                 until ((history.eq."y").or.(history.eq."n"))
141:             end
142:             if ((history.eq."Y").or.(history.eq."y")) then
143:                 begin
144:                     predate = 19760101
145:                     postdate = 30000000
146:                 end
147:             else
148:                 begin
149:                     do
150:                         begin
151: first,             display(TT_CH, $scr_pos(5,0), "Start: (MM/DD/YYYY)")
152:                         reads(TT_CH, entdate)
153:                     end
154:                     until (%rsize.eq.10)
155:                     dday=entday
156:                     dmon=entmonth
157:                     dyr=dyr
158:                     predate=date
159:                     do
160:                         begin
161:                             display(TT_CH, $scr_pos(6,0), "End: (MM/DD/YYYY)")
162:                             reads(TT_CH, entdate)
163:                         end
164:                         until (%rsize.eq.10)
165:                         dday=entday
166:                         dmon=entmonth
167:                         dyr=entyear
168:                         postdate=date
169:                     end
170:                 end
171:             do
172:                 begin
173:                 display(TT_CH, $scr_pos(7,0),
```

## Debugging Synergy Programs

### Sample Debugging Session

```
174:  &          "Print the report to the (S)creen or (F)ile? "
175:      reads(TT_CH, out_flg)
176:  end
177:  until ((out_flg.eq."s").or.(out_flg.eq."f"))
178:  if ((out_flg.eq."F").or.(out_flg.eq."f"))
179:      begin
180:          do
181:              begin
182:                  display(TT_CH, $scr_pos(8,0),
183:  &          "Should output be sent to the printer? (Y/N)")
184:                  reads(TT_CH, printer_flg)
185:              end
186:              until ((printer_flg.eq."y").or.(printer_flg.eq."n"))
187:          end
188:          if (printer_flg.eq."n")
189:              begin
190:                  display(TT_CH, $scr_pos(9,0),
191:  &          "Output will be placed in file: ", BADFILE,10,13)
192:              end
193:              display(TT_CH, "Press return to continue or ^D to exit",10,13)
194:              reads(TT_CH, incode) [eof=exit]
195:              if (incode.eq.' ')
196:                  incode = "BAD"
197:              open(CLNT_CH, i:i, "sclnt")
198:              open(FACT_CH, i:i, "sfact")
199:              open(FOLD_CH, i:i, "sfhdr")
200:              open(OUT_CH, o, BADFILE)
201:              if ((out_flg.eq."S").or.(out_flg.eq."s")) then
202:                  begin
203:                      date=predate
204:                      display(TT_CH, "BAD Ship list for ",dmon,"/",dday," ",dyr)
205:                      date=postdate
206:                      display(TT_CH, " - ",dmon,"/",dday,"/",dyr)
207:                      forms(TT_CH, 2)
208:                  end
209:              else
210:                  begin
211:                      date=predate
212:                      display(OUT_CH,"BAD Ship list for ",dmon,"/",dday,"/",dyr)
213:                      date=postdate
214:                      display(OUT_CH, " - ",dmon,"/",dday,"/",dyr)
215:                      forms(OUT_CH, 2)
216:                  end
217:              isam_pre=predate
218:              read(FACT_CH, cmfact, isam_pre, KRF=2) [err=next]
219:              next, while (cadate.ge.predate) .and. (cadate.le.postdate) do
220:                  begin
221:                      if ((caactc.eq.incode(1,%trim(incode))).and.(ans.eq."n"))
```

```

222:          begin
223:             chcomp = cacomp
224:             chclnt = cacnt
225:             chfldr = cafldr
226:             read(FOLD_CH, cmfhdr, chkey) [err=skip]
227:             if (chftyp.ne."WISH" .and. chclos.eq.0)
228:                call dumpit
229:             end
230:             if ((caactc.eq.incode(1,%trim(incode))).and.
231: &             (ans.eq."Y").and.(cmp_id.eq.cacnt))
232:                begin
233:                   chcomp = cacomp
234:                   chclnt = cacnt
235:                   chfldr = cafldr
236:                   read(FOLD_CH, cmfhdr, chkey) [err=skip]
237:                   if (chftyp.ne."WISH" .and. chclos.eq.0)      ;Is it open?
238:                      call dumpit
239:                end
240: skip,
241:       reads(FACT_CH, cmfact) [eof=done]
242:       end
243: done, if (.not.find_flg)
244:       begin
245:          if (out_flg.eq."S") then
246:             writes(TT_CH, "No bad records were located in query.")
247:          else
248:             writes(OUT_CH, "No bad records were located in query.")
249:          end
250:       close CLNT_CH
251:       close FACT_CH
252:       close TT_CH
253:       close OUT_CH
254:       if ((printer_flg.eq."Y").or.(printer_flg.eq."y"))
255:          begin
256:             lpque(BADFILE, LPNUM:"conan")
257:             xcall delet(OUT_CH, BADFILE)
258:          end
259: exit, stop

```

From the listing, we can see that **cmp\_id** in line 134 is loaded with **cust\_id**. Since **cmp\_id** is used for the search, we'll want to see if it is loaded correctly in line 134. The second problem in the program involves the starting date. From the listing, we can see that **predat** is loaded at line 158. The third problem involves the "No bad records..." message, which is located around line 243.

DBG> trace

```
%DBR-I-ATLINE, at line 120 in routine MAIN$BADSHIP (badship.dbl)
```

## Debugging Synergy Programs

### Sample Debugging Session

*We'll use the **HELP** command to see what our **BREAK** options are.*

```
DBG> help break
```

Set program breakpoints:

```
BREAK rtn      - Set a break at the entry to routine <rtn>
BREAK ln       - Set a break at line <ln> in the current routine
BREAK rtn ln   - Set a break at line <ln> in routine <rtn>
BREAK lbl [/LABEL]
                - Set a break at <lbl> in the current routine
BREAK .        - Set a break at the current line and routine
```

Multiple breaks may be specified, separated by commas, with the "current routine" the last <rtn> encountered.

```
DBG> break 132
```

```
DBG> show breaks
```

```
MAIN$BADSHIP: 132
```

```
DBG> go
```

Generate bad shipment list.

Generate for a specific customer? y

Break at 132 in BADSHIP (badship.dbl)

```
132>          display(TT_CH, $scr_pos(3,0), "Enter Customer ID: ")
```

```
DBG> step
```

Enter Customer ID: Step to 133 in BADSHIP (badship.dbl)

```
133>          reads(TT_CH, cust_id)
```

```
DBG> step over
```

```
3040
```

Step to 134 in BADSHIP (badship.dbl)

```
134>          cmp_id(1,4) = cust_id
```

```
DBG> examine cust_id
```

```
3040
```

```
DBG> examine cmp_id
```

```
000000
```

```
DBG> step
```

Step to 3 in BADSHIP (badship.dbl)

```
137>          display(TT_CH, $scr_pos(4,0), "Want a total history?")
```

```
DBG> examine cmp_id
```

```
304000
```

*Here we learn that **cmp\_id** isn't loaded correctly: **cmp\_id(1,4)=cust\_id** should be changed to **cmp\_id(3,6)=cust\_id**.*



*Now we'll jump to the next possible problem area.*

```
DBG> go 150
Want a total history? n
Break at 151 in BADSHIP (badship.dbl)
151> first,      display(TT_CH, $scr_pos(5,0), "Start:  (MM/DD/YYYY) ")
```

*We'll do a TRACE command to see where the last command was executed.*

```
DBG> trace

%DBR-I-ATLINE, at line 147 in routine MAIN$BADSHIP

DBG> step
Start:  (MM/DD/YYYY) Step to 152 in BADSHIP (badship.dbl)
152>      reads(TT_CH, entdate)
DBG> step 5
05/22/1986
step to 154 in BADSHIP (badship.dbl)
154>      until (%rsize.eq.10)
DBG> view
150:      begin
151: first,      display(TT_CH, $scr_pos(5,0), "Start:  (MM/DD/YYYY) ")
152:      reads(TT_CH, entdate)
153:      end
154>      until (%rsize.eq.10)
155:      dday=entday
156:      dmon=entmonth
157:      dyr=dyr
158:      predate=date
DBG> step
Step to 155 in BADSHIP (badship.dbl)
155>      dday=entday
DBG> step
Step to 156 in BADSHIP (badship.dbl)
156>      dmon=entmonth
DBG> examine dday
22
DBG> step
Step to 157 in BADSHIP (badship.dbl)
157>      dyr=dyr
DBG> examine dmon
05
DBG> step
Step to 158 in BADSHIP (badship.dbl)
158>      predate=date
DBG> examine dyr
```

## Debugging Synergy Programs

### Sample Debugging Session

*Dyr should have contained 1986, not a blank. Instead of **dyr=dyr**, line 157 should be **dyr=entyear**.*

```
DBG> step
Step to 161 in BADSHIP (badship.dbl)
    161>          display(TT_CH, $scr_pos(6,0), "End:  (MM/DD/YYYY) ")
DBG> examine predate
522
```

*This **predate** value of 522 verifies the above problem. **Predate** should have contained 19860522.*

```
DBG> go 244
End:  (MM/DD/YYYY) 01/01/1992
Print the report to the (S)creen or (F)ile? s
Press return to continue or ^D to exit
```

```
BAD Ship list for 5/22/          - 01/01/1992
```

```
DBL error trapped at 218 in BADSHIP (badship.dbl), jumping to line 219
    218>          read(FACT_CH, cmfact, isam_pre, KRF=2)          [err=next]
```

*The error we expected was trapped.*

```
DBG> go 244
DBL error trapped at 241 in BADSHIP (badship.dbl), jumping to line 243
```

*Again, the expected error was trapped.*

```
    241>          reads(FACT_CH, cmfact) [eof=done]
DBG> go 244
break at 245 in BADSHIP (badship.dbl)
    *245:          if (out_flg.eq."S") then
DBG> view
    241:          reads(FACT_CH, cmfact) [eof=done]
    242:          end
    243: done, if (.not.find_flg)
    244:          begin
    245:          if (out_flg.eq."S") then
    246:              writes(TT_CH, "No bad records were located in query.")
    247:          else
    248:              writes(OUT_CH, "No bad records were located in query.")
    249:          end
DBG> examine out_flg
```

*The IF statement checks for uppercase "S" above, but because **out\_flg** is lowercase "s," it never allows output to be sent to the console. To check this theory, we'll put uppercase "S" into **out\_flg**.*

```
DBG> deposit out_flg = 'S'
DBG> examine out_flg
S
DBG> step
```

```
Step to 246 in BADSHIP (badship.dbl)
246>          writes(TT_CH "No bad records were located in query.")
```

*This fixed the problem, which means that the IF statement in line 245 should be changed to check for a lowercase "s."*

```
DBG> step
No bad records were located in query.
Step to 250 in BADSHIP (badship.dbl)
250>          close CLNT_CH
DBG> st
```

*Notice that you can abbreviate the STEP command.*

```
Step to 251 in BADSHIP (badship.dbl)
251>          close FACT_CH
DBG> s
Step to 252 in BADSHIP (badship.dbl)
252>          close TT_CH
DBG> go
```

```
%DBR-S-STPMMSG, STOP
%DBR-I-ATLINE, at line 259 in routine BADSHIP (badship.dbl)
$
```



# 3

## Synergy DBMS

Synergy DBMS is the file management system for Synergy/DE.

### **Synergy File Types 3-2**

Describes the four types of Synergy database files: Synergy ISAM (or RMS on OpenVMS), relative, sequential, and stream. Each section discusses file structure and concepts, file types, record access, and the I/O statements and system-supplied routines that enable you to manipulate each type of file. The ISAM section also describes keys and how they are used in ISAM files and discusses change tracking, file and data corruption, and the options and strategies for corruption recovery.

### **Synergy DBMS Utilities 3-34**

Describes how to use the following Synergy DBMS utility programs:

bldism – Create an ISAM file .....	3-38
chklock – Report file lock information.....	3-48
ctutl – Manipulate change tracking parameters .....	3-50
fcompare – Compare database files to system catalog or repository.....	3-56
fconvert – Convert database files to other file types .....	3-62
ipar – Generate parameter file descriptions .....	3-68
irecovr – Recover Revision 4 or higher ISAM files .....	3-71
isload – Load, unload, or clear an ISAM file .....	3-72
ismvfy – Verify structure of a Revision 4 or higher ISAM file.....	3-76
isutl – Verify, recover, and optimize Revision 4 and higher ISAM files .....	3-77
status – Report the status of an ISAM file.....	3-87

### **ISAM Definition Language 3-89**

Defines all keywords in the ISAM definition language (XDL), specifies the syntax for an XDL file, and describes the **xdlchk** utility.

### **Moving Database Files to Other Systems 3-101**

Describes how to move your database files to other platforms.

# Synergy File Types

## Synergy ISAM files

---

### VMS

---

We use RMS ISAM on OpenVMS for native compatibility. All other systems use Synergy ISAM. See [Chapter 3](#) of your *Professional Series Portability Guide* for information about Synergy ISAM features that are not compatible with RMS ISAM. See the *OpenVMS Record Management Services Reference Manual* for information on how to use RMS ISAM.

---

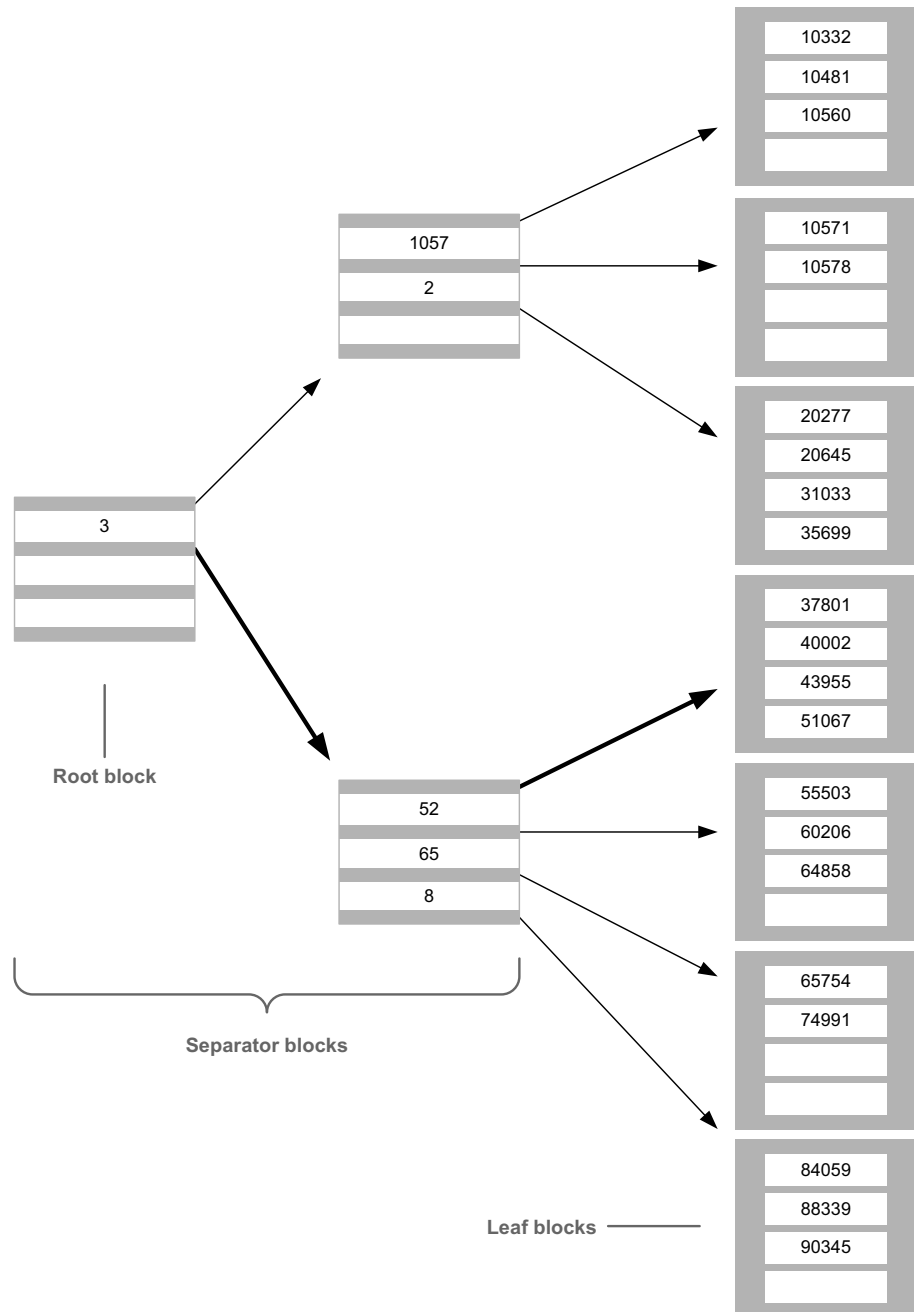
A Synergy ISAM file is used for high-speed, keyed access and ordered sequential access. As your file grows, high-speed, keyed access is maintained. Your ISAM file's size can grow to fit the need of your application, given the physical limitations of your disk. For example, we can have an ISAM file that contains a record for each of our customers. A record in an ISAM file consists of a set of fields. Each field stores a specific item of data, such as a customer number, a first and last name, a company name, a street address, a city, a ZIP Code, or a telephone number.

If we wanted to find a particular customer in a non-ISAM database file, such as customer number 125 or customer B. Jones, our program might have to search the entire file. Synergy ISAM, however, provides an access method that uses an index. An index enables you to quickly find specific records in a database file without having to search the entire file and without your program having to look at extra records. This is called keyed access. It also enables you to define an order for the sequential processing of a database file. This is called sequential access.

Each index in an ISAM file is defined by a key. A key is one or more fields or portions of fields from a record that are used to locate that record. Keys are defined when the ISAM file is created. For example, we can define a key for our customer number field that places customer numbers in ascending or descending order within the corresponding index.

An ISAM file's index contains leaf entries, which are sequentially ordered key values that point to corresponding data records. For example, the index defined by our customer number key, as illustrated in [figure 3-1](#), contains the customer numbers (key values) in descending order for all customers in an ISAM file. These customer number entries point to data records. The data records are in no particular order. The index, however, is always in a specific order as defined by the key. In this case, the index is in descending order by customer number.

ISAM indexes are also structured hierarchically so that access to any particular record occurs with a minimum of index reading. Access to any data record by a particular key requires the same amount of index reads, provided the index doesn't change. This number is determined by the index depth. Keyed access requires one index READ for each level of the index. The number of levels required for an index is universally proportional to key length. Large ISAM files (one million or more records) with long keys (45 or more characters) have five or more levels of index depth.



*Figure 3-1. Diagram of an ISAM file's index.*

As shown in [figure 3-1](#), an ISAM file's index is composed of blocks. A block is the smallest unit of an index that can be read or written at one time, and block size is set when you create an ISAM file. (Note that to simplify the diagram, the block size in the diagram is artificially small.) The index in the diagram is three levels deep. In this case, separator blocks make up the first two levels of the index. (Any ISAM file with more than one index block has separator blocks.) The third or last level is made up of leaf blocks, which contain sorted key values (in this case, customer numbers) with a one-to-one correspondence between each leaf entry and the data record to which it points. The separator blocks exist as pathways to the leaf entries and are composed of pointers to lower-level index blocks and separator values that narrow the range of key values. The root block is a special separator block that is read first on any keyed access.

For example, let's assume we want to access the data record for customer number 40002 by our customer number key using the following statement:

```
read(ch, rec, "40002")
```

First the root block is read. Synergy ISAM will determine that the number of our customer number (4) is greater than the value 3. We therefore read the separator block indicated by the first bold pointer in the diagram. Synergy ISAM will then determine that the first two numbers of our customer number (40) are less than the value 52 in the separator block. We therefore read the next block indicated by the second bold pointer in the diagram. Synergy ISAM will then find the entry 40002 within this block, which reads the data record for customer number 40002. By structuring the index hierarchically, Synergy ISAM enables us to access data records without having to read through the entire index. Also notice that reading any record by key value requires the same number of index reads.

An ISAM file can have more than one index. For example, we can define an index for the customer number field of our ISAM file and another index for the customer name field. These indexes enable us to quickly access the record for customer number 125, or the record for customer B. Jones.

The keys of an ISAM file also define the sequential order in which the file may be processed. For example, we can access our ISAM file sequentially by the customer number key, alphabetically by the customer name key, geographically by the customer city key, or by any other key that we define.

You can create an ISAM file using the ISAMC subroutine, the OPEN statement, or the **bldism** utility. See [ISAMC](#) in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual*, [OPEN](#) in the “Synergy DBL Statements” chapter of the *Synergy DBL Language Reference Manual*, or [bldism on page 3-38](#) for details.



## Revision levels and file compatibility

ISAM Revision 6 is the default revision level for all ISAM files being created, unless 6 is overridden by the ISAMC\_REV environment variable. (ISAMC\_REV enables you to create ISAM files compatible with previous versions of Synergy or to convert ISAM files to a higher revision level. See [ISAMC\\_REV](#) in the “Environment Variables” chapter of *Environment Variables & System Options* for more information.)

Starting with Revision 6, ISAM files also have a *compat level*, which enables them to change their file structure without requiring an entire ISAM revision change. The compat level begins with the revision number, followed by a period and another digit that is incremented by 1 each time new ISAM features are added.

Current compat levels and their features are as follows:

Compat level	Release version	ISAM features
6.0	10.1	—
6.1	Non-Synergy release	—
6.2	10.3	Support of records greater than 64K CTIMESTAMP key SIZE_LIMIT option RECORD_LIMIT option

All REV6 files created without any of the above listed features have the initial 6.0 compat level. Using one or more of the features above gives a file the specified compat level.

As an example, let’s say you install Synergy/DE version 10.1. If you create files with any ISAM options available through version 10.1, the compat level for these files is 6.0. You can access these files with version 10, but because the revision level is REV6, you can’t access them with earlier Synergy versions (9.5.3, 8.3.1, etc.), which had a lower revision level.

If you upgrade one of your systems to 10.3, both versions 10.1 and 10.3 can access the files you created with 10.1, because version 10.3 can access prior revisions (4 and 6) and compat levels. If you create new files with 10.3 and continue to use the 10.1 compatible options, both 10.1 and 10.3 can still access the files, and the compat level is 6.0. However, if you create more new files with 10.3 and this time use new 10.3 options such as the CTIMESTAMP autokey, the compat level for these files will be 6.2. If version 10.1 attempts to open the file now, it can’t, because 10.1 can only access a compat level of 6.0.

To view the compat level of a file, use the **ipar** utility. Below is some sample output of a file created with 10.3 with new options. Note that it says “revision 6” and then below that, “Creation version 10.2.5c {Compat Level 6.2}.”

```
;      Synergy ISAM PAR File created Fri Oct 17 15:14:10 2014
test
50      ;Record size
1      ;Number of keys
        ;5ca5 magic, revision 6, 1 byte record overhead
        ;Static vectoring disabled
        ;Default R6 file attributes:
        ;  Static RFAs enabled (Required)
        ;  Page size 4K
        ;Creation version 10.2.5c {Compat Level 6.2}
        ;0 of 32 byte file text allocation in use, @0x1694
        ;Update revision count 0
        ;File created on Fri Oct 17 15:14:05 2014
        ;8 byte longest key
        ;0 free index blocks, 0x0 free list head
        ;0 records, 0 free

/type
        ;Primary key
C      ;Create timestamp autokey type
8      ;Key size
1      ;Start position
N      ;Duplicates allowed
A      ;Ascending/descending
        ;Root 0x2000, index depth 1
        ;Minimum keys per block 170
```

## Large sector drives

Revision 6 ISAM files target the performance of Advanced Format large sector drives by reading and writing index blocks on 4K boundaries. The page size defaults to 4K, or 4096 bytes. (See “[Page size](#)” on [page 3-9](#) for more information.) If you want a smaller page size, you must explicitly specify it when the file is created.



You must use Revision 6 ISAM with large sector drives if you want them to operate efficiently.

---

## ISAM file structure

The physical representation of a Synergy ISAM file reflects two files: one contains the data records and the other contains the indexes that point to the data. The index file has the default extension **.ism** (for example **customer.ism**). The data file has the extension **.is1** (for example, **customer.is1**). The two files are always referenced together as one ISAM file with the extension **.ism**.

If you specify an ISAM filename with an extension other than **.ism** (for example, **customer.dat**), the last character of the data file's extension is replaced with a **1** (for example, **customer.da1**). However, if the last character of the specified filename's extension is already numeric (for example, **cust.ab1**), the last character of the data file's root filename is replaced with an underscore (for example, **cus\_.ab1**). In both cases, the index file (**customer.dat** or **cust.ab1**) and the data file (**customer.da1** or **cus\_.ab1**) are always referenced by the name of the index file (**customer.dat** or **cust.ab1**).

## ISAM file types

When creating an ISAM file, you can specify one of three file types:

- ▶ Fixed-length
- ▶ Variable-length
- ▶ Multiple fixed-length

If your ISAM file is used for only one data structure, you can use the fixed-length format. With this file type, all data in the ISAM data partition is stored in the same length record, regardless of the actual size of the data within the record.

If your ISAM file is used for a predefined group of different sized data structures, you can use the multiple fixed-length record format. The size of the stored record is determined by the data passed to the STORE statement. With this file type, you can't change the record lengths, using the WRITE statement, after the data has been stored. Multiple fixed-length files can have up to 32 different record lengths. Using this file type enables you to reduce disk storage requirements and the number of open files. This file type is more efficient in disk space usage than variable-length records for cases where there are a limited number of different record sizes. Change tracking is not allowed with multiple fixed-length files.

If your ISAM file is used to store different types of records and it has no set pattern to the record size, or if the data length might change after the initial data is stored, you can use variable-length records. Like multiple fixed-length records, the initial size of the stored data is determined by the size of the data passed to the STORE statement. With variable-length records, however, you can change the size of the record using the WRITE statement.

The **isload** and **fconvert** ISAM utilities recognize one additional file type called a *counted file*. The **isutl** utility may also create a counted file in the form of an exception file, due to specific failures encountered during the recovery process. Counted files are *not* supported by the OPEN statement. Each record in a counted file starts with a two-byte length, which is the length of the record written as a portable integer, followed by the record itself, padded out to an even number of bytes if the length is odd. The final two bytes of the file are 0xFFFF, or integer -1.

## ISAM index density

Three forms of index density occur during file updates:

- ▶ Natural fill or default density. Index blocks are always filled to maximum capacity and then split in half (50/50).
- ▶ Balanced fill. Index blocks are filled to maximum capacity, balanced by rotating key entries between adjacent blocks to avoid splitting, and then split in half (50/50). Key rotation fills adjacent blocks to the specified density first and then fills the current block to maximum capacity before splitting.
- ▶ Context fill. Index blocks are filled to maximum capacity, and sequential keys generate a split at the specified density.



A split generates between 3 and  $(2 * \text{index depth}) + 1$  file writes with almost certain file extension. A rotation generates no more than 3 writes and no file size extension.

---

Natural fill is enabled by default (in other words, if you don't specify the file or key density options). Balanced and context fill are enabled for all keys when the file density option is specified, or for the specified keys when the key density option is specified. Routine file updates (STORE, WRITE, and DELETE) use the defined density form and pack indexes accordingly. In addition, the **isutl** utility (**-rp** option) can be used to pack indexes to a desired density without changing the defined density option.

Keys stored randomly using natural fill will typically fill to about 68 percent, while keys stored sequentially will fill to 50 percent. Keys stored randomly using balanced fill will typically fill to about 73 percent, while keys stored sequentially will fill to the defined density.

We believe natural fill is best for files that regularly have large volumes of file updates, and it should be followed up by running **isutl** at regular intervals to optimize the indexes. For files that normally have few updates, like window libraries, archives, and other read-only files, use the natural fill and then use the **isutl -rp** option to pack the indices to 100 percent. This will reduce the size of the file by compacting the indexes and improving read performance.

Keys that represent a sequence number or any continually increasing value that grows sequentially (such as a key based on time) where insertions to the middle of the sequence are rare may be candidates for context fill at 100 percent density. If these keys allow duplicates, make sure the duplicates are inserted at the end. The STORE operations will maintain the index density at 100 percent. Keys of this nature take up half the space normally taken by keys with no specified density.

### File creation density options

- ▶ **File density.** The file density designates the key density for each index when the file is created. If specified, each index is populated with a matching key density, and then the file density is discarded from the file options. The **isutl** utility (with the **-qfile=density** option) can be used to add or change the file density option, which in turn adds or changes the key density option for each index.
- ▶ **Key density.** The key density designates how a specific index will be packed during update operations.

### Data compression

When creating an ISAM file, you can specify that you want your file to compress data. A repeated string of characters can be compressed to a few bytes. Compression can save from 10 percent to 80 percent of your disk space requirements for the data portion of an ISAM file (**.is1**), with no program changes. Records containing text fields are ideal candidates for compression. You can compress fixed-length or variable-length files but not multiple fixed-length files.



If you require that your RFAs remain the same on WRITE operations, you must use static RFAs. (Static RFAs are required with Revision 6 ISAM files.) With data compression, your compressed data record size may change, causing the RFA to change. Do not use data compression if you expect a record's RFA to remain the same after a WRITE, unless you build the file with static RFAs as well. (In other words, your response to the **bldism** prompt "Enter name of the ISAM file to create:" would be *filename,compress,static\_rfa*.) This is especially true if you use manual locking.

If you use xfODBC and queries that have joins that do not use a default index (i.e., joins for which xfODBC must create temporary indexes), you *must* use static RFAs on your files.

---

### Page size

You can control the size of each index page or block for each key in the index file by specifying a file page size of 512, 1024, 2048, 4096, 8192, 16384, or 32768 in the ISAMC routine. The default page size is 4096 for Revision 6 ISAM files. (For Revision 4 files, the default is 1024.) In addition to considering the operating system, the size of the largest key should also determine what page size should be used. This enables more keys to fit in a single block, thus reducing the overall depth of the key's index. Keyed access is faster with a smaller depth, and the file is smaller due to less index at the top. Also, very large files benefit most from increased page size. For optimal keyed READ performance, try to keep the index depth around 3 or 4, even though the CPU time required to search the larger index is increased. In most cases, the tradeoff is worth it, because CPU is faster than I/O.

When creating a file with a page size of 512 (PAGE=512), the maximum internal key size allowed is 100 bytes. The internal key size is the specified key size plus 3 if the key allows duplicates (4 for terabyte files). All keys defined for this file are limited to this maximum.



With page sizes larger than 1024, consider defining a higher index packing density. (See “ISAM index density” on page 3-8 for more information.) Maximum gain from larger page sizes can only be achieved in conjunction with higher blocking factors. **lpar** reports the current depth of a file, and **isutl -v** reports the actual density.

---

## Change tracking

Change tracking is an internal structure maintained by each individual ISAM file. Internal control entries, or change records, are automatically recorded to a file for each file update that occurs. Updates include STORE, DELETE, and WRITE operations, as well as the `Select.AlphaEnumerator.Current` property set method. Periodic timing controls, called snapshots, can then be applied to manage these change records over time. These controls are transparent to normal file operation, but you can use external routines to access these controls and influence sequential operations on the file. Change records represent all user updates and are accessed as a whole rather than by individual user.

Change tracking enables you to quickly access changes made to a file over a period of time. For example, change records can be used to apply period-end figures, update an SQL database, or do anything that would otherwise require a top-to-bottom sequential scan of an entire file when only changes made during a specific time period need to be queried. Reclamation and rollback operations are allowed at file snapshot points unless the rollback function is turned off. A file snapshot can also be used as a backup restore point.

When change tracking is first applied to a file, or when the change tracking file has been cleared, an implied snapshot {#0} is applied to represent the beginning point, and the current snapshot number is set to 1. (This beginning point remains throughout the life of the file; however, as old snapshots are freed, the beginning point assumes the identity of the oldest applied snapshot.) All file inserts, updates, and deletes are logged with the current snapshot number until a new snapshot is applied using the **ctutl** utility (numerically identified by the current snapshot number, which is then incremented). Snapshot numbers continue to grow numerically in value until the file is cleared or re-initialized. Rolling a file back to an earlier snapshot also sets the current snapshot number back. Snapshots are freed (beginning with the oldest) and rolled back (beginning with the newest) using the **ctutl** utility. A maximum of 255 concurrent snapshots is allowed at one time.



When using change tracking, we recommend using frequent snapshots (daily, weekly, or monthly, depending on your transaction cycle) and then managing those snapshots by freeing old ones as they become unnecessary. Neglecting to manage snapshots will result in unwanted file growth and/or exceeding the 255 concurrent snapshot maximum.

---

A file's change tracking history is accessed using the Select class. The From class Changes and NetChanges methods allow you to programmatically select all changes and net changes, respectively, made between two snapshots. In addition, the From class Snapshot method enables you to make selections against a file entirely as it was when an applied snapshot was made. Using this method, end-of-period processing can occur without suspending current file update activity.

All changes made between one snapshot and the next snapshot get recorded as net changes. In other words, if you insert a record with a STORE and then update the same record with a WRITE before the next snapshot is made, the net result is an insert with the contents as of the last WRITE. Similarly, if a WRITE occurs at the beginning of a snapshot followed by another WRITE before the next snapshot, the result is an update with the contents of the last WRITE. Lastly, if a record is inserted and then deleted during the same snapshot, the record and its change history are removed entirely. The following table summarizes these results:

Changes	Net result
Insert + Update	Insert
Update + Update	Update
Update + Delete	Delete
Insert + Delete	<none>



Snapshots are maintained within the ISAM file itself. If you unload the records or convert them with **fconvert**, the change history is not preserved in the new file.

ISAM files must be at Revision 6 in order to use change tracking capabilities. Use the **ctutil** utility to manage change tracking information on the files you've chosen. (See [ctutil](#) on page 3-50 for details.) Change tracking is not allowed with multiple fixed-length files.

## Data caching

The Synergy runtime performs three types of caching, depending on how the file was opened.

- ▶ Files opened with exclusive access get full cache (read and write). The file is not written to disk until a CLOSE or FLUSH statement is processed.
- ▶ Files opened with exclusive "allow readers" access get a write-through form of cache. No WRITE operations are cached, but all READ operations attempt to come from cache, which is not fully cleared until a CLOSE or FLUSH statement is processed.
- ▶ Files opened without exclusive access get a write-through form of cache. No WRITE operations are cached, but all READ operations attempt to come from cache, which is not fully cleared until the file is updated by another user or a CLOSE or FLUSH statement is processed without system option #3. (Note that system option #3 is automatically implied in Synergy/DE 10 and higher.) READ and WRITE operations are not cached without exclusive access.

See [NUMBUFS](#) in the “Environment Variables” chapter of *Environment Variables & System Options* for information on cache tuning.

## Data file record structure

On operating systems that support large files, Synergy DBL supports individual ISAM files (**.ism** and **.is1**) of up to 256 terabytes in size. To create a terabyte file, specify the TBYTE option in the ISAMC subroutine. (See [ISAMC](#) in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual* for details.) By default, ISAM files are limited to 2 GB.

On Windows, terabyte files are supported by Synergy and xfServer/xfServerPlus only on systems with the NTFS file system. They are not supported on FAT or FAT32 file systems.

On UNIX, some operating systems require you to set a large-file option on the file system being used. Some (AIX in particular) require this to be done when the file system is first created, and others allow the option to be added later.

When creating an ISAM file, you can declare a minimum record size of 4 bytes (5 for TBYTE files) and a maximum of 65,535 bytes minus the following overhead:

- ▶ Each duplicate key adds 3 bytes.
- ▶ All file types except fixed-length add 2 bytes.
- ▶ Static RFAs on any file type except fixed-length adds 6 bytes.
- ▶ One delete byte is automatically added.
- ▶ Change tracking adds 12 bytes.

On OpenVMS, the maximum RMS record size is 32,234.

## Portable storage format

Synergy ISAM storage format is the same on all Synergy DBL systems (except OpenVMS); therefore, you can copy ISAM files to any Windows or UNIX system and access them without conversion. This portable storage format also enables you to access ISAM files across heterogeneous networks.



Using integer data in your records may affect portability. Integer data is not universally portable unless you define it using the I option in **bldism** or the ISAMC subroutine. If you use the I option, files can be moved to other machines and accessed across heterogeneous networks without having to apply any conversion at the application layer.

---

Portable integer data can be stored in an ISAM file and retrieved portable across all platforms except OpenVMS.



## Keys in ISAM files

When you create an ISAM file, you must define at least one key (the primary key) by which to access that file. You can define up to 255 keys: 1 primary key and 254 alternate keys.

A defined key can have the following attributes:

- ▶ Named key of reference
- ▶ Key type/segment types
- ▶ Duplicate
- ▶ Modifiable
- ▶ Segmented
- ▶ Null value
- ▶ Ascending or descending order per key or per segment
- ▶ Specified density

The maximum overall length of a key may not exceed 254 bytes on Windows and UNIX (251 if the key allows duplicates or 250 if the key allows duplicates and this is a terabyte file), or 255 bytes on OpenVMS.

### Named key of reference

When defining a key, you can specify an optional identifying string to be used in key-of-reference specifications for Synergy ISAM file access.

### Key type

Each key in an ISAM file may be made up of one or more segments of the following key types:

- ▶ ALPHA (default)  
Alphanumeric key. The standard ASCII character set is valid for each character of the key (although the entire binary 0 to 255 range is allowed).
- ▶ NOCASE  
Case-insensitive alphanumeric key.



Case-sensitive keys cannot be used on OpenVMS nor for optimization with our ODBC drivers.

Using NOCASE keys with 8-bit multinational characters is not recommended and may cause unpredictable results.

---

- ▶ DECIMAL  
Zoned decimal key (Synergy DBL decimal data type). The valid range of values allowed for a decimal key is the maximum negative value to the maximum positive value for the size of the

defined key. Implied-decimal values may also be used; however, the number of digits to the right of the decimal point must be maintained by the application.

d1 = -9 to 9

d2 = -99 to 99

etc.



Decimal keys cannot be used on OpenVMS.

---

► **INTEGER**

Native integer key (**i1**, **i2**, **i4**, or **i8**). The valid range of values allowed for an integer key is the maximum negative value to the maximum positive value for the size of the defined key.

i1 = -128 to 127

i2 = -32768 to 32767

etc.

► **UNSIGNED**

Native unsigned integer key (with the same restraints as integer). The valid range of values allowed for an unsigned integer key is 0 to the maximum positive value for the size of the defined key.

%unsigned(i1) = 0 to 255

%unsigned(i2) = 0 to 65535

etc.



Unsigned keys cannot be used for optimization with our ODBC drivers.

---

► **SEQUENCE**

An **i8** key that creates an automatically incrementing number between 1 and 9,223,372,036,854,775,807 that is guaranteed to generate a unique key within the file. The key's value is generated on the initial STORE and will remain for the life of the record until deleted.

► **TIMESTAMP**

An **i8** key that represents the current UTC time in microseconds. It maintains a timestamp of the last time modified. A timestamp key is set to the current time (or that of the server) on every STORE, WRITE, and optionally DELETE. To programmatically convert the **i8** time to the local DATETIME, use the %DATETIME\_FROM\_I8 routine.

► **CTIMESTAMP**

An **i8** key that represents the create timestamp time in microseconds. Unlike **TIMESTAMP**, which changes on every record update, the **CTIMESTAMP** value is only set on the initial **STORE** of a record.

Numeric keys or key segments may not overlap or be overlapped by any other key segment (alpha or numeric). However, you may specify the same numeric key segment in more than one key.



On Windows and UNIX, using integer keys (**INTEGER**, **UNSIGNED**, **SEQUENCE**, **TIMESTAMP**, and **CTIMESTAMP** key types) in an ISAM file requires special handling when unloading and loading is performed. Due to the chance that one of these keys contains data that could be interpreted as a record terminator, unloading to a sequential (or text) file is not recommended. Instead, we recommend using the **fconvert** utility to generate a counted file (**-oc**). (See [fconvert](#) on page 3-62 for more information and further restrictions on text files.)

---

Synergy DBMS automatically fills autokeys (**SEQUENCE**, **TIMESTAMP**, and **CTIMESTAMP**) with the appropriate values. A sequence key starts at 1 and is incremented in sequence for every record stored until the file is cleared with **ISCLR**. A sequence key cannot be descending. A record with autokeys passed to a **STORE** or **WRITE** statement will be updated to reflect the autokey values that were stored or written, unless a literal was passed. This allows you to retrieve the autokey values from the record without having to reread it. An autokey must be declared the appropriate size and cannot be modified, does not allow duplicates, cannot be segmented, and cannot be a null key. By default, autokey values are preserved (with the exception of a timestamp key where the incoming key data amounts to an empty key; in this case, all zeros, blanks, or nulls will be generated to the current timestamp).



A file that contains a timestamp key cannot be opened in update mode across a network share or NFS drive unless it's opened exclusively (**SHARE:Q\_EXCL\_RW** or **SHARE:Q\_EXCL\_RO**). If you try to open a file with a timestamp key in update mode on a mapped drive, you'll get a "Network share is not allowed with this file" error (**NONETSHR**). This restriction exists to prevent damaging the integrity of the timestamp key. Use **xfServer** in this situation. Note that utilities that gain exclusive access (e.g., **fconvert** and **isutl**) will allow this kind of access.

---

If both **TIMESTAMP** and **CTIMESTAMP** are in the same record, you can compare their values to see if the record's original contents have been updated.

### **Duplicate keys**

A duplicate key is a key value found in more than one record of an ISAM file. If you don't allow duplicate key values in an index, each record in the file is uniquely identified by its key value. With duplicate keys, for example, we can define our zip code field as a duplicate key so that many different records can contain the same value for the zip code. However, if we also define a key for our customer number field, we probably don't want to allow duplicate keys, so that there will only be one record in the index for each customer number.

When duplicate keys are allowed, you must also specify the order in which a set of duplicate key values are stored within an index and retrieved from a file. Since duplicate keys are internally unique, they are stored sequentially based on the order defined for duplicates. You can either insert duplicate keys at the end of a list of records possessing the same key value or insert duplicates at the front of such records. If you insert duplicate keys at the end, records are retrieved in the same order that they were stored in the file: “first in, first out” (FIFO) order. If you insert duplicates at the front, the first records retrieved are those stored most recently in the file: “last in, first out” (LIFO) order. The default is to insert at the end (FIFO), which is the same as on OpenVMS, where duplicate records are always inserted at the end.

For example, we can define our customer city field as a duplicate key with duplicates inserted at the front of a list of matching records. If our customer ISAM file contains five customers from Baltimore and we accessed that file by the customer city key, we’d retrieve the most recently stored customers first, as shown below:

STORE order	READS order
B. Jones	L. Peterson
C. Smith	R. Carey
A. Johnson	A. Johnson
R. Carey	C. Smith
L. Peterson	B. Jones

---

## WIN, UNIX

Allowing duplicates adds 3 bytes (4 bytes for terabyte files) to the internal size of the key, which cannot exceed a total of 254 bytes.

---

## Modifiable keys

If a key is modifiable, Synergy ISAM allows your application to update an existing record and change the value of the defined key using the **WRITE** statement.

---



The primary key cannot be a modifiable key.

---

For example, if we define our customer telephone field as a modifiable key, we can change the value of this key using the **WRITE** statement if a customer’s phone number changes. However, we probably don’t want to define our customer number field as a modifiable key, since this value should not change during the life of the file. To change a nonmodifiable key, you must use the **DELETE** and **STORE** statements.

## Segmented keys

Keys can consist of up to eight segments. The total length of the key (up to 254 characters on Windows and UNIX [251 if the key allows duplicates or 250 if the key allows duplicates and this is a terabyte file], or 255 characters on OpenVMS) is equal to the sum of the lengths of the key segments. Key segments usually correspond to fields in a record, but they do not have to be in any particular order. Segments can be defined as different types and ordered ascending or descending.

### VMS

---

Due to an RMS limitation, multiple segments of a key must all have the same order.

For example, we can define a customer address key with four segments. The first segment can be 25 characters long and correspond to our street address field; the second can be 15 characters long and correspond to our city field; the third can be 2 characters long and correspond to our state field; and the fourth can be 5 characters long and correspond to our zip code field. The total length of this key is 47 characters long.

Different alpha keys and key segments can overlap each other in a record. Numeric keys and key segments cannot overlap any other key segments unless the segment types, starting positions, and lengths are equal.

To access a segmented key, you must first construct that key by concatenating each segment together. You can use the `%KEYVAL` intrinsic function to return the extracted key value from the specified record. See `%KEYVAL` in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual* for more information.

### WIN, UNIX

---

Partial key specifications on segmented keys are allowed when system option #45 is set.

## Null keys

You can specify a null value for any key except the primary key. No entry is made in an index that is defined to have a null value if the inserted record contains the null value for that key. Therefore, when accessing a file by a null key, Synergy ISAM skips over records that contain the specified null value.

Null keys can be useful in a record that contains an optional key field. When the field is blank or contains a value of 0 (depending on the field's data type), the field doesn't occupy space in the index. Thus, the use of null keys reduces the size of the index file as well as the overhead time required to insert, delete, or modify a record with a null value. An index allowing null keys can only be used for limited optimization with our ODBC drivers.

You can specify one of three different types of null keys:

- ▶ Replicating
- ▶ Nonreplicating (Windows and UNIX only)
- ▶ Short (Windows and UNIX only)

A replicating null key's value must be either a decimal character or its corresponding ASCII character. This type of null key generates a null entry if every byte of that key matches the specified null value.

The following table shows some possible null values in decimal and ASCII form:

	Null values for alpha keys		
	Zero	Space	Null
<b>Decimal</b>	48	32	0
<b>ASCII</b>	"0"	" "	"\0"

The null value for a numeric key defined as a replicating null key is always binary zero for unsigned and integer keys and decimal zero for decimal keys. A specified *null\_value* for this key type is ignored. When defining a replicating null key on a key that has multiple segments of different types, the *null\_value* only refers to the alpha segment (if any). If there are no alpha segments and *null\_value* is specified, *null\_value* is ignored.

A nonreplicating null key's value is a string (quotes are optional). This type of null key generates a null entry if the key matches the string for the length of the string starting at the beginning of the key. Nonreplicating null keys can be defined for either alpha or numeric keys; however, the allowable value depends on the type:

- ▶ If the key is alpha, an alpha string is specified for the null key value. If that key is segmented, the length of the alpha string must not cause the value to overlap a numeric segment.
- ▶ If the key is numeric, a numeric value is specified for the null key. The key may not be segmented. The allowable numeric values depend on the type and length of the key.

A short null key does not have a specified null value. This type of null key generates a null entry if the record doesn't include the entire key on a STORE or WRITE operation. Short null keys can only be defined for ISAM files that are not fixed-length.

### Ascending or descending keys

By default, Synergy ISAM sequentially retrieves keys in ascending order (lowest to highest). When creating an ISAM file, however, you can specify that you want a particular key or segment retrieved in descending order (highest to lowest).

## Key density

You can define a specific density for an individual key or keys, while leaving the rest of the keys at the default file density. See “[ISAM index density](#)” on page 3-8 for more information about density.

## File corruption vs. data corruption

As it applies to ISAM files, file corruption occurs when the control information in an index file doesn’t correspond to the records in the data file. Data corruption occurs when records in the data file have been unexpectedly overwritten or inserted. File corruption can be detected by running the **isutl -v** utility and corrected by running the **isutl -r** utility. When data corruption is detected, **isutl** fails, usually with a BADSEG error, and the file remains undisturbed. You will be prompted to run **isutl** again with the **-a** option. Any records that cannot be processed will be written to an exception file with the extension **.exc**.

## Recovery options and strategies

When recovering data from a corrupted ISAM file, we recommend you first copy the ISAM file (both **.ism** and **.isl**). Then, if recovery should fail with one method, you can try other methods.

The file with the highest chances for recovery from data corruption is a file that employs data compression. This is because the codes used to compress the data can also be used as a roadmap to distinguish between good data and data corruption. If one or more data records near the beginning of a file are corrupted, **isutl** can skip them (these bad record segments are automatically sent to the exception file as they are found in the data file) and continue recovering the rest of the file. You may be able to reconstruct the lost records by examining the exception file.



The exception file is a counted file and the data written is a copy of the binary data segment exactly as it was in the data file. You can use **fconvert** to convert the counted file to something easier to work with.

---

It is more difficult to attempt recovery from data corruption on files without data compression. **Isutl** may detect data corruption in files with variable-length records, but the only thing it can do is to write the rest of the data file to the exception file. If the index file is in good shape (with at least one good key), you may recover more data by using **fconvert** to recover the file.



When running **isutl -v** on a file with corrupted data, look for keys displaying fewer index related errors or data pointer errors.

---

Files with fixed-length records and no data compression make detecting data corruption most difficult. The data retrieved from the data file is whatever happens to be at the stored location. If one record gets written with the wrong size, every record after it will be at the wrong file boundary. This phenomenon has been known to happen during system crashes and other abnormal terminations. Previous versions of Synergy ISAM continued storing records and making index links to new records, and the file continued to operate as normal. **ismvfy** detected the problem, but **irecovr**

wasn't able to recover from it. The current version of Synergy ISAM checks record boundaries before it writes data. If an invalid boundary is detected, an error is produced, and you cannot extend the file by adding records to it until you recover the file using **isutl**.

## ISAM limits

The following are the capacities and minimum/maximum limits of Synergy ISAM.

<b>Synergy ISAM capacities and limits</b>		
<b>Capacity</b>	<b>Maximum</b>	<b>Minimum</b>
Keys defined per file	255	1
Segments defined per key	8	1
Length of key	254 on Windows and UNIX (or, 251 if the key allows duplicates or 250 if the key allows duplicates and this is a terabyte file) 255 on OpenVMS	1
Length of key segment	Same as defined key length	—
Number of records per file	Approximately 250,000,000 for nonterabyte files The actual value varies depending on key size, index density, and available disk space.	0
Record length <sup>a</sup>	On Windows and UNIX, use this formula: 65,534 – (3 * number of keys allowing duplicates), – 2 if variable or compressed, – 6 if static RFA, – 2 if static RFA and variable or compressed 32,234 on OpenVMS	4 (or 5 for terabyte files)
Size per disk file (.ism and .is1)	2 <sup>31</sup> (or 2 <sup>48</sup> for terabyte files)	—
Combined null key size per file	1K	0
Keys per duplicate key value	16,777,216 (or 4 billion for terabyte files)	—
Index depth per key	16	—
Static RFA reuse	127	—

- a. In Synergy .NET and 64-bit traditional Synergy, a file created with a variable record type and a maximum record size of 0 can contain larger records that are limited only by the amount of memory you have available or 2 GB.



## ISAM input and output statements

Input and output (I/O) statements that can be used with ISAM files are as follows:

CLOSE	Close a channel.
DELETE	Delete a record.
FIND	Find a record.
OPEN	Open a channel.
READ	Read a specified record.
READS	Read the next sequential record.
STORE	Store a record to an ISAM file.
UNLOCK	Release a record lock.
WRITE	Update a record.

The “[Synergy DBL Statements](#)” chapter of the *Synergy DBL Language Reference Manual* contains the syntax, arguments, discussion, and examples for each of the I/O statements.

## ISAM routines

The system-supplied ISAM subroutines and functions enable you to manipulate ISAM files from within your applications:

FREE	Release all locks on a specified channel.
ISAMC	Create an ISAM file.
ISCLR	Clear or empties an ISAM file.
%ISINFO	Return numeric ISAM file status and key information.
%ISINFOA	Return alpha ISAM file status and key information
ISKEY	Return information about a specified key in an ISAM file.
ISSTS	Return the status of an ISAM file.

The “[System-Supplied Subroutines and Functions](#)” chapter of the *Synergy DBL Language Reference Manual* contains the syntax, arguments, discussion, and examples for each of the ISAM routines.

## Synergy relative files

Synergy relative files are used to access records by relative record number. The physical file format varies slightly on each operating system.

On Windows and UNIX, Synergy DBMS accesses records in relative files. Binary data can be in the records because all records are assumed to be of the same length followed by the record terminator, so the data is not scanned for the end of the record. If the record terminator is not found in the correct location, an “Invalid relative record” error (\$ERR\_RELREC) is generated. The record size can be specified either by the destination size on the first I/O statement or by the RECSIZ qualifier on the OPEN statement.

On OpenVMS, the RMS file system is used for native compatibility. Binary data can be in the records because RMS knows the length of each record and does not depend on the record terminator to separate each record. When the file is opened, the record length is automatically retrieved. If the RECSIZ is specified, the value is compared against the actual record size and the error generated is IRCSIZ if they do not match.

To support terabyte relative files, both the operating system and file system must be 64 bit.

### Relative file structure

On Windows and UNIX, a relative file consists of a byte stream where the records all contain the same number of bytes followed by the record terminator. On Windows, the record terminator is a CR-LF (carriage return and line feed) byte pair. On UNIX, the record terminator is a single LF (line feed) byte. Random record positioning is accomplished by multiplying the record number minus one by the sum of the record size and the number of bytes in the record terminator to determine the byte offset from the beginning of the file.

On OpenVMS, a relative file is a specific RMS file type. Each record consists of only the data without record terminators. Each record is accessed by a record number index that ranges from 1 through 2147483647.

### Relative file types

On Windows and UNIX, there is only one file type where all of the records are the same length.

On OpenVMS, relative files can either contain fixed-length records or variable-length records depending on how the record format for the file was specified when it was created. The maximum size for fixed-length records in a relative file is 32,255 bytes. The maximum size for variable-length records is 32,253 bytes. For VFC (variable-length with fixed-length control field) records, the maximum size is 32,253 bytes minus the size of the fixed-length control field, which may be up to 255 bytes long. The RECTYPE qualifier of the OPEN statement is used to specify the record format.

To create a relative file, specify the mode as O:R or A:R in the OPEN statement. To open an existing relative file, specify the mode as I:R, U:R, or A:R in the OPEN statement. The record size can be specified in the OPEN statement using the RECSIZ qualifier. On OpenVMS, the record size is stored in the file header, so if RECSIZ is specified, the record size is compared against the

RECSIZ value. If the RECSIZ qualifier is not specified on Windows or UNIX, the record size is determined by the size of the destination area on the first I/O operation. If the RECSIZ qualifier is specified as -1 on Windows or UNIX, the record size is determined by the size of the first record in the file.

## Record access

Records are accessed randomly by specifying the record number as a numeric field in the key field of READ, FIND, and WRITE statements, or sequentially using the READS or WRITES statements.

## Relative record input and output statements

This section lists the primary input and output statements and describes how their use affects relative files. System-specific differences are also listed. See the [“Synergy DBL Statements”](#) chapter of the *Synergy DBL Language Reference Manual* for more information about other statement qualifiers that are not specific to relative file access.

### READ statement

`read(channel, record, record_number)`

You can use the READ statement to retrieve a record from the file by specifying the record number. To specify the first or last record in the file, replace the record number with ^FIRST or ^LAST, respectively. The POSITION qualifier can replace ^FIRST or ^LAST.

On Windows or UNIX, a READ of an unwritten record returns data of indeterminate contents. If the record terminator is not found in the file at the end of the fixed number of bytes, an “Invalid relative record” error (\$ERR\_RELREC) is generated.

On OpenVMS, a READ of an unwritten record results in a “Record not found” error (\$ERR\_RNF).

### FIND statement

`find(channel, record, record_number)`

The FIND statement positions to the record specified by *record\_number*. To specify positioning to the first or last record in the file, replace the record number with ^FIRST or ^LAST, respectively. To specify positioning to the beginning of the file (before the first record) or end of the file (after the last record), replace the record number with ^BOF or ^EOF, respectively. The POSITION qualifier can replace ^FIRST, ^LAST, ^BOF, or ^EOF.

On Windows or UNIX, a FIND to an unwritten record proceeds without error, as positioning in the file is all that occurs.

On OpenVMS, a FIND to an unwritten record results in a “Record not found” error (\$ERR\_RNF).

**WRITE statement**

`write(channel, record, record_number)`

The WRITE statement updates a record in the file or adds the specified record to the file. To specify the first or last record in the file, replace the record number with ^FIRST or ^LAST, respectively. The POSITION qualifier can replace ^FIRST or ^LAST.

On Windows or UNIX, if a WRITE statement specifies a record number beyond the last record written to the file when it was opened in append or output mode, the file is extended by the size required to include the unwritten records. The contents of these unwritten records are undefined.

On OpenVMS, if a WRITE statement specifies a record number beyond the last record written to the file when it was opened in append or output mode, the specified record is written to the file, and the unwritten records between the previous last record and the record just written are left as “holes” in the file.

**READS statement**

`reads(channel, record, eof_label [, DIRECTION=Q_REVERSE])`

The READS statement retrieves the record that is sequentially next in the file. When DIRECTION=Q\_REVERSE is specified, the previous sequential record is retrieved.

On Windows or UNIX, a READS of an unwritten record returns data of indeterminate contents. If the record terminator is not found in the file at the end of the fixed number of bytes, an “Invalid relative record” error (\$ERR\_RELREC) is generated.

On OpenVMS, a READS statement skips unwritten records and retrieves the next record in the file without error.

**WRITES statement**

`writes(channel, record)`

The WRITES statement updates the record that is sequentially next in the file. If the file is opened in append or output mode, the operation adds the next sequential record to the file.

**DELETE statement**

`delete(channel)`

The DELETE statement is only available for use on relative files on OpenVMS, because RMS allows “holes” of unwritten records to exist in relative files.

**UNLOCK statement**

`unlock channel[, RFA:match_rfa]`

The UNLOCK statement unlocks any records that have automatic locks. If RFA:match\_rfa is specified, the specified record with the manual lock is the only record unlocked. The RFA:match\_rfa qualifier is ignored on Windows and UNIX.

### **%RSIZE function**

*size* = %rsize

The %RSIZE function returns the size of the last record read, excluding the record terminator. The returned value is for the last operation, regardless of the channel used.

### **%RTERM function**

*value* = %rterm

For READ and READS operations, %RTERM returns the record terminator of the last operation, regardless of the channel used.

### **%RECNUM function**

*number* = %RECNUM(*channel*)

The %RECNUM function returns the relative record number of the last accessed record.

## **Record locking**

When a relative file is opened in update or output mode, record locking is in effect by default unless the LOCK:Q\_NO\_LOCK option is specified on the OPEN statement. The FIND, READ, and READS statements unlock any previously locked record. READ and READS also lock the specified record.

## **Synergy sequential files**

Synergy sequential files are used to access records sequentially from the beginning of the file to the end of the file. Records are not accessed randomly. The physical file format varies slightly on each operating system.

On Windows and UNIX, Synergy DBMS accesses records in sequential files. We do not recommend placing binary data in the records, as the size of each record is determined by the placement of the record terminator and the binary data can be mistaken for the record terminator.

On OpenVMS, the RMS file system is used for native compatibility. The records can contain binary data because RMS knows the length of each record and does not depend on the record terminator to separate each record. When the file is opened, the record type is automatically retrieved.

To support terabyte sequential files, both the operating system and file system must be 64 bit.

## **Sequential file structure**

On Windows and UNIX, a sequential file consists of a byte stream where the end of each record is defined by the location of the record terminator. On Windows, the record terminator is normally a CR-LF (carriage return and line feed) byte pair but can also be a single LF byte. On UNIX, the record terminator is a single LF (line feed) byte. On either system, a record terminator can also be a VT (vertical tab) byte or an FF (form feed) byte.

On OpenVMS, a sequential file is a specific RMS file type. Each record consists of only the data without record terminators.

## Sequential file types

On Windows and UNIX, there is only one sequential file type: a byte stream where the records are defined by the placement of the record terminator.

On OpenVMS, sequential files can either contain fixed-length records or variable-length records depending on how the record format for the file was specified when it was created. The maximum size of a record in a sequential file is 65,535 bytes. The records in a sequential file are preceded by two bytes that specify the length of the record, and if the record length is odd, a null byte follows the record. RMS masks this physical format of the file so only the data is stored or retrieved.

To create a sequential file, specify the mode as O:S or A:S on the OPEN statement. On OpenVMS, if the mode is specified as O or A without a submode, and the program was either not compiled with the /STREAM switch or the OPTIONS="/STREAM" qualifier was not specified, the file is created as a sequential file. To open an existing sequential file, specify the mode as I:S, U:S, or A:S on the OPEN statement.

## Record access

Records are accessed sequentially from the first record through the end of the file.

## Sequential record input and output statements

This section lists the primary input and output statements and describes how their use affects sequential files. System-specific differences are also listed. See the [“Synergy DBL Statements”](#) chapter of the *Synergy DBL Language Reference Manual* for more information about other statement qualifiers that are not specific to sequential file access.

### **READS statement**

`reads (channel, record, eof_label)`

The READS statement retrieves the next sequential record in the file.

On Windows or UNIX, a READS statement retrieves data from the file based on the size of the destination field plus the size of a record terminator and then searches the data for the record terminator. The data up to the record terminator is transferred to the destination field and left-justified over blanks.

### **WRITES statement**

`writes (channel, record)`

The WRITES statement writes the record plus the record terminator to the file at the current location. If the file is opened in U:S mode, the WRITES statement can be used for updating the record that was previously read.

### **FIND statement**

```
find(channel, , , POSITION:Q_BOF)
```

The FIND statement repositions to the beginning of the file.

### **%RSIZE function**

```
size = %rsize
```

The %RSIZE function returns the size of the last record read but does not include the record terminator. The returned value is for the last operation, regardless of the channel used.

### **%RTERM function**

```
value = %rterm
```

For READ and READS operations, %RTERM returns the record terminator of the last operation, regardless of the channel used.

## **Record locking**

Record locking occurs by default if the file is opened in U:S mode unless the LOCK:Q\_NO\_LOCK option is specified. The READS statement causes the previous record to be unlocked and the record specified by the statement to be locked.

## **Synergy stream files**

Synergy stream files are used to access records sequentially or by relative record number.

### **Stream file structure**

A stream file consists of a byte stream where the end of each record is defined by the location of the record terminator. On UNIX, the default record terminator is a single LF (line feed) byte. On Windows and OpenVMS, the default record terminator is a LF CR (line feed and carriage return) byte pair, but the record terminator can also be a single LF (line feed) byte. On all systems, a record terminator can also be a VT (vertical tab) byte or a FF (form feed) byte.

Random record positioning is accomplished by multiplying the record number minus one by the sum of the record size and the number of bytes in the default record terminator to determine the byte offset from the beginning of the file.

## Stream file types

On Windows and UNIX, there is only one file type: a byte stream, where the records are defined by the placement of the record terminator.

To create a stream file, specify the mode as O or A without a submode on the OPEN statement. On OpenVMS, the `OPTIONS="/STREAM"` qualifier must also be present or the file must have been compiled with the `/STREAM` switch to create a stream file. To open an existing stream file, specify the mode as I, U, or A without a submode on the OPEN statement.

## Record access

Records are accessed randomly by specifying the record number as a numeric field in the key field of a READ, FIND, or WRITE statement, or sequentially using the READS or WRITES statements.

## Stream record input and output statements

This section lists the primary input and output statements and describes how their use affects stream files. System-specific differences are also listed. See the [“Synergy DBL Statements”](#) chapter of the *Synergy DBL Language Reference Manual* for more information about other statement qualifiers that are not specific to stream file access.

### READ statement

```
read(channel, record, record_number)
```

You can use the READ statement to retrieve a record from the file by specifying the relative record number. To specify the first or last record in the file, replace the record number with `^FIRST` or `^LAST`, respectively. The `POSITION` qualifier can replace `^FIRST` or `^LAST`. READ returns the data from the new file position to the next record terminator.

### FIND statement

```
find(channel, record, record_number)
```

The FIND statement positions to the record specified by *record\_number*. To specify the first or last record in the file, replace the record number with `^FIRST` or `^LAST`, respectively. To specify the beginning (before the first record) or end of the file (after the last record, replace the record number with `^BOF` or `^EOF`, respectively. The `POSITION` qualifier can replace `^FIRST`, `^LAST`, `^BOF`, or `^EOF`.

### WRITE statement

```
write(channel, record, record_number)
```

You can use the WRITE statement to replace a record in the file or add a record at the end of the file. You can extend a stream file opened in update mode by specifying the record at the current end-of-file position. The WRITE statement writes the contents of the record plus the default record terminator.



**READS statement**

*reads (channel, record, eof\_label)*

The READS statement retrieves the next sequential record in the file. It returns the data from the current file position to the next record terminator.

**WRITES statement**

*writes (channel, record)*

The WRITES statement replaces the next sequential record in the file or adds a record at the end of the file. You can extend a stream file opened in update mode by writing the record when positioned at the current end-of-file position. The WRITES statement writes the contents of the record plus the record terminators.

**GET statement**

*get (channel, record, record\_number)*

The GET statement retrieves a record from the file by specifying the relative record number. It returns the contents of the file from the new file position for the length of the record, regardless of any record terminators.

**GETS statement**

*gets (channel, record, eof\_label)*

The GETS statement retrieves the next sequential record in the file. It returns the contents of the file from the current file position for the length of the record, regardless of any record terminators.

**PUT statement**

*put (channel, record, record\_number)*

The PUT statement replaces a record in the file or adds a record at the end of the file. You can extend a stream file opened in update mode by specifying the record at the current end-of-file position. The PUT statement writes only the contents of the record; the default record terminator is not written.

**PUTS statement**

*puts (channel, record, eof\_label)*

The PUTS statement replaces the next sequential record in the file or adds a record at the end of the file. You can extend a stream file opened in update mode by writing the record at the current end-of-file position. The PUTS statement writes only the contents of the record; the default record terminator is not written.

**UNLOCK statement**

`unlock(channel)`

The UNLOCK statement unlocks any records with automatic locks. On OpenVMS, the blocks encompassing the locked record are unlocked.

**%RSIZE function**

`size = %rsize`

The %RSIZE function retrieves the size of the last record read but does not include the record terminator. The returned value is for the last operation, regardless of the channel used.

**%RTERM function**

`value = %rterm`

For READ and READS operations, %RTERM returns the record terminator of the last operation, regardless of the channel used.

**%RECNUM function**

`number = %recnum(channel)`

The %RECNUM function returns the relative record number of the last accessed record.

**Record locking**

On Windows and UNIX, the bytes encompassing the record plus the record terminator are locked when the READ or READS statement is used. When the GET or GETS statement is used, only the bytes in the file for the length of the specified record are locked.

On OpenVMS, the blocks encompassing the record plus the record terminator are locked when the READ or READS statement is used. When a GET or GETS statement is used, the blocks encompassing the bytes in the file for the length of the specified record are locked. If the BUFSIZ qualifier is specified in the OPEN statement, the number of locked blocks may be greater.

## Synergy block file I/O

Synergy block file I/O is used to access files and block devices on a binary basis. All I/O to or from the file must be in multiples of 512 bytes. (For most operating systems, a minimum of 4096 bytes at a time is most efficient.) This type of I/O bypasses the file organization and retrieves or writes the raw data to and from the file or device.

### Block file structure

The block submode is a way to bypass the native file organization to manipulate the binary data in a file in multiples of 512-byte blocks.

### Block file types

There is no specific block file type on any system. Block file I/O is only a way to access the raw data in a file.

You can create a file in block mode by specifying a mode of O:B or A:B in the OPEN statement. To open an existing relative file, specify the mode as I:B, U:B, or A:B on the OPEN statement.

### Data access

Data is accessed randomly if you specify the block number as a numeric field in the key field of READ, FIND, and WRITE statements, or sequentially using the READS or WRITES statements.

### Block mode input and output statements

This section lists the primary input and output statements and describes how their use affects block I/O. See the [“Synergy DBL Statements”](#) chapter of the *Synergy DBL Language Reference Manual* for information about other statement qualifiers that are not specific to block mode access.

#### READ statement

```
read(channel, block, block_number)
```

You can use the READ statement to retrieve records from the file by specifying *block\_number*. To specify the first or last record in the file, replace the block number in the READ statement with ^FIRST or ^LAST, respectively. The POSITION qualifier can replace ^FIRST or ^LAST.

#### FIND statement

```
find(channel, block, block_number)
```

The FIND statement positions to the block specified by *block\_number*. To position to the first or last block in the file, replace the block number in the FIND statement with ^FIRST or ^LAST. To position to the beginning (before the first block) or end (after the last block) of the file, replace the block number in the FIND statement with ^BOF or ^EOF, respectively. The POSITION qualifier can replace ^FIRST, ^LAST, ^BOF, or ^EOF.

**WRITE statement**

```
write(channel, block, block_number)
```

You can use the WRITE statement to update blocks or add blocks to the file by specifying the block number. To specify the first or last block in the file, replace the block number in the WRITE statement with ^FIRST or ^LAST, respectively. The POSITION qualifier can replace ^FIRST or ^LAST. To extend a file opened in update mode, specify the block at the current end-of-file position.

**READS statement**

```
reads(channel, block, eof_label)
```

The READS statement retrieves the next sequential block in the file.

**WRITES statement**

```
writes(channel, block)
```

The WRITES statement updates the next sequential block in the file. If the file is opened in append or output mode, the operation adds the next sequential block to the file.

**%RSIZE function**

```
size = %rsize
```

The %RSIZE function retrieves the size of the last block read. When the end-of-file block is read, *size* is the actual number of bytes contained in that block. The value returned is for the last operation, regardless of the channel used.

## Synergy counted files

**WIN, UNIX** 

---

Synergy counted files are currently only used by the **isload** and **fconvert** utilities to safely unload and reload ISAM files that contain binary data (including INTEGER, UNSIGNED, SEQUENCE, TIMESTAMP, and CTIMESTAMP keys). You may decide to unload an ISAM file to a counted file instead of a relative file because with a counted file you don't need to specify the record size when you reload your ISAM file. For variable-length records containing binary data, we highly recommend you use a counted file.

### Counted file structure

A counted file consists of record entries in format of a two-byte (portable integer) record length followed by the record itself. When the record length is odd, a null byte is appended to the record and is discarded when read. A record length of 0xFFFF indicates the end of the file.

## Record access

The only access to a counted file is sequentially via **isload** or **fconvert**. The file type **/COUNTED** is required when specifying the file in **isload**, and the **-c** option is required when specifying the file using **fconvert**.



Files created with record sizes greater than 64K cannot be counted files.

---

## Input and output statements

There are currently no I/O statements in the language to directly read or write records in a counted file.

---

## Synergy DBMS Utilities

**Fconvert**, **bldism**, **isload**, and **status** are Synergy DBL utility programs intended to manipulate ISAM files from outside your programs. You can use these utilities to create, clear, load, and unload ISAM files and to retrieve ISAM file status. If you need to perform these functions from within your existing programs, you can chain to these utilities. We suggest, however, that for new development you follow these guidelines:

To	Use	Instead of
Create ISAM files	The ISAMC subroutine or the OPEN statement with O:I and an XDL file	<b>bldism</b>
Clear ISAM files	The ISCLR subroutine	(not applicable)
Retrieve ISAM file status	The ISINFO subroutine (or the ISSTS and ISKEY subroutines)	<b>status</b>
Load files	<b>fconvert</b>	The STORE statement or <b>isload</b>

See the “[System-Supplied Subroutines and Functions](#)” chapter of the *Synergy DBL Language Reference Manual* for more information about the ISAMC, ISCLR, %ISINFO, ISSTS, and ISKEY routines.

The **ipar** utility is used to generate parameter file descriptions of existing ISAM files. These files can be used by **fconvert** or as input to **bldism** to create new ISAM files. (See “[Parameter and XDL files](#)” on page 3-35 for a discussion of using parameter or XDL files as an alternative method of input for the **bldism**, **isload**, and **status** utilities.) **Ip**ar can also be used to view attributes of an existing ISAM file quickly.

In addition to loading files, the **fconvert** utility also converts database files from one file type to another. Local or remote files can be specified using xfServer file specifications. This utility has been optimized to attain the highest file load/unload performance.

The **isutl** utility verifies, reindexes, and performs maintenance on Revision 4 or higher ISAM files. Maintenance includes recovery of corrupted files, reclamation of deleted record space, conversion to compressed or static RFA file types, ordering of data by a particular key, and index packing. This utility has been optimized to attain the highest file performance. The **ismvfy** and **irecovr** utilities are front-ends to **isutl**.

The **fcompare** utility compares database files to a system catalog or repository.

The **chklock** utility (Windows and UNIX only) reports information about locks on a file.

The **ctutl** utility (Windows and UNIX only) controls change tracking parameters.

The most current version of select utilities may be available for download from the Downloads section of [resourcecenter.synergex.com](http://resourcecenter.synergex.com).

## Parameter and XDL files

The **bldism**, **isload**, and **status** utilities can accept input either typed directly by a user or from a parameter file that contains the input in the appropriate order. In addition, **fconvert** and **bldism** can accept input from a file that contains a valid FDL or XDL description. (See [“ISAM Definition Language” on page 3-89](#) for additional information about XDL files.) Parameter and XDL files can be useful when you’re creating an ISAM file or modifying the definitions of one; you simply modify or create the records in your file and then run that file through the utility.



If you’re creating a file from scratch for a new ISAM file, we recommend you create an XDL file rather than a parameter file. An XDL file is much easier to read and maintain, and the order doesn’t matter.

---

A parameter or XDL file can contain any of the file options available to **bldism** or the ISAMC subroutine (record type, compression, terabyte, density, and so on). For a list of these options, run **bldism** and enter “?” at the first prompt, or see [“File specification”](#) in the Discussion for ISAMC in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual*. For example, if you add “, TBYTE” to the file specification line (the first uncommented line) in a parameter file and then run the **fconvert** utility on that file, the file will be converted to a terabyte file. If conflicting qualifiers and settings are specified in a parameter file, the last one takes precedence.

You can also include comments and descriptions in your parameter or XDL files by preceding them with a semicolon; **fconvert** and **bldism** ignore all text following a semicolon.

To specify that input is from a parameter file, enter the name of the parameter file preceded by one or two “at” signs (@) at the first prompt displayed by the utility.

- ▶ If you enter one “at” sign, the utility program takes input directly from the parameter file and doesn’t prompt further.
- ▶ If you enter two “at” signs, the utility program takes input from the parameter file, but displays a trace of the activity. This trace consists of the usual prompts along with the responses read from the parameter file.

To specify that input is from an XDL file, enter the name of the XDL file preceded by one “at” sign (@) at the first prompt displayed by **bldism**. Input then comes directly from the XDL file, and **bldism** doesn’t prompt further.



Once you specify a parameter or XDL file, all subsequent input must come from that file. Any errors that usually require resolution by the user cause the utility program to terminate. Errors encountered from a parameter or XDL file are reported to you, but they cause abnormal termination of the utility program.

---

## Parameter file example

*Below is a parameter file named **cusmas.par** that we can use to create the ISAM file **cusmas.ism**, which we created using **bldism** in “Sample bldism” on page 3-39. Remember, any text starting with a semicolon is ignored.*

```
; Parameter file cusmas.par used to create cusmas.ism
cusmas.ism, variable, compress      ;ISAM filename
2000                                ;record size
4                                   ;number of keys
name/segmented                      ;primary key
                                   ;total key size 30
2                                   ;number of segments
15                                  ;length of segment #1
16                                  ;start position
15                                  ;length of segment #2
1                                   ;start position
n                                   ;duplicates allowed
a                                   ;ascending
company                             ;first alternate key
30                                  ;key size
31                                  ;start position
Y                                   ;duplicates allowed
Y                                   ;insert at front
a                                   ;ascending
address/segmented/modify            ;second alternate key
                                   ;total key size 40
3                                   ;number of segments
20                                  ;length of segment #1
61                                  ;start position
10                                  ;length of segment #2
51                                  ;start position
10                                  ;length of segment #3
91                                  ;start position
Y                                   ;duplicates allowed
n                                   ;insert at front
a                                   ;ascending
act_code/null                       ;third alternate key
r                                   ;replicating null key
32                                  ;null value
5                                   ;key size
101                                 ;start position
Y                                   ;duplicates allowed
Y                                   ;insert at front
a                                   ;ascending
```

*If we wanted to use this parameter file as input to **bldism**, we’d enter the filename preceded by one or two “at” signs at the first prompt, as follows:*



Enter name of the ISAM file to create: @cusmas.par

*In our example, **bldism** uses the parameter file **cusmas.par** as its input and creates the ISAM file **cusmas.ism** without displaying any further prompts.*



You can create a parameter or an XDL file that contains a description of an existing ISAM file using the **ipar** utility. See [ipar on page 3-68](#) for step-by-step instructions on using this utility program.

---

## WIN, UNIX

---

You can also redirect input from a parameter file as follows:

```
dbr DBLDIR:utility <filename
```

*utility*

One of the following utilities: **bldism**, **isload**, or **status**.

*filename*

The name of the parameter file from which you want to get your input.

*In our example, we can use the following command*

```
dbr DBLDIR:bldism <cusmas.par
```

*to create the ISAM file **cusmas.ism** using the parameter file **cusmas.par** shown in “[Parameter file example](#)” on page 3-36.*

---

## bldism – Create an ISAM file



The **bldism** utility enables you to create an ISAM file from outside your programs. It prompts you to specify the name and type of ISAM file you want to create, and to define the length and key structure of its records.

To run **bldism**,

On	Enter this at the command line
Windows and UNIX	<code>dbr DBLDIR:bldism [-k <i>filename</i>]</code>
OpenVMS	<code>run DBLDIR:bldism</code>  Or, if you want to specify the <b>-k</b> option, set <b>bldism</b> up as a foreign command and start it from a symbol: <code>\$ bldism:==\$DBLDIR:bldism</code> <code>\$ bldism -k <i>filename</i></code>

### Arguments

**-k *filename***

(optional) Specifies that **bldism** should not prompt for input but should instead create an ISAM file according to the XDL or FDL description in the specified file. *Filename* must be a specification for a file that contains a valid XDL or FDL description.

### Discussion

To find out what the valid input is at any prompt, enter a question mark (?). To terminate **bldism** at any time, type the end-of-file character for your operating system.



On OpenVMS, generating an FDL with the “EDIT/FDL” system command optimizes files.



In Synergy .NET and 64-bit traditional Synergy, if you create a file with a variable record type and a maximum record size of 0, the records in that file can exceed 64K and are limited only by the amount of memory you have available or 2 GB.

## Sample bldism

Enter name of the ISAM file to create: **cusmas, v, c**

What is the length of data records? (1-65534) **2000**

How many different keys? (1-255) **5**

Primary key:

Enter name of the Key field: **name/segment/type/order**

How many different segments? (1-8) **2**

What type is segment #1? (A, D, I, U, or N [/ALL]) **A/ALL**

How long is segment #1? (1-99) **15**

Where does segment #1 start? (1-1986) **16**

Segment order to be ascending or descending? (A/D) **D**

How long is segment #2? (1-85) **15**

Where does segment #2 start? (1-1986) **1**

Segment order to be ascending or descending? (A/D) **D**

Are duplicate keys to be permitted? (Y/N) **N**

1st alternate:

Enter name of the Key field: **company/type/density**

What type is the key? (A/D/I/U/N/S/T/C) **A**

How long is the key? (1-255) **30**

Where does the key start? (1-1971) **31**

Are duplicate keys to be permitted? (Y/N) **Y**

Insert duplicates at front? (Y/N) **Y**

Key order to be ascending or descending? (A/D) **A**

What is the key packing density? (50-100) **75**

2nd alternate:

Enter name of the Key field: **address/segment/modify/density**

How many different segments? (1-8) **3**

How long is segment #1? (1-98) **20**

```
Where does segment #1 start? (1-1981) 61
How long is segment #2? (1-79) 10
Where does segment #2 start? (1-1991) 51
How long is segment #3? (1-70) 10
Where does segment #3 start? (1-1991) 91
Are duplicate keys to be permitted? (Y/N) Y
Insert duplicates at front? (Y/N) N
Key order to be ascending or descending? (A/D) A
What is the key packing density? (50-100) 70
```

3rd alternate:

```
Enter name of the Key field: act_code/null
Replicating, Non-replicating, or Short null key? (R/N/S) R
Null value: 32
How long is the key? (1-255) 5
Where does the key start? (1-1986) 101
Are duplicate keys to be permitted? (Y/N) Y
Insert duplicates at front? (Y/N) Y
Key order to be ascending or descending? (A/D) A
```

4th alternate:

```
Enter name of the Key field: cust_number/density/type
What type is the key? (A/D/I/U/N/S/T/C) D
How long is the key? (1-255) 10
Where does the key start? (1-1986) 120
Are duplicate keys to be permitted? (Y/N) N
Key order to be ascending or descending? (A/D) A
What is the key packing density? (50-100) 90
```

ISAM file successfully initialized

## Running the bldism utility

To illustrate how to use **bldism** to create an ISAM file, let's assume we want to create an ISAM file that stores customer information, such as customer name, company, and address. The **bldism** utility prompts us as follows for the information needed to create our ISAM file. (The example to which we refer throughout this section is found in *"Sample bldism"* on page 3-39.)

### File prompts

- **Enter name of the ISAM file to create:** Enter the name of the ISAM file you want to create as follows:

```
filename[, record_type][, COMPRESS][, DENSITY=file_density][, I=pos:len[, ...]]
[, ERASE_ON_DELETE][, NETWORK_ENCRYPT][, PAGE=page_size][, [NO]ROLLBACK]
[, SGRFA][, SIZE_LIMIT=max_size][, RECORD_LIMIT=recs][, STATIC_RFA][, [NO]TBYTE]
[, TEXT=text_spec][, TRACK_CHANGES]
```

*filename*

The name of the ISAM file you want to create. The default extension is **.ism**.

*record\_type*

(optional) One of the following types of ISAM files:

<b>fixed</b>	Fixed-length records (default)
<b>multiple</b>	Multiple fixed-length records (up to 32 record lengths per file)
<b>variable</b>	Variable-length records

### COMPRESS

(optional) Compresses the data of the specified ISAM file. You cannot specify this option with a record type of **multiple**.

### DENSITY

(optional) Indicates that the file density follows.

*file\_density*

A number between 50 and 100 that represents the default density percentage for each key in the file, which is the percentage each index block is filled. (See *"ISAM index density"* on page 3-8 for more information and suggestions about setting the file density.)

### I

(optional) Indicates that a portable integer definition follows. You can specify this option more than once to define up to 255 portable integers per file.

*pos*

The starting position of nonkey integer data.

*len*

The length of nonkey integer data. The following values are valid:

1  
2  
4  
8

### **ERASE\_ON\_DELETE**

(optional) Indicates that when a record is deleted, it will be erased (i.e., nulled out). If ERASE\_ON\_DELETE is not specified, records that are deleted are marked deleted, but their contents remain in the file until the space is physically reused.

ERASE\_ON\_DELETE cannot be used if TRACK\_CHANGES is set.

### **NETWORK\_ENCRYPT**

(optional) Sets the network encryption flag on the new file, which ensures that any client accessing that file must use encryption. (See [“Specifying the data to encrypt for slave encryption”](#) in the “Configuring xfServer” chapter of the *Installation Configuration Guide* for more information.)

### **PAGE**

(optional) Indicates that a specific page size follows.

*page\_size*

The size of each index block in the ISAM file. The following values are valid:

512  
1024  
2048  
4096 (default)  
8192  
16384  
32768

(See [“Page size” on page 3-9](#) for more information and page size suggestions about setting the page size.)

### **/NO/ROLLBACK**

(optional) Control the rollback function. ROLLBACK (default) allows it, and NOROLLBACK prohibits it.

### **SGRFA**

(optional) Generate and store the CRC-32 part of an RFA to each record header on each STORE or WRITE.

### **SIZE\_LIMIT**

(optional) Indicates that the file size limit follows.

*max\_size*

The maximum number of megabytes that the **.is1** file is allowed to reach. If a STORE or WRITE operation exceeds this limit, an “Output file is full” error (FILFUL) occurs.

## RECORD\_LIMIT

(optional) Indicates that the record limit follows.

*recs*

The maximum number of records allowed in the **.is1** file. If a STORE operation exceeds this limit, a “File record limit exceeded” error (RECLIMIT) occurs.

## STATIC\_RFA

(optional) Ignored on REV 6 and higher files because it is set automatically. Ensures that a record retains the same RFA across WRITE operations. Static RFA files aren’t fully self-reorganizing.

## /NO/TBYTE

(optional) TBYTE creates a file capable of holding 256 terabytes of data or index. NOTBYTE prohibits an implicit TBYTE setting.

## TEXT

(optional) Add the specified text to the header of the file being created and/or allocate space for user-defined text.

*text\_spec*

A text allocation size and/or text string. See [“Specifying user-defined text in the file data header”](#) in the ISAMC Discussion in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual* for detailed syntax.

## TRACK\_CHANGES

(optional) Enable change tracking. Requires a REV 6 or greater file. Specifying TRACK\_CHANGES also enables TBYTE automatically (unless NOTBYTE is specified). You cannot specify this option with a record type of **multiple**.



You can abbreviate any of the above options. However, to eliminate confusion with future additions of options and for program clarity, you can also specify the full option name. See [“ISAM file types” on page 3-7](#) for a discussion about the different ISAM file types. Also see [“Static RFAs”](#) in the “Synergy DBL Statements” chapter of the *Synergy DBL Language Reference Manual* for more information about static RFAs and data compression.

---

*In our example, we entered*

```
cusmas, v, c
```

*at this prompt to create an ISAM file named **cusmas.ism** that has variable-length records and compressed data.*

- ▶ **What is the length of data records? (1 - 65534)** Enter the expected maximum size of any data record (up to 65,534 characters long on Windows and UNIX or 32,234 characters on OpenVMS). The minimum record size is 4. To determine the record size, remember to include the key field, but don't include space for record terminators; no record terminators are stored as part of an ISAM file.

*In our example, we entered **2000** at this prompt; therefore, the maximum size of any data record within our ISAM file, **cusmas.ism**, is 2000 characters long.*

- ▶ **How many different keys? (1 - 255)** Enter the number of keys you want to specify (up to 255 keys). A separate index is created for each key, thereby enabling keyed access to records using any one of the keys. Updating an ISAM file, however, modifies each key index. Remember that for each key you specify, you increase update processing time and increase disk space usage.

*In our example, we entered 4 at this prompt to specify four keys for our **cusmas.ism** file.*

## Key prompts

- ▶ **Primary key:**

**Enter name of the Key field:** Enter a name for the primary key field of the specified ISAM file, as follows:

*[key\_name][ /SEGMENT ][ /MODIFY ][ /NULL ][ /TYPE ][ /ORDER ][ /DENSITY ]*

*key\_name*

(optional) An alpha expression that represents the name of the key and is used in key-of-reference specifications. The maximum key name length, including quotes, is 15 for Synergy ISAM. (RMS ISAM has no maximum length, but because the ISAMC subroutine accepts a maximum of 32 characters, your key name shouldn't be any longer than 32 characters.)

*/SEGMENT*

(optional) Specifies that the key is segmented.

*/MODIFY*

(optional) Specifies that the key field is modifiable.

*/NULL*

(optional) Specifies that the key is a null key.

*/TYPE*

(optional) Indicates that you want to define a specific key type for each segment or for all segments.



## /ORDER

(optional) Indicates that you want to define a specific key order for each segment. To assign the same order to all segments, don't specify this option as part of the key specification; **bldism** prompts for the key order at the end of the key definition.

## /DENSITY

(optional) Indicates that you want to define a specific density for this key.

You can abbreviate any of the above options; for example, /S for /SEGMENT, /M for /MODIFY, and /N for /NULL.



You can't specify the /MODIFY or /NULL options on the primary key. See [“Keys in ISAM files” on page 3-13](#) for more details about segmented, modifiable, and null keys.

*In our example, we entered*

`name/segment/type/order`

*at this prompt to specify a segmented primary key called **name**.*

- ▶ **Replicating, Non-replicating, or Short null key? (R/N/S)** If you specified /NULL at the previous prompt, enter **R** if you want the key to be a replicating null key, **N** if you want it to be a nonreplicating null key, or **S** if you want it to be a short null key. If the key is not a null key, this prompt doesn't appear.

*In our example, this prompt does not appear, because our primary key is not a null key.*

- ▶ **Null value:** Enter the value of the null key. If you specified a replicating null key, enter a value representing a single character. If you specified a nonreplicating null key, enter a string as the null value. If you specified a short null key or if this key is not a null key, this prompt doesn't appear.

*In our example, this prompt does not appear, because our primary key is not a null key.*

- ▶ **How many different segments? (1-8)** Enter the number of segments you want assigned to this key. You can specify up to eight segments. If this is not a segmented key, this prompt doesn't appear.

*In our example, we entered 2 at this prompt to assign two segments to the primary key, **name**.*

- ▶ **What type is segment #1? (A, D, I, U, or N[/ALL])** This prompt is displayed if you specified /TYPE at the “Enter name of the Key field” prompt. Enter **A** for alpha, **D** for decimal, **I** for integer, **U** for unsigned integer, or **N** for case-insensitive alpha. If you want to define the rest of the segments as having the same type as the first without being prompted again, type /ALL immediately following the A, D, I, U, or N.

*In our example, we entered A/ALL at this prompt to specify that the type for both segments of our primary key is alpha.*

- **How long is segment #1? (1-*nnn*)** Enter the length of the first segment. The value *nnn* is displayed as **255** or as the difference between 255 and the previously defined record length, whichever is smaller. The length of the segment must not be greater than *nnn*.



An overall key length of 255 is only valid for OpenVMS files. A warning is displayed if the overall key length exceeds 254 characters when **bldism** is run on a Windows or UNIX machine. An error is generated at creation time if the destination of a file being created with a key that exceeds 254 characters is Windows or UNIX. (The term *overall key length* refers to the combined segment length if the key is segmented or the length of the key if the key is not segmented, plus 3 bytes on Windows or UNIX if the key allows duplicates, or 4 bytes for terabyte files.)

---

*In our example, we entered **15** at this prompt to specify that the length of the first segment of our primary key is 15 characters long.*

This prompt is repeated for each segment assigned to this key. If this is not a segmented key, however, the following prompt appears:

**How long is the key? (1-*nnn*)**

to which you should respond with the length of the key field. Again, the length must not be greater than *nnn*. (See the note above.)

- **Where does segment #1 start? (1-*nnn*)** Enter the starting character position of the segment. The value *nnn* is calculated from the defined record and segment lengths. The starting position must not be larger than *nnn*.

*In our example, we entered **16** at this prompt to specify that the first segment of our primary key starts at position 16.*

This prompt is repeated for each segment assigned to this key. If this is not a segmented key, however, the following prompt appears:

**Where does the key start? (1-*nnn*)**

to which you should respond with the starting character position of the key field. Again, the starting position must not be larger than *nnn*.

- **Segment order to be ascending or descending? (A/D)** This prompt is displayed if you specified **/ORDER** at the “Enter name of the Key field” prompt. Enter **A** to force the segment into an ascending order or **D** to force it into a descending order. If you enter **A**, the **READS** statement sequentially retrieves records starting with the lowest segment value and progressing to the highest segment value. If you enter **D**, **READS** retrieves records from the highest to the lowest segment values. Note that the order of segment values is based on the eight-bit ASCII collating sequence.

*In our example, we entered **D** at this prompt to specify that the first segment of our primary key should be sorted in descending order.*

See “Ascending or descending keys” on page 3-18 for more information.

- ▶ **Are duplicate keys to be permitted? (Y/N)** Enter **Y** if you expect the file to contain multiple records having the same key value. Enter **N** to ensure that no two records will ever have the same key value. If you enter **N**, the STORE statement signals a “Duplicate key specified” error (SERR\_NODUPS) each time you attempt to store a record having a key value that is already present in the file.

*In our example, we entered **N** at this prompt to specify that the primary key should not allow duplicate keys.*

See “[Duplicate keys](#)” on page 3-15 for more information about duplicate keys.

- ▶ **Insert duplicates at front? (Y/N)** This prompt appears only if you are allowing duplicate keys for this key field. Enter **Y** if you want to insert duplicate keys at the front of records possessing the same key value. Enter **N** if you want to insert duplicates at the end of such records. If you enter **Y**, duplicate records are retrieved in last-in-first-out (LIFO) order. If you enter **N**, duplicate records are retrieved in first-in-first-out (FIFO) order.

*In our example, this prompt does not appear, because we did not allow duplicate keys for this field.*

---

## VMS

You must answer **N** at this prompt; OpenVMS only allows duplicates to be inserted at the end of a list of matching records.

---

- ▶ **Key order to be ascending or descending? (A/D)** Enter **A** to force keys into an ascending order or **D** to force them into a descending order.

*In our example, this prompt does not appear, because we specified **/ORDER** in the key specification and we were therefore prompted for the order of each segment individually.*

See “[Ascending or descending keys](#)” on page 3-18 for more information about ascending and descending keys.

- ▶ **What is the key packing density? (50-100)** This prompt is displayed if you specified **/DENSITY** at the “Enter name of the Key field” prompt. Enter a value between 50 and 100 to specify the density percentage for this key.

*In our example, this prompt does not appear, because we did not specify **/DENSITY** in the specification for the primary key.*

- ▶ **1st - 255th alternates:** The **bldism** utility repeats the preceding key prompts for as many keys as you specified for the ISAM file you want to create.

*In our example, **bldism** prompted us to name and define four key fields (the primary key and three alternate keys) because we specified four keys for our **cusmas.ism** file.*

## chklock – Report file lock information

WT	WN	U	
----	----	---	--

The **chklock** utility reports information about locks on a file. To run the utility, type the following at the command line:

`chklock [-options] filename`

### Arguments

#### UNIX

---

##### *options*

(optional) One or more of the following options:

- p**        Display the process ID that holds each lock, the byte position of each lock, and the length (number of bytes) of each lock.
- r**        Display one of the following values:
  - WT\_SHARE**        One or more processes have the file opened in update mode.
  - RD\_SHARE**        One or more processes have the file opened in input mode.
- v**        Display both types of file locks on Oracle Solaris. (See [system option #33](#) in the “System Options” chapter of *Environment Variables & System Options*.)

---

##### *filename*

The name of the file for which you want to retrieve locking information. For ISAM files, this must be the name of the data file (usually *filename.is1*).

### Discussion

For ISAM files, Synergy DBL locks the first byte of the record in the data file. For other file types, Synergy DBL locks the whole record.

If you run **chklock** without any options, it reports the position of the first byte of each locked record in the file.

If a process has opened a file in exclusive mode (SHARE:0), **chklock** returns the message “File locked.” On UNIX, the **-p** option indicates the process ID that has it locked. (See the last example below.)

## Examples

In the first example below, two records are locked. One starts at byte 1240 (in the ISAM data file), while the other starts at byte 1535. In the second example, one record is locked starting at byte 129.

```
$ chklock myfile.is1
1240 1535

$ chklock myfile.ddf
129
```

In the following example, one record in the file is locked by process ID number 3209. It starts at byte 0 and it is 129 bytes long.

```
$ chklock -p myfile.ddf
3209: 0 - 129
```

The following example indicates that at least one process has opened the file in update mode.

```
$ chklock -pr myfile.ddf
3209: 0 - 129
WT_SHARE
```

In the following example, the file was opened in exclusive mode by process ID number 3209.

```
$ chklock -p myfile.ddf
3209: File locked
```

## ctutl – Manipulate change tracking parameters

WT	WN	U	
----	----	---	--

`ctutl -option [-other_options] filename`

### Arguments

#### *option*

One of the following change tracking commands:

- a** Enable change tracking for the specified files. This option can be combined with **-n**. You cannot specify this option with a record type of **multiple**.
- f [n]** Free the oldest snapshot. If *n* is specified and is positive, free the specified number of snapshots, starting with the oldest. (For example, **ctutl -f 3** will release the three oldest snapshots.) If *n* is specified and is negative, *keep* the specified number of snapshots. (For example, **ctutl -f -1** will release all snapshots except the most recent one.)
- n** Enable the NOROLLBACK file option. This option can be combined with **-a**.
- r [n]** Roll back the most recently changed records, up to but not including the most recent snapshot in the specified file(s). If *n* is specified, roll back the most recent changes up to and including the *n*th most recent snapshot.
- s** Start a new change tracking snapshot.

#### *other\_options*

(optional) One or more of the following options:

- h** Display help screen.
- mlevel#** Specify a message level that defines the amount of information displayed during an operation, where *level* is a value from 0 to 2.
- v** Show snapshot details for the specified file. (default)
- %** Display a running status (0 to 100) to indicate the percentage completed by the operation.

#### *filename*

The ISAM file(s) for which change tracking is desired. Use a wildcard character to designate multiple files. An xfServer remote file specification is not supported.

## Discussion

The **ctutl** utility is used to manage the change tracking feature in one or more ISAM files. (See [“Change tracking” on page 3-10](#) for an overview of change tracking.) For files without change tracking enabled, **ctutl** can be used to add the feature. For files with change tracking enabled, you can use **ctutl** to synchronize new snapshots, free old snapshots, roll back recent snapshots, and view current snapshots.



ISAM files must be at Revision 6 in order to use change tracking capabilities.

---

To synchronize adding snapshots to more than one file, we recommend you use the **synbackup** utility to freeze and then restore I/O. See [“The Synbackup Utility” on page 4-34](#) for more information.

We recommend using **ctutl** as follows:

1. Run **ctutl -a** to initiate transparent recording of file changes to existing files. For example,

```
ctutl -a order*.ism
```

With new files, you can skip this step and immediately start recording changes by specifying the **TRACK\_CHANGES** file option when creating the file.)

After the application runs for some amount of time,

2. (optional) Run **synbackup -s** to temporarily freeze all Synergy I/O.
3. Run **ctutl -s** on the specified files to start a new change tracking snapshot. For example,

```
ctutl -s order*.ism
```

You may want to perform this task as a cron job or scheduled task.

4. If you froze Synergy I/O in [step 2](#), run **synbackup -x** to restore I/O.
5. Immediately after [step 3](#) or [step 4](#), or at some predetermined interval, run **ctutl -f** to release the oldest snapshot(s). For example,

```
ctutl -f order*.ism
```

If you don't release the snapshots frequently enough, the designated files may become excessively large, which leads to poor performance. If this occurs, increase the frequency at which you run **ctutl -f**.

Snapshots continue to grow numerically until the file is cleared, so it's possible that after some time you could be dealing with snapshot #500 or even #1000.

A change tracking snapshot (created with **ctutl -s**) includes the current date and time to a one second granularity. The snapshot number is set to the current snapshot ID, beginning at 0 and incremented for each successive snapshot. Each snapshot is optimized to keep only the most recent net change of any one record.

In the event you need to roll back changes to a previous snapshot, you can use **ctutl -r**. For example,

```
ctutl -r order*.ism
```

If you want to roll changes back two snapshots, keeping the prior snapshot and the time it was applied, you'd do the following:

```
ctutl -r 1 myorders
ctutl -r
```

The first line rolls back all changes made after the last applied snapshot and removes that snapshot, and the second line rolls back all changes made between that snapshot and the one preceding it without removing it.

If the NOROLLBACK file option is enabled and you attempt to roll back changes using **ctutl -r**, the operation will fail with a “Rollback is not permitted” error.

The **-n** option must either be used in conjunction with **-a** or the file must first be configured with Change Tracking (if **-a** is not specified). Otherwise, the utility will abort with the error “Change Tracking required - operation ignored.” If **-n** is specified when the NOROLLBACK option is already enabled, the utility will abort with a “NOROLLBACK option already applied - operation ignored.”

### Using change tracking for end-of-period processing

One of the most common reasons to take advantage of change tracking is to handle processing at the end of a period. The following procedure shows how you might use change tracking features for that purpose.

1. To begin logging changes following prior end-of-period processing, freeze all Synergy I/O with **synbackup**, apply a change tracking snapshot, and then restore Synergy I/O with **synbackup**:

```
$ synbackup -s
$ ctutl -s *.ism
$ synbackup -x
```

Your end users can perform their normal daily activities, as file updates to designated files are being recorded transparently.

2. At the next end-of-period, repeat the commands in [step 1](#) to apply another change tracking snapshot. Further user updates occur but do not affect period-end-processing.

```
$ synbackup -s
$ ctutl -s *.ism
$ synbackup -x
```

3. Add application code to locate all net changes made to key files between two snapshots. Use the NetChange method in the Select class to make an appropriate selection, and then after the selection is made, identify the change tracking status with the CTState enumerator. To programmatically identify the most recent snapshots, rather than keeping track of snapshot numbers, consider using



relative snapshot numbering, where -1 is the most recently applied snapshot, -2 is the snapshot before that, etc. When specifying relative snapshots, use 0 to represent the current point, which means all changes made after the last snapshot. For example,

```
foreach rec in new Select (from, Where.Netchange(snapshot1, snapshot2))
begin
    type = Select.GetEnum().GetCTInfo.CTState
    ; Processing
end
```

4. Apply the change record as required for the function of the operation. This may be a call to update a SQL database, for example.
5. As part of post processing, use **ctutl** to release the oldest set of change tracking information from each file processed at the successful completion of the operation. By doing this regularly, the amount of unused excess in the file is reduced.

```
$ ctutl -f file
```

### Interpreting the -v option output

Even before any snapshots have been applied, a file will have one active “snapshot,” or beginning point, with 0 change entries.

After a few snapshots have been applied, output from using the **-v** option looks something like this:

```
Change tracking information:
  There are 4 snapshots currently active
  (They are ordered oldest to most recent)
  (Change entries (Insert/Update/Delete/Ulink) occur between snapshots as shown)

Beginning (#0) Fri Jun 08 14:50:00 2012
  5 change entries (5/0/0/0)
Snapshot (#1) Sat Jun 09 14:50:00 2012
  3 change entries (0/2/1/0)
Snapshot (#2) Sun Jun 10 14:50:00 2012
  6 change entries (5/1/0/0)
Snapshot (#3) Mon Jun 11 14:50:00 2012
  5 change entries (5/0/0/0)
```

The numbers separated by slashes (for example, “(5/0/0/0)”) refer to Insert, Update, Delete, and Ulink recorded entries, respectively. Ulink is a special entry that records any change that isn’t represented by a prior change entry. For example, if you apply change tracking to an existing file of records using **ctutl -a**, the existing records have not previously been tracked, and any change such as Write or Delete will generate a Ulink entry along with a Write or Delete entry.

## Examples

### Example 1

The example below illustrates the effect of rolling back changes.

Running **ctutl -v myfile** generates the following:

Change tracking information:

```
There are 4 snapshots currently active
(They are ordered oldest to most recent)
(Change entries (Insert/Update/Delete/Ulink) occur between snapshots as shown)
```

```
Beginning (#0) Fri Jun 08 15:03:15 2012
  5 change entries (5/0/0/0)
Snapshot (#1) Sat Jun 09 15:03:16 2012
  3 change entries (0/2/1/0)
Snapshot (#2) Sun Jun 10 15:03:17 2012
  6 change entries (5/1/0/0)
Snapshot (#3) Mon Jun 11 15:03:18 2012
  5 change entries (5/0/0/0)
```

Then, running **ctutl -r1 myfile** generates the following:

Deleting 1 snapshot and rolling back the following changes:

```
Snapshot (#3) Mon Jun 11 15:03:18 2012
  5 current changes
```

Deleting snapshot (#3)

Another **ctutl -v myfile** generates the following. (Notice that #3 and the entries that follow are gone.)

Change tracking information:

```
There are 3 snapshots currently active
(They are ordered oldest to most recent)
(Change entries (Insert/Update/Delete/Ulink) occur between snapshots as shown)
```

```
Beginning (#0) Fri Jun 08 15:03:15 2012
  5 change entries (5/0/0/0)
Snapshot (#1) Sat Jun 09 15:03:16 2012
  3 change entries (0/2/1/0)
Snapshot (#2) Sun Jun 10 15:03:17 2012
  6 change entries (5/1/0/0)
```

Then, **ctutl -r myfile** generates the following:

Rolling back 6 changes to snapshot (#2) Fri Jun 10 15:03:17 2012

Running **ctutl -v myfile** again generates the following. (Notice that the most recent changes after #3 are gone, but the snapshot remains.)

Change tracking information:

```
There are 3 snapshots currently active
(They are ordered oldest to most recent)
(Change entries (Insert/Update/Delete/Ulink) occur between snapshots as shown)
```

```
Beginning (#0) Fri Jun 08 15:03:15 2012
  5 change entries (5/0/0/0)
Snapshot (#1) Sat Jun 09 15:03:16 2012
  3 change entries (0/2/1/0)
Snapshot (#2) Sun Jun 10 15:03:17 2012
  0 change entries (0/0/0/0)
```

## Example 2

The example below illustrates the effect of freeing snapshots. (Notice that running **ctutl** with no options defaults to the **-v** option.)

Running **ctutl test** generates the following:

Change tracking information:

```
There are 4 snapshots currently active
(They are ordered oldest to most recent)
(Change entries (Insert/Update/Delete/Ulink) occur between snapshots as shown)
```

```
Beginning (#0) Tue Jun 12 10:10:38 2012
  5 change entries (5/0/0/0)
Snapshot (#1) Wed Jun 13 10:10:39 2012
  3 change entries (0/2/1/0)
Snapshot (#2) Thu Jun 14 10:10:40 2012
  6 change entries (5/1/0/0)
Snapshot (#3) Fri Jun 15 10:10:41 2012
  5 change entries (5/0/0/0)
```

Next, running **ctutl -f 2 test** to free the two oldest snapshots generates the following:

Deleting 2 snapshots and freeing the following change entries:

```
Beginning (#0) Tue Jun 12 10:10:38 2012
  5 snapshot (#1) change entries
Snapshot (#1) Wed Jun 13 10:10:39 2012
  3 snapshot (#2) change entries
Deleting snapshot (#0)
Deleting snapshot (#1)
```

Running **ctutl test** again generates the following. (Notice the first two snapshots are gone, and the beginning snapshot now starts at #2.)

Change tracking information:

```
There are 2 snapshots currently active
(They are ordered oldest to most recent)
(Change entries (Insert/Update/Delete/Ulink) occur between snapshots as shown)
```

```
Beginning (#2) Thu Jun 14 10:10:40 2012
  7 change entries (5/1/0/1)
Snapshot (#3) Fri Jun 15 10:10:41 2012
  5 change entries (5/0/0/0)
```

## fcompare – Compare database files to system catalog or repository



`fcompare [-system_catalog_options] [-repository_options] [-output_options]`

### Arguments

#### *system\_catalog\_options*

(optional) One or more of the following options, which cause **fcompare** to compare database file definitions with system catalog definitions.

**g** *connect\_file* Specify the name and path of the connect file.

**t** *table* Specify a system catalog table name to check.

#### *repository\_options*

(optional) One or more of the following options, which cause **fcompare** to compare database file definitions with repository file definitions.

**r** *rpsmain rpstext* Specify the repository main and text files to use when comparing repository metadata against a database file.

**f** *file\_def\_name* Specify a specific repository file definition name to check. **Fcompare** checks all structures assigned to the file described by that file definition.

**c** *convert\_setup\_file* Override the conversion setup file specified by the SODBC\_CNVFIL environment variable, and specify the name and path of the file to use in its place.

#### *output\_options*

(optional) One or more of the following options:

**dv** Turn on data verification mode to compare ISAM file data with system catalog or repository metadata and generate verification output. This option can only be used if **-t** or **-f** is also being used.

**l** *log\_file* Specify the name of a log file that will contain the output from the **fcompare** program.

**i** Generate error, warning, and informational messages. (Do not use with **-v**.)

**v** Generate error and warning messages. (Do not use with **-i**.)

## Discussion

The Synergy File Compare utility (**fcompare**) can be used to debug synchronization problems between repository or system catalog metadata and an ISAM, RMS, relative, or text file definition. It can also compare ISAM, RMS, relative, or text file *data* against repository or system catalog metadata. **Fcompare** does not compare metadata between a repository and a system catalog. For relative files, **fcompare** verifies record size and number of records only.

When discrepancies are found, **fcompare** produces either an error or a warning (if **-v** is specified). An error or warning indicates that the specified attribute, as defined in the repository or system catalogs, doesn't match the "actual" attribute of the database file. Both the defined and the actual values are displayed, along with the name of the attribute being compared. Error messages are designed to assist xfODBC users. If you are using **fcompare** to compare file definitions to a repository, you should use the **-v** option to also output warning messages. See ["Errors and warnings" on page 3-59](#) for a list of possible discrepancies.

---

### VMS

The **fcompare** utility is set up as a verb, which means you cannot pass more than eight parameters. Each option counts as one parameter, and each path specification counts as one parameter. If you have more than eight parameters, you must work around the limitation by enclosing the entire set of parameters in double quotation marks.

For example:

```
fcompare "-r RPSDAT:rpsmain.ism RPSDAT:rpstext.ism -f customer -dv  
-l compare.log -v"
```

---

If neither **-g** nor **-r** is specified, **fcompare** compares repository definitions with database files. If both *system\_catalog\_options* and *repository\_options* are specified, an error message is generated and processing is terminated.

### System catalog comparisons

You can perform a comparison of all tables in the system catalog by specifying **-g** without **-t**, or you can limit the comparison to a single table by using the **-t** option. With the single table comparison, you can request that the data in the database file be verified against the catalog definitions by using the **-dv** option.

We recommend that ODBC users run **fcompare** using the repository option first, so that any discrepancies can be resolved before the system catalogs are generated.

For **fcompare** to access the database files for the system catalog option, any logicals used must be defined in the connect file or in the environment. If *connect\_file* doesn't include a path, **fcompare** looks for the file in the directory specified by the GENESIS\_HOME environment variable.

## Repository comparisons

You can perform a comparison of all file definitions in the repository by specifying **-r** without **-f**, or you can limit the comparison to a single file by using the **-f** option. (Keep in mind that **-f** expects a specific repository file definition name, not the repository “open filename.”) With the single file comparison, you can request that the data in the database file be verified against the definitions by using the **-dv** option.

If **-r** is not specified (assuming **-g** is not specified either), **fcompare** uses the environment variables RPSMFIL and RPSTFIL. If they are not set, **fcompare** looks for **rpsmain.ism** and **rpstext.ism** in the directory specified by the environment variable RPSDAT.

**Fcompare** reads data logicals from the environment, the **synergy.ini** file, or an environment setup file whose name and location are specified by the SODBC\_INIFIL environment variable. Any logicals used in the repository “open filename” must be defined in one of these places.

If a conversion setup file is being used to specify filenames during conversion to system catalogs, **fcompare** can read the conversion setup file. When using the repository option (**-r**), set the SODBC\_CNVFIL environment variable to the location and name of the conversion setup file, and **fcompare** will read the filenames from there. To use a different setup conversion file than that specified by SODBC\_CNVFIL, use the **-c** option and specify the path and file you want to use.

To eliminate the errors detected by **fcompare**, you must examine the repository and the database file definition to resolve the discrepancies. The **ipar** utility (see [ipar on page 3-68](#)) can help you view the definition of the database file. (On OpenVMS, you can use the Analyze Utility.)



If you define a key as having two segments in the ISAM file but only set up the first segment in your repository, **fcompare** will not report an error, just a warning.

---

## Output options

The data verification option (**-dv**) must be used in combination with **-t** or **-f**. When data verification mode is on, **fcompare** reads through all records in the database file and verifies that date and decimal fields contain valid values for their field types—in other words, that the date fields contain valid dates and the decimal fields contain numbers. (The SYNCENTURY environment variable is used to determine the default century for two-digit years.) Using the **-dv** option significantly increases the amount of time it takes to run **fcompare**, which is why its usage is limited to one file or table at a time.



If the file contains tag definitions, data verification is skipped.

---

Output goes to the console unless the **-l** option is specified.

Informational messages (for example, the record size and number of keys) are only displayed if you specify the **-i** option. Warning messages are displayed if you specify **-i** or **-v**. Error messages are displayed in all cases, even if no options are specified.

## Errors and warnings

Errors indicate fixes necessary to prevent potentially incorrect data from being returned from xfODBC. Warnings identify fixes required to prevent performance problems due to loss of optimization opportunities. Synergex recommends that all errors and warnings be fixed. The Synergy/DE Developer Support department will require that errors be fixed before providing assistance with xfODBC optimization.

If **fcompare** finds discrepancies between repository or system catalog metadata and the ISAM, RMS, relative, or text file definition, it generates one or more of the following errors to indicate which attributes do not match and what the unmatched values are:

Error message	Description
Cannot open file	The database file for this table cannot be located.
Cannot retrieve information	There is a problem retrieving column, index, or tag information from the system catalogs. Contact Synergy/DE Developer Support.
Cannot retrieve key information for krf	The defined krf number doesn't exist in the file.
Collation (seg <i>n</i> ) defined as [ <i>x</i> ], actual [ <i>x</i> ]	The sort direction (ascending/descending) of the key segment does not match.
Defined date length is not [ <i>n</i> ]	With the <b>-dv</b> option in use, the data length defined for the field does not match the user-defined date specification.
Defined record length [ <i>n</i> ], bytes read [ <i>n</i> ]	With the <b>-dv</b> option in use, a read of a variable-length record exceeds the defined record size.
Defined segment collation does not match actual segment collation	Fix the key definition.
Defined segment positions do not match actual segment positions	Fix the key definition.
Duplicates defined as [ <i>x</i> ], actual [ <i>x</i> ]	A key is defined as "unique," but the file is not using a unique key.
Foreign key	All access keys must be defined before any foreign keys.
Hour field [ <i>n</i> ] is greater than 23	With the <b>-dv</b> option in use, a time field has an invalid hours value.

Error message	Description
Invalid date field value	With the <b>-dv</b> option in use, a date field contains an invalid value.
Invalid decimal field value	With the <b>-dv</b> option in use, a decimal field contains an invalid character.
Invalid relative file	The relative file record size does not match.
Key length (seg <i>n</i> ) defined as [ <i>x</i> ], actual [ <i>x</i> ]	The length of the field used as the key segment is larger than the key on the file.
Key of reference number <i>n</i> used more than once	The key of reference number is not unique.
Minute field [ <i>n</i> ] is greater than 59	With the <b>-dv</b> option in use, a time field has an invalid minutes value.
Non-numerical data in date field	With the <b>-dv</b> option in use, a date field contains nonnumeric data.
Non-numerical data in time field	With the <b>-dv</b> option in use, a time field contains nonnumeric data.
Null defined as [ <i>x</i> ], actual [ <i>x</i> ] <sup>a</sup>	For keys defined to allow null, the replication type (replicating, nonreplicating, or short) does not match.
Number of access keys defined as [ <i>x</i> ], actual [ <i>x</i> ]	More access keys are defined in the repository than on the physical file. (Foreign keys are not included in the count.)
Offset (seg <i>n</i> ) defined as [ <i>x</i> ], actual [ <i>x</i> ]	The starting location of the field used as the key segment does not match.
Record size	The length of fixed-length records does not match. Variable-length records are only checked with the <b>-dv</b> option.
Repository file not found <sup>a</sup>	<b>Fcompare</b> could not find one or more repository files. See <a href="#">“Repository comparisons” on page 3-58</a> .
Unsigned field [ <i>x</i> ] contains signed data	With the <b>-dv</b> option and either <b>-r</b> or <b>-g</b> in use, a field that is designated as unsigned contains signed data.

a. With *repository\_options* only.



If a verbose option (**-v** or **-i**) is specified, one or more of the following warnings may be generated:

Warning message	Description
Data type (seg <i>n</i> ) defined as [ <i>x</i> ], actual [ <i>x</i> ]	The data type of the field used as the key segment does not match, or a signed decimal field without a positive range is defined in the repository when the actual key in the ISAM data file is defined as alpha.
Duplicates defined as [ <i>x</i> ], actual [ <i>x</i> ]	The key is defined with duplicates. Better performance occurs if the repository is changed to say duplicates not allowed, as the file does.
Key <i>n</i> , Key length defined as [ <i>n</i> ]	A key is defined smaller than the file's physical key.
Modifiable defined as [ <i>x</i> ], actual [ <i>x</i> ] <sup>a</sup>	Whether keys are defined as modifiable or nonmodifiable does not match.
No access keys defined for structure <i>x</i> <sup>a</sup>	The structure definition has no access key defined. xfODBC optimization is impossible!
Number of access keys defined as [ <i>x</i> ], actual [ <i>x</i> ]	Fewer access keys are defined in the repository than on the physical file. (Foreign keys are not included in the count.)

a. With *repository\_options* only.

## Examples

The following example compares the database file **cust** (**-f** option) to its repository definition (**-r** option), including verification of data (**-dv** option). Errors and warnings (**-v** option) are written to a log file called **compare.log** (**-l** option).

```
fcompare -r RPSDAT:rpsmain RPSDAT:rpstext -f cust -dv -l compare.log -v
```

Using the database defined by the connect file **sodbc\_sa** (**-g** option), the example below compares the database file containing the table **customers** (**-t** option) to its system catalog definition, including verification of the data (**-dv** option). The errors, warnings, and informational messages (**-i** option) are written to the log file **compare.log** (**-l** option) in the location defined by RPSDAT.

```
fcompare -g sodbc_sa -t customers -dv -l RPSDAT:compare.log -i
```

Using the database defined by the connect file **sodbc\_sa** (**-g** option), the example below compares all database files to the system catalog definitions. Errors and warnings (**-v** option) are output to the screen.

```
fcompare -g sodbc_sa -v
```

## fconvert – Convert database files to other file types



```
fconvert [-switches] infile_spec [...] outfile_spec
```

### Arguments

*switches*

(optional) An option string that determines general processing for **fconvert**. You can either prefix the whole string with a minus sign (for example, **-xsv exceptfile**) or prefix each option with a minus sign (for example, **-x -s**). The switches are as follows:

<b>x</b> [ <i>exceptfile</i> ]	Create an exception file (for failed writes).
<b>s</b>	Display a processing summary on completion of <b>fconvert</b> .
<b>t</b> <i>temp_directory</i>	Create all temporary files in the specified directory.
<b>v</b> [ <i>count</i> ]	Display in-progress count of records processed where <i>count</i> is the record display multiple. If <i>count</i> is not specified, the display counter is throttled by a factor of 10 up to 100,000 records. If fast load optimization is occurring, an “Optimizing...” message is displayed.
<b>%</b>	Display the completion status as a percentage for all file conversions.
<b>h or ?</b>	Display the online help.

*infile\_spec*

A specification for the files to be converted, transferred, or modified. See the [Discussion](#) for syntax. You can specify more than one type of input file by specifying multiple *infile\_spec* specifications.

*outfile\_spec*

A specification for the output file. See the [Discussion](#) for syntax.

### Discussion

**Fconvert** converts, transfers, and modifies Synergy database files. In a client/server configuration, **fconvert** transfers and converts files directly to a remote host. If a remote host employs a different file structure from the client, **fconvert** automatically converts files to the host file structure. Network transfers are cached automatically. **Fconvert** can also modify the parameters of existing ISAM files.

**Fconvert** reads and writes data records to and from any of these file types (remote or local):

- ▶ ISAM
- ▶ Relative

- ▶ Counted
- ▶ Text (stream LF and stream CR/LF)

An *infile\_spec* has the following syntax:

*-infile\_type [-infile\_locking] [-infile\_options] infile\_s*

where

*infile\_type*

Determines the file type of the input file or files that follow it.

<b>ii</b>	ISAM file
<b>ir</b>	Relative file
<b>ic</b>	Counted file
<b>it</b>	Text (native stream)
<b>i1</b>	Text (stream LF)
<b>i2</b>	Text (stream CR/LF)

*infile\_locking*

(optional) An option that determines how the input files will be opened in **fconvert**. The locking options are as follows:

<b>l</b>	Open the file exclusively using Q_EXCL_RO. (default)
<b>n</b>	Open the file with NO_LOCK set, so input-file locking does not occur.

*infile\_options*

(optional) Options that determine how **fconvert** handles records in a specified input file. You can either prefix the whole string with a minus sign (for example, **-rt 50** for an input record size of 50) or prefix each option with a minus sign (for example, **-r 50 -t**). The input record file switches are as follows:

<b>r</b> <i>resize</i>	Specify input record size.
<b>t</b>	Trim blanks from end of records until record size equals the output record size.
<b>8</b>	Suppress “Binary data” warning on 8-bit characters for sequential input files ( <i>infile_type</i> of <b>-it</b> , <b>-i1</b> , or <b>-i2</b> ). (Decimal values less than a space [32], except CR, LF, HT, and FF, will still cause a “Binary data” warning on an <i>infile_type</i> of <b>-it</b> .)
<b>k</b> <i>krf</i>	Specify key of reference for ISAM (default 0).



*Infile\_type* must precede *infile\_locking* and *infile\_options*, and all three must precede the names of the files they define.

### *infile*

The name(s) of the file(s) of the same type to be converted, transferred, or modified by **fconvert**. Separate multiple filenames with blanks.

An *outfile\_spec* has the following syntax:

*-outfile\_type* [*-outfile\_locking*] [*-outfile\_options*] *outfile*

where

### *outfile\_type*

Determines the type for the file created by **fconvert**. If **o** is the first character of *outfile\_type*, **fconvert** creates a new file. If the file already exists, it will be replaced if you specify the **-f** *outfile\_options* flag. If **a** is the first character of *outfile\_type*, **fconvert** appends output to an existing file. The output file types are as follows:

<b>oi</b> or <b>ai</b>	ISAM file
<b>or</b> or <b>ar</b>	Relative file
<b>oc</b> or <b>ac</b>	Counted file
<b>ot</b> or <b>at</b>	Text (native stream)
<b>o1</b> or <b>a1</b>	Text (stream LF)
<b>o2</b> or <b>a2</b>	Text (stream CR/LF)

### *outfile\_locking*

(optional) An option that determines whether the output file is opened with or without locking. The locking options are as follows:

<b>l</b>	Open the file exclusively using Q_EXCL_RO. (default)
<b>n</b>	Open the file with NO_LOCK set, so output-file locking does not occur. Fast load optimization is turned off.

### *outfile\_options*

(optional) Options that define the way **fconvert** handles the output file. These options do not necessarily need to directly precede *outfile*.

<b>ak</b>	Generate new autokey values.
<b>f</b>	Force an existing output file to be overwritten.

<b>r</b> <i>resize</i>	Specify the output record size.
<b>d</b> <i>descfile</i>	Specify a description file. To create an ISAM file, you must specify a description file, which must either be a “parfile” (the output of the ISAM utility <b>ipar</b> ) or an XDL file.

#### *outfile*

The output filename.

**Fconvert** can fast-load all of these input file types into an empty ISAM file (or an ISAM file that is being created). The fast-load operation is highly optimized for speed. If the output file is not empty, normal processing occurs. The fast-load operation requires that several temporary work files be created. These temporary files are created in the current directory by default. Use **-t temp\_directory** to alter the location where temporary files are created.



Writing temporary files to a secondary disk may improve overall performance of the fast-load operation.

---

Make sure sufficient free disk space is available wherever temporary files are to be created.

If change tracking is enabled for the input file, change tracking will be enabled for the output file.

### ISAM files

To create an ISAM file from any file type other than ISAM, you must supply a description file. When creating an ISAM file from another ISAM file (a remote file, for example), **fconvert** uses the first input ISAM file structure by default. To override this default, use a description file; description files can be used to change the parameters of ISAM files (number of keys, record size, key characteristics, and so forth).

### Relative files

To convert a variable-length text file or variable-length-record ISAM file to a relative file, use **-r resize** to specify the record size of the relative file (see [outfile\\_options on page 3-64](#)).

### Counted files

The counted file is a derived file type; there is no equivalent Synergy DBL OPEN submode for counted files. You can, however, read or create a counted file using **isload**.



Files created with record sizes greater than 64K cannot be counted files.

---

### Text files (stream LF and stream CR/LF)

When specifying a text file, use the **-it**, **-at**, or **-ot** option for native stream files. To convert a Windows text file to a UNIX text file from a UNIX system, specify **-it** for the Windows file and **-ol** for the UNIX output file.



We don't recommend using text files that contain binary data (such as records containing integer fields or alpha fields that contain RFAs.) For example, when reading a record with an integer field that contains the value 2600 (0x0A28 in hex), **fconvert** will terminate the record when it encounters the 0x0A (LF) and start a new record with the next character. Therefore, as a precaution, a "Binary data (*n*) detected in *file* at *nn*" warning is displayed if binary data is encountered (excluding the CR, LF, HT, and FF line terminators).

---

### Exception files

If the **-x** option is specified, an exception file is created if any records fail to convert due to an "Illegal record size" or "No duplicates allowed" error. The exception file is a counted file. Use **fconvert** to convert the records to another file format. The default name for the exception file is *infile.exc*. To designate a different exception filename, specify the optional *exceptfile* argument to the **-x** option. You cannot specify a name for an exception file if you are converting more than one file. **Fconvert** will not run with an existing exception file.



The **fconvert** utility operates on Revision 4 or higher files. To convert files to a lower or higher revision, set the ISAMC\_REV environment variable before running **fconvert**. (See [ISAMC\\_REV](#) in the "Environment Variables" chapter of *Environment Variables & System Options* for more information.)

---

### 8-bit characters

When processing sequential input files, **fconvert** will detect and warn of possible binary data. The presence of binary data could cause premature record termination when the file is handled as a sequential file. (Note: You may be able treat the file as a relative file if the record size is a fixed length.) If you use 8-bit characters in your sequential files, you will want to specify the **-8** option, which suppresses the warning message on ASCII characters above 127. This option must be specified for each input file for which message suppression is desired.

## Examples

The example below converts the ISAM file **file1** to a relative file named **file2**:

```
fconvert -ii file1 -or file2
```

The following is an example of an **fconvert** command to transfer the ISAM file **file1** and the relative file **file2** to the ISAM file **file3**. If **file3** does not exist, **fconvert** creates a file named **file3** described by the parameter file **file3.par**. If **file3** exists, **fconvert** overwrites it with a new **file3**. **Fconvert** displays a counter of processed records and a summary on completion.

```
fconvert -sv -ii file1 -ir file2 -oif file3 -d file3.par
```

The example below transfers **file1** to remote host **server1**. If the file exists, it is overwritten with the new file. When the transfer is completed, a summary is displayed with statistics.

```
fconvert -s -ii file1 -oif file1@server1
```

In the following example, three similar sequential files named **file1**, **file2**, and **file3** are loaded into the new ISAM file **file4**. The temporary file created during fast-load processing is written on a secondary disk E: in the work subdirectory. A completion percentage is displayed as the files are processed, and a summary is displayed at the end.

```
fconvert -s%t e:\work -it file1 file2 file3 -oif file4 -d file4.par
```

The example below loads the file **isamfile.ism** with records from **file1.ddf** and recognizes that there may be valid 8-bit ASCII characters in the file.

```
fconvert -it8 file1 -ai isamfile
```

## ipar – Generate parameter file descriptions

WT	WN	U	
----	----	---	--

```
ipar [-option] filename[, ...]
```

### Arguments

*option*

(optional) One or more of the following options:

- b**        Display brief output by suppressing comments. This makes it possible to compare the output from multiple issuances on the same or a similar file.
- g**        Generate a parameter file (or an XDL/FDL file if the **-x** option is also specified) that has the same name as each ISAM file for which you want a description but with the extension **.par** (or **.idl**). If both **-x** and **-g** are specified, the filename extension is **.xdl**.
- x**        Generate an XDL or FDL file to the terminal.

*filename*

The name of the ISAM file(s) for which you want to generate parameter file descriptions. The default extension is **.ism**.

### Discussion

The **ipar** utility generates parameter file descriptions of existing ISAM files, which you can then use as input to **bldism** to rebuild the same files. These descriptions contain current content information about the specified ISAM file in their comment lines.

If you don't specify any option, **ipar** sends parameter information to the terminal by default.

If **ipar** detects an **.is2** file, it generates a warning message but generates the file description anyway.

### Examples

The following example generates a parameter file named **file1.par** for **file1.ism** and a file named **file2.par** for **file2.ism**:

```
ipar -g file1 file2
```

The next example generates a parameter file with the extension **.par** for each of the ISAM files in the current directory:

```
ipar -g *.ism
```

The following example generates parameter information for our ISAM file, **cusmas.ism**:

```
ipar cusmas
```



This example sends the following information to the terminal:

```

;          Synergy ISAM PAR File created Fri Feb 12 17:36:23 1999
cusmas.ism, variable, compress
2000      ;Record size
5         ;Number of keys
          ;5ca5 magic, Revision 4, 14 byte record overhead
          ;Shared index cache allowed
          ;Creation version 7.1
          ;File created on Mon Feb 01 12:29:04 1999
          ;43 byte longest key
          ;0 free index blocks, 0x0 free list head
          ;100 records, 0 free

name/segment
          ;Primary alpha key
          ;30 key size
2         ;Number of segments
          ;Segment #1
15        ; Segment length
16        ; Start position
          ;Segment #2
15        ; Segment length
1         ; Start position
N         ;Duplicates allowed
D         ;Ascending/descending
          ;Root 0x800, index depth 1
          ;Minimum keys per block 14

company/density
          ;Alternate alpha key #1
30        ;Key size
31        ;Start position
Y         ;Duplicates allowed
Y         ;Insert at front
A         ;Ascending/descending
75        ;Index density percentage
          ;Root 0xc00, index depth 1
          ;Qualifier offset 5
          ;Minimum keys per block 13

address/modify/segment/density
          ;Alternate alpha key #2
          ;40 key size
3         ;Number of segments
          ;Segment #1
20        ; Segment length
61        ; Start position
          ;Segment #2
10        ; Segment length
51        ; Start position
          ;Segment #3

```

```

10          ; Segment length
91          ; Start position
Y           ; Duplicates allowed
N           ; Insert at front
A           ; Ascending/descending
70          ; Index density percentage
           ; Root 0x1000, index depth 1
           ; Qualifier offset 8
           ; Minimum keys per block 10
act_code/null
           ; Alternate alpha key #3
R           ; Replicating null key
32          ; Null value ' '
5           ; Key size
101         ; Start position
Y           ; Duplicates allowed
Y           ; Insert at front
A           ; Ascending/descending
           ; Root 0x1400, index depth 1
           ; Qualifier offset 11
           ; Minimum keys per block 36
cust_number/density/type
           ; Alternate key #4
D           ; Decimal type
10          ; Key size
120         ; Start position
N           ; Duplicates allowed
A           ; Ascending/descending
90          ; Index density percentage
           ; Root 0x1800, index depth 1
           ; Minimum keys per block 32

```

## irecovr – Recover Revision 4 or higher ISAM files

WT	WN	U	
----	----	---	--



The **irecovr** utility is now just a front-end for **isutl**, and we recommend that you use **isutl** instead. See [isutl on page 3-77](#).

```
irecovr [-options] filename[, ...]
```

### Arguments

*options*

(optional) One or more of the following options:

**a**            Apply suggested database file corrections. (Equivalent: **isutl -ra**)



Improper use of the **-a** option can result in loss of data. See [isutl on page 3-77](#) for more information.

**c**            Compress data records. (Equivalent: **isutl -lc**)

**f** *seqfile*    Fast load a sequential file into an empty ISAM file. (Equivalent: **isutl -f**)

**k**            Load counted database files. (Equivalent: **isutl -fk**)

**q**            Give no status messages (quiet mode). (Equivalent: **isutl -m0**)

**s**            Use static RFAs. (Equivalent: **isutl -rs**)

**v**            Display a more detailed account of processing (verbose mode). (Equivalent: **isutl -m2**)

**%**            Display the percentage of the entire file that has been recovered. (Equivalent: **isutl -%**)

*filename*

The name of the ISAM file(s) that you want to convert or recover. The default extension is **.ism**.

### Discussion

An error occurs if you use pre-Revision-4 files with **irecovr**. You can use **isutl -p** to patch such files to the latest revision.

## isload – Load, unload, or clear an ISAM file



The **isload** utility enables you to load or unload ISAM or relative files or to clear ISAM files from outside your programs. It is primarily used to load ISAM files from text to sequential files.



Whenever possible, isload opens the ISAM file exclusively. Because this support may not be available on all operating systems, please use **isload** with caution.

To run **isload**,

On	Enter this at the command line
Windows and UNIX	dbr DBLDIR:isload
OpenVMS	run DBLDIR:isload

To find out what the valid input is at any prompt, enter a question mark character (?). To terminate **isload** at any time, type the end-of-file character for your operating system.



On Windows and UNIX we suggest using the **fconvert** utility (see [fconvert on page 3-62](#)) for much faster operation of loading and unloading files.

On OpenVMS we suggest using the system command CONVERT/FDL for much faster operation of loading and optimizing files.

### Sample isload

Option: unload

Enter name of ISAM file to be UNLOADed: `cusmas,key=1`

Record length: 2000

Number of keys 4

Enter name of sequential file into which to UNLOAD: `secust`

What progress reporting interval? 25

--> Begin unloading ISAM to sequential (at 14:14:43)

at 14:14:43 25 in 25 out 0 errors

at 14:14:43 50 in 50 out 0 errors

at 14:14:43 75 in 75 out 0 errors

at 14:14:43 100 in 100 out 0 errors

... End of input file test

--> Finish unloading ISAM to sequential (at 14:14:43, took 0 seconds)

Records input: 100 Records output: 100 Errors detected: 0

... normal termination of ISLOAD

## Running the isload utility

To illustrate how you use **isload** to unload a file, let's assume we want to unload our ISAM file, **cusmas.ism**. The **isload** utility prompts us as follows for the information needed to unload our ISAM file. (The example to which we refer throughout this section is found in ["Sample isload" on page 3-72.](#))

- ▶ **Option:** Enter one of the following options:

<b>UNLOAD</b>	Unloads the specified file to a sequential file without modifying the file.
<b>LOAD</b>	Loads the specified file by adding new records to the existing records.
<b>CLEAR</b>	Clears the specified ISAM file and restores it to its original empty state.
<b>STOP</b>	Terminates the <b>isload</b> utility.



The file you want to unload, load, or clear must already have been created using either **bldism** or the ISAMC subroutine. You can abbreviate any of the above options; for example, U for UNLOAD or L for LOAD.

---

In our example, we entered the UNLOAD option to unload our ISAM file, **cusmas.ism**.



Use CLEAR with extreme caution; clearing an ISAM file deletes *all* existing records.

---

- ▶ **Enter name of ISAM file to be UNLOADed:**
- ▶ **Enter name of ISAM file to be LOADed:**
- ▶ **Enter name of ISAM file to be CLEARed:**

Enter the specification for the ISAM file you want to unload, load, or clear as follows:

*file\_spec*[, RELATIVE][, KEY=keynum]

*file\_spec*

The specification for the file you want to load, unload, or clear. The default extension is **.ism**. *File\_spec* can be up to 100 characters long.

## RELATIVE

(optional) Indicates that the file to load or unload is a relative file. You can abbreviate this option to any number of characters (for example, R or REL).



The RELATIVE specification is valid for LOAD or UNLOAD operations only.

---

*keynum*

(optional) The number of the alternate key by which you want to unload the specified ISAM file. By default, the file is unloaded in primary key order. Specifying the KEY=*keynum* option enables you to unload a file by an alternate key.



The KEY=*keynum* option is valid for UNLOAD operations only.

---

In our example, we entered the filename **cusmas,key=1** to unload our ISAM file by the first alternate key. The **isload** utility then displayed a record length of **2000** and **4** keys for our ISAM file, **cusmas.ism**.

After you enter a filename and press ENTER, the **isload** utility displays the maximum length of records and the number of keys for the specified file. If you selected the clear option, the **isload** utility now clears the specified ISAM file and terminates. If you selected the unload or load option, **isload** displays the next prompt.

- **Enter name of sequential file from which to LOAD:**
- **Enter name of sequential file into which to UNLOAD:**

Enter the name of the sequential file into which you want to write records during the unload operation or from which you want to read records during the load operation. The file specification and corresponding qualifier list can be up to 100 characters long, as follows:

*file\_spec*[, FIXED|COUNTED][, NOLOCK]

*file\_spec*

The specification for the sequential file into which you want to unload records or from which you want to load records. The default extension is **.ddf**. (TT: is recognized as the terminal.) *File\_spec* can be up to 100 characters long, including any of the optional qualifiers listed below.

**FIXED**

(optional) Specifies that the sequential file will contain fixed-length records of the maximum length specified at creation.

**COUNTED**

(optional) Specifies that the sequential file will contain variable-length records. A counted file is used and recognized only by **isload** and **fconvert** for the purpose of loading and unloading ISAM files. (COUNTED is not available for files created with record sizes greater than 64K.)

## NOLOCK

(optional) Specifies that record locking will not occur. NOLOCK can only be used when a file is being loaded; it is not valid with the UNLOAD option.



By default, the file is loaded from or unloaded to a sequential file. Because ISAM files allow binary data, however, record terminators and end-of-file characters can be embedded in records. Therefore, for ISAM files containing records that have binary data or integer fields, use the optional FIXED or COUNTED options in the sequential filename specification. (When **isload** unloads to a counted sequential file, it stores the count in portable integer form. These counted sequential files are portable between big-endian and little-endian machines.)

During unload operations, if the specified output file fills up, **isload** displays an “Output file is full” error (\$ERR\_FILFUL) and prompts you for the name of the continuation file into which you want to write further output. If you type the end-of-file character at this prompt, **isload** terminates without unloading all of the records from the specified ISAM file.

During load operations, if you encounter the end of the specified sequential file, **isload** displays the message

### End of input file *filename*

and prompts you for the name of the continuation file from which to continue reading records. If you type the end-of-file character at this prompt, **isload** closes the specified ISAM file and terminates.

*In our example, we entered the filename **secust** to unload our **cusmas.ism** file to the sequential file **secust.ddf**.*

- **What progress reporting interval?** Enter a number representing the frequency (in records) with which you want **isload** to display the progress of the unload or load operation. Press ENTER if you don’t want **isload** to display the progress.

If you enter a number of records at this prompt, the progress notification line displays the time **isload** began loading or unloading, the number of records that **isload** has processed, and the number of errors that occurred during that portion of the unload or load operation.

*In our example, we entered the number **25** at this prompt and **isload** displayed a progress notification line every 25 records.*

During both load and unload operations, I/O errors might occur. Whenever an error is encountered, **isload** displays an error message that identifies the file involved, the record number, and the error detected. You can then either skip the record that triggered the error or abort processing. If you abort, **isload** closes both the ISAM file and the sequential file, retaining whatever it unloaded or loaded prior to the abort.

*In our example, **isload** didn’t encounter any errors and terminated normally.*

## ismvfy – Verify structure of a Revision 4 or higher ISAM file

WT	WN	U	
----	----	---	--



The **ismvfy** utility is now just a front-end for **isutl**, and we recommend that you use **isutl** instead. See [isutl on page 3-77](#).

```
ismvfy [-options] filename[, ...]
```

### Arguments

*options*

One or more of the following options:

- b**        Display bucket usage statistics. (Equivalent: **isutl -vb**)
- c**        Give approximate compression saving for noncompressed ISAM files. (Equivalent: **isutl -c**)
- l**        Don't require exclusive access during verification. (Equivalent: **isutl -vl**)
- n**        Verify the index only. (Equivalent: **isutl -vn**)
- v**        Print key values (verbose mode). (Equivalent: **isutl -m3**)
- %**        Display the percentage that has been verified for each key. (Equivalent: **isutl -%**)

*filename*

The name of the ISAM file(s) whose structure you want to verify. The default extension is **.ism**.

### Discussion

An error occurs if you use pre-Revision-4 files with **ismvfy**. You can use **isutl -p** to patch such files to the latest revision.



## isutl – Verify, recover, and optimize Revision 4 and higher ISAM files

WT	WN	U	
----	----	---	--

```
isutl -r [-reload_options] [-other_options] filename[ ...]
```

or

```
isutl -v [-verify_options] [-other_options] filename[ ...]
```

or

```
isutl -f[ak] sequential_filename [-fastload_options] [-other_options] filename[ ...]
```

or

```
isutl -p [rev] [-patch_options] filename[ ...]
```

or

```
isutl -i filename
```

### Arguments

**-r** Reload the index for the specified ISAM file.

*reload\_options*

(optional) One or more of the following options for rebuilding the index:

**a** Apply suggested database file corrections.



Improper use of the **-a** option can result in loss of data. Read the [Discussion](#) before using this option.

**o** *key#* Order database file by the specified key during the reload operation.

**p** *density* Pack index blocks to the specified percentage for each defined key during the reload operation. Note that **-p** does not change the file density setting.

**qfile=opt[,opt,...]** Convert file using the specified options. Valid *opt* values are **compress**, **tbyte**, **static\_rfa**, **page=page\_size**, and **density=density**. (See the [Discussion](#).)

**s** Convert file to use static RFAs. (This option is ignored on Revision 6 ISAM files.)

- v Verify the specified ISAM file.

#### *verify\_options*

(optional) One or more of the following options for verifying the ISAM file(s):

- b** Report bucket usage statistics and freelist usage.
  - i** Launch the information advisor to provide a full analysis of file organization and content. (See the [Discussion](#).)
  - l** Verify using cooperative locking without requiring exclusive access. If a locked record is encountered, **isutl** waits for the record to become available. (default)
  - le** Verify with exclusive access locking.
  - ln** Verify files that don't allow locking (e.g., read-only files).
  - n** Bypass use of the data file and verify only the index during the verify operation. Do not use this option when file integrity is in question.
  - z** Scan for *all* problems, rather than stopping at the first detected problem.
- f Fast-load a sequential file into an empty ISAM file. This operation is highly optimized for speed.
- ak** (optional) Generate new autokey values.

#### *sequential\_filename*

The name of the sequential file to load into the specified empty ISAM file.

#### *fastload\_options*

(optional) One or more of the following options for fast-loading a sequential file:

- k** Identify the sequential file as a Synergy counted file.
  - p density** Pack the index blocks to the specified percentage for each defined key during the load operation.
- p Patch an existing ISAM file to a higher or lower revision.

#### *rev*

(optional) Revision to patch to. The default is 6 or the value of ISAMC\_REV, if set. (See [“Patching an ISAM file to another version”](#) on page 3-82 for more information.)

#### *patch\_options*

(optional) One of the following options:

- qfile=/no/convert** Force large files to be converted or small files to be patched. (By default, small files will be converted and large files will be patched. A patched file can later be patched back to the original revision, whereas a file that has been converted cannot be patched back.)

**qfile=[no/network\_encrypt** Set or unset the network encryption flag on the specified file.

**-i** Launch the information advisor.

#### *other\_options*

(optional) One or more of the following options:

**c** Convert file to compressed data (when used with **-r**) or report expected compression savings (when used with **-v**).

**h** or **?** Display help screen.

**mlevel#** Specify a message level that defines the amount of information displayed during an operation, where *level* is a value from 0 to 3. (See the [Discussion](#).)

**t directory** Specify a temporary file directory.

**%** Display a running status (0 to 100) to indicate the percentage completed by the operation.

#### *filename*

The name of the ISAM file(s) whose index structure you want to reload or verify, or into which you want to load a sequential file. The default extension is **.ism**.

## Discussion

The ISAM File Maintenance Utility (**isutl**) can perform one of five functions:

- ▶ Rebuild an ISAM file
- ▶ Verify the integrity of an ISAM file
- ▶ Quickly load (fast-load) a sequential file into an existing, empty ISAM file
- ▶ Patch an existing ISAM file to another version
- ▶ Launch the information advisor

### **Rebuilding an ISAM file**

Rebuilding an ISAM file (**-r**) can take several forms: re-indexing only, ordering data with re-index, converting data with re-index, and recovering data with re-index.

Re-indexing causes index blocks to be packed and arranged adjacent to related index blocks to enhance ISAM lookup performance. If index density is not defined, the following default packing percentages are used:

Page size	Packing percentage
512	80%
1024	80%
2048	90%
4096	95%
8192	97%
16384	97%
32768	97%

If at least three key entries cannot fit into the space that's left, the default percentage is reduced to 80% for all page sizes. If more (or less) empty index space is desired, specify the packing density explicitly with **-p** or change the density setting for the file with **-qfile=density=density**. See [“ISAM index density” on page 3-8](#) for more information about density.

In addition, the data file can be ordered by a preferred key to maximize sequential read performance. As a file grows, the speed of sequential access to a primary key is greatly reduced due to large file disk seeks. The **-o** option orders the data in *key#* order for high-speed sequential file access of the key specified, making it significantly more efficient. *Key#* must be a valid key number defined by the file.



When ordering data (**isutl -ro filename**), **isutl** generates a sort temporary file called *filename\_is1.257*. (For example, if your **.is1** file is named **armast.ms1**, the temporary file is **armast\_ms1.257**.) If this file is left on your system due to abnormal termination of **isutl**, *do not remove it*. If **isutl** was in the process of writing data records to the **.is1** file when termination occurred, the **.257** file is required to completely restore your data. All other temporary files created by **isutl** (having the same filename with the extensions **.000 – .256** or **.258**) are short-lived. Assuming these files are not in use, they can be removed without consequences. To resume the sort, run **isutl -r** again.

---



xfODBC always assumes the primary key is the most optimal key by which to read the data. Therefore, xfODBC performance is affected if you reorder your data on an alternate key.

---

The **-c** and **-s** options will convert ISAM data to compressed and/or static RFA respectively. Once converted, these file attributes cannot be reversed using **isutl**. (To reverse the attributes, see [fconvert on page 3-62](#).) Subsequent use of these options on the same file is ignored. If the compressed data option (**-c**), data reorganization option (**-o key#**), or data correction option (**-a**) is not specified, a new index file is created without altering the existing data file.



Static RFAs for a file may no longer be valid after altering the ISAM data file. The following reload options alter the data file: **-c**, **-o**, or **-s**.

---

When converting (**-c** or **-s**) or ordering (**-o**) data, all unused RFA space and free space due to record deletions is reclaimed. To explicitly reclaim this space and reduce data file size, we suggest ordering the data periodically.

If the index packing density option (**-p density**) is not specified, the density defined for that key (or the default density if none is defined) is used.

The **-qfile** option enables you to set a string of file options in the format **-qfile=opt[,opt,...]**, where each *opt* is one of the following:

<b>compress</b>	Convert file to compressed data. (This is identical to the <b>-c</b> option.)
<b>tbyte</b>	Convert to a terabyte file.
<b>static_rfa</b>	Convert file to use static RFAs. (This is identical to the <b>-s</b> option. It is ignored on Revision 6 ISAM files.)
<b>page=page_size</b>	Change the file's index page block size. Valid <i>page_size</i> values are 512, 1024, 2048, 4096, 8192, 16384, and 32768.
<b>density=density</b>	Change the file density and pack the file, where <i>density</i> is the density percentage. (This differs from the <b>-p density</b> option in that it permanently sets the file density for all keys.)

Rebuilding an ISAM file always corrects existing index errors; however, errors in the data file may not be completely recoverable. Use of the **-a** option may be useful in these situations.



**Isutl** instructs you to use the **-a** option when necessary. This option causes **isutl** to recover as much data as possible, but to do so, it must alter the original data file. Once **-a** is performed, lost data cannot be recovered, so we highly recommend that you back up the file before using it. An alternative method of recovery is **fconvert**. Unless you are prompted to do so, we do not advise using **-a**. An exception file may be produced containing the unrecognized data removed from the file. Specifying multiple filenames is not allowed. Do *not* automate this process.

---

Use of this utility on a pre-Revision-4 file generates an error, unless the patch (**-p**) option is used to convert the file to an appropriate revision first. See [“Patching an ISAM file to another version” on page 3-82](#).

## Verifying the integrity of an ISAM file

By default, the verify command (**-v**) uses the fastest optimized methods to assure file integrity. If **isutl** detects something that would result in a Synergy data access failure, it generates an error and stops the verify operation. By using the **-z** option, a more linear scan is performed and all errors are displayed in context, which may reveal the entire severity. The **-z** option, however, can be more time consuming, especially with very large nonoptimized files. In other words, regardless of how many problems a file contains, **isutl -v** stops at the first problem detected and generates an error, while **isutl -vz** detects all the problems that it can but might be somewhat slower.



We recommend running without **-z** first, and then using **-z** only on files that show problems.

---

By default, the **-l** option is enabled, which turns on cooperative locking. Cooperative locking means other processes can have the file open, but processes attempting to read or write to the file may be suspended until the verify is complete. The **-le** option turns on exclusive access locking, which requires that the file not be open by any other process, and it cannot be opened by any other process while the verify is running. The **-ln** option allows verification of files that don't support locking, such as read-only files or files on a CD-ROM.

## Fast-loading a sequential file

The fast-load command (**-f**) quickly loads a sequential file or counted file into an empty ISAM file. When loading an empty ISAM file with **isutl -f**, benchmark tests indicate performance is three to five times faster than the **isload** utility.



See the **fconvert** utility for loading ISAM files. You can expect the same performance with fewer file restrictions.

---

## Patching an ISAM file to another version

The patch command (**-p**) patches one revision to another.

By default (if *rev* is not specified), **isutl -p** patches Revision 2, 3, 4, or 5 files to Revision 6. For example, the following automatically patches all ISAM files with a **.ism** extension in the current directory to Revision 6:

```
isutl -p *.ism
```

If you need to support Synergy versions prior to 10, set **ISAMC\_REV=4**. This will ensure that your ISAM files remain compatible with your pre-v10 Synergy. (When **ISAMC\_REV=4**, **isutl -p** patches Revision 2 or 3 files to Revision 5, which is a Revision 2 or 3 file structure in a Revision 4 format. Then the file can be used with **isutl**. Note that a Revision 5 file can only be accessed by Synergy version 7.5 or higher.)

If *rev* is specified, **isutl** will attempt to patch the file to that revision.

If **-p** is specified without the **-qfile=convert** option, a revision-only patch (to *rev* or the default revision if *rev* is not specified) occurs. The following files are implicitly converted to the new revision:

- ▶ Files that contain 10,000 or fewer records whose record size is 4K or less.
- ▶ Files that contain 1,000 or fewer records whose record size is greater than 4K.

Any files larger than these are patched to an intermediate revision (5r6 for Revision 2, 3, or 5 files and 4r6 for Revision 4 files). An intermediate revision file occupies the same footprint as the original file, but it's a newer representation whose header contains structural differences for Revision 6. Over time, intermediate revision files are automatically converted to Revision 6 either after being cleared or after being rebuilt using **isutl -ro**. Alternatively, at a time when it's convenient, you can issue the **isutl -p -qfile=convert** command to convert all of the large files to Revision 6.

When **-p** is specified with the **-qfile=network\_encrypt** option, a revision patch (if *rev* was specified) occurs first and then the **network\_encrypt** option is applied. If *rev* is not specified, the file revision is not changed; only the **network\_encrypt** option is applied.

### Launching the information advisor

The information advisor command (**-i**) displays helpful advice based on file organization and content. The identified file conditions can range from high-risk issues that may result in file failure to low-risk, performance-related issues. If the file condition is correctable, the information advisor will suggest corrective actions and/or ways to enhance performance.



The conditions reported are *not* errors. They simply provide helpful information to point out things that may or may not be otherwise detectable and suggestions that you can choose to ignore or act upon. Having one or more conditions be displayed for a file does *not* mean the file is corrupted.

---

Use **isutl -i filename** to quickly check static configuration information, or **isutl -vi filename** to generate a full analysis based on content.

**Isutl -i** (or **-vi**) reports the following static conditions:

- ▶ **Duplicates exceed 80% full on one or more keys.** Duplicates that exceed 100% will be denied with an \$ERR\_FILFUL error.
- ▶ **Index freelist overflow.** Performance during STORE operations may be suffering.
- ▶ **Invalid file organization for pre-7.3.1a versions.** The potential for file corruption exists if you're using an old Synergy version.
- ▶ **Static RFA: High data segment re-use.** This file may benefit from reorganization.
- ▶ **Static RFA: Large number of vectored segments.** This file may benefit from reorganization.
- ▶ **Duplicates ordered at the beginning on one or more keys.** **Isutl** performance may suffer. Consider changing to ATEND unless absolutely necessary.

- ▶ **Index depth exceeds 3 on one or more keys.** Increasing PAGE size may improve overall performance.
- ▶ **Free space exceeds half.** The percentage of free space is specified. Reducing file size may improve performance.

The following conditions are only available with **isutl -vi**:

- ▶ **One or more keys exhibit excessive blank duplicates.** Change to a replicating null key to improve performance.
- ▶ **Index exhibits a low optimization level.** Keyed and sequential file access may not be optimal.
- ▶ **Data exhibits a low optimization level.** Sequential file access may not be optimal.
- ▶ **Existing data exhibits some compression benefit.** Compressing data will reduce file size and improve performance.
- ▶ **Static RFA: Vectored records exceed 1%.** RFA record access performance may suffer.
- ▶ **Static RFA: Unusable data exceeds *percentage*.** Specifies the percentage and number of data bytes that are not usable.
- ▶ **Variable segment reduction waste: (n).** When compressed records are rewritten and are smaller, they reuse the same segment but generate more waste. This information represents the savings that would occur if these smaller records were moved to a smaller segment. Use **isutl -ro** to reclaim this space.

## Other options

The *level#* argument to the **-m** option can have one of the following values:

- 0** No output is generated. All errors generated are returned in the form of an exit value.
- 1** Only errors and necessary output is displayed. (default)
- 2** Process information is displayed, in addition to errors and necessary output.
- 3** Verbose key information is displayed if this message level is specified with the **-v** option. (If **-m3** is specified with **-r** or **-f**, the message level defaults to **-m2**.)

To measure the degree of a file's optimization, specify the **-m2** flag on the verify (**-v**) command. For each key, a line will be displayed that indicates, as a percentage, the optimization level as well as the sequential order of the data file. For example:

```
Primary key, 751406 blocks (728570 leaf), 19416428 records
Index density 50%, leaf 50%, separator 50%
Optimization: index 44%, data 95%
```

The index percentage indicates the percentage of on-disk index blocks for the target key that can be accessed quickly from a previous index block with the least amount of disk overhead. The data percentage indicates the percentage of data records in a sorted order giving the least amount of disk overhead when reading sequentially by that key. The effectiveness of these percentages vary



depending on file size and hardware configuration. They tend to become more significant as the file size exceeds the available file cache memory on a system. No additional overhead is consumed as a result of getting this information.



To get this optimization information quickly and accurately on a large file, you can also specify the **-n** verify flag. However, you shouldn't rely on the verification results, because only the index is verified.

The **-t** option specifies a directory for all temporary work files and can be specified with either the **-r** or **-v** command. The directory specification must be a valid path specification or logical that references a local or network drive. An xfServer remote file specification is not supported. The default location for temporary files is the current directory.



Writing temporary files to a secondary disk may improve overall performance.



When processing large ISAM files, make sure sufficient disk space is available for the temporary work files.

The amount of disk space required for temporary files varies with the operation. In general, you can assume the following:

Operation	Maximum temporary file size (approximate)
Re-index only ( <b>-r</b> )	2 * (size of largest key * #records)
Order data ( <b>-ro</b> )	1.2 * size of in-use data
Convert data ( <b>-rc</b> or <b>-rs</b> )	size of in-use data
Verify ( <b>-v</b> )	(overall index density * size of index file) + (size of largest key * #records) or ~ 80% size of index file
Verify linear ( <b>-vz</b> )	No temporary files used
Patch ( <b>-p</b> )	No temporary files used

When re-indexing only, the total disk space occupied (ISAM file plus temporary files) will not exceed the original size of the ISAM file (unless the packing density is reduced).

The **-%** option can be specified with either the **-r** or **-v** command. When specified with **-r**, the numbers displayed indicate the percentage of the overall reload operation completed for each file. When specified with **-v**, the numbers displayed indicate the percentage of the overall verify operation completed for each file. When message level 2 (**-m2**) is also specified, an individual process percentage as well as a total overall percentage is displayed for each file.

**Isutl** generates an exit status, which can be especially useful if you've used the **-m0** option. Possible exit statuses are as follows:

This status	Indicates
0	<b>Isutl</b> was successful.
A Synergy DBMS error number	<b>Isutl</b> failed as a result of the specified error. (See <a href="#">“Synergy DBMS Errors” on page 5-128</a> for the message text that maps to each error number.)
-1	More than one file was specified on the command line, and at least one file failure occurred.

If you use the **-z** option on a corrupted file, the exit status reflects the first error only.

When the file is successfully processed, the current date is written to the index control record to indicate the last recover or verify. This information can be accessed using the **ipar** utility.



**Isutl** does not support loading records with binary data from sequential files (excluding counted files). Attempting to do so can cause some records to split into two records in the ISAM file. To load a relative file that contains binary data into an ISAM file, use the **fconvert** utility.

---

**Isutl** generates a log file named **isutl.log** that records its operations and results. Each log file entry specifies the ISAM filename, the operation performed, the date and time the operation was performed, the command line options supplied to **isutl**, the exit status, and the amount of time the operation took. The log file is created in the TEMP directory. (We recommend always using ISUTLLOG to specify the log file location on these systems.) The maximum size of the log file defaults to 1 megabyte, unless the ISLOGMAX environment variable defines a different maximum. To disable logging, set ISLOGMAX to 0. (See [ISUTLLOG](#) and [ISLOGMAX](#) in the “Environment Variables” chapter of *Environment Variables & System Options* for more information.)

## status – Report the status of an ISAM file



The **status** utility generates a report that describes the organizational characteristics of a specified ISAM file and indicates how many records are currently in the file.

To run **status**,

On	Enter this at the command line
Windows and UNIX	dbr DBLDIR:status
OpenVMS	run DBLDIR:status

To find out what the valid input is at any prompt, enter a question mark (?). To terminate **status** at any time, type the end-of-file character for your operating system.

### VMS

The **status** utility always returns 90,000,000 as the number of records in an RMS ISAM file, as it relies on the ISSTS subroutine. There is no way to find this information on an RMS file unless you read sequentially through the file.

## Sample status

The following example is run on UNIX.

Enter ISAM file name: cusmas

File to write status to: cusmas

Enter ISAM file name: ^D

... normal termination of STATUS

*In our example, **status** writes the following information to a sequential file named **cusmas.ddf**:*

The record length for this file is 2000 characters.

There are 5 keys.

There are currently 100 records in this ISAM file.

Primary key is name

The key is 30 characters long, segmented and is ordered in descending sequence with no duplicates allowed.

This key may not be modified by WRITE.

Segment #1 starts at 16 with length 15.

Segment #2 starts at 1 with length 15.

1st alternate is company

The key is 30 characters long, starting at position 31 within the record, and is ordered in ascending sequence with duplicates allowed.

This key may not be modified by WRITE.

2nd alternate is address

The key is 40 characters long, segmented and is ordered in ascending sequence with duplicates allowed.

This key may be modified by WRITE.

Segment #1 starts at 61 with length 20.

Segment #2 starts at 51 with length 10.

Segment #3 starts at 91 with length 10.

3rd alternate is act\_code

The key is 5 characters long, starting at position 101 within the record, and is ordered in ascending sequence with duplicates allowed.

This key may not be modified by WRITE.

4th alternate is cust\_number

The key is 10 characters long, starting at position 120 within the record, and is ordered in ascending sequence with no duplicates allowed.

This key may not be modified by WRITE.

## Running the status utility

*To illustrate how you use **status**, let's assume we want to report the status of our ISAM file, **cusmas.ism**. The **status** utility prompts us as follows for the information needed to retrieve the status of our ISAM file. (The example to which we refer throughout this section is found in [“Sample status” on page 3-87.](#))*

- ▶ **Enter ISAM file name:** Enter the name of the ISAM file for which you want to report the current status. The default extension is **.ism**.

*In our example, we entered the filename **cusmas** to report the current status of our ISAM file, **cusmas.ism**.*

- ▶ **File to write status to:** Enter the name of the output file to which you want to write the status report of the specified ISAM file. The default extension is **.ddf**. If the specified file already exists, **status** generates a “Cannot supersede existing file” error (\$ERR\_REPLAC) and prompts you for another output file.

If you press ENTER at this prompt without specifying a filename, the status report is sent to the terminal.

The **status** utility closes each output file at the conclusion of the status operation. After sending the status report to the specified output file or terminal, **status** repeats the first prompt. If you want to terminate **status** operations, type the end-of-file character.

*In our example, we entered the filename **cusmas** to send the status report to a sequential file named **cusmas.ddf**, and then typed the end-of-file character to terminate the status utility.*

# ISAM Definition Language

The **bldism** utility enables you to create an ISAM file outside your application by getting information from keyboard input, from a parameter file (output from the **ipar** utility), or from an ISAM definition language (XDL) keyword file. This section describes the ISAM definition language file and the keywords required to create one.

## Rules for XDL keyword files

The following rules apply to XDL keyword files:

- ▶ The file must contain valid XDL keywords followed by their assigned values. (See “[XDL keywords](#)” below.)
- ▶ The file may contain comments, which must begin with an exclamation point. The rest of the line following an exclamation point is ignored. Blank lines are also allowed.
- ▶ An XDL description must contain one FILE and one SIZE keyword. In addition, each key definition must contain exactly one LENGTH and one START keyword.
- ▶ The file attribute section of the definition may not contain more than one FILE, NAME, ADDRESSING, PAGE\_SIZE, KEYS, NETWORK\_ENCRYPT, SIZE\_LIMIT, RECORD\_LIMIT, ROLLBACK, STORED\_GRFA, TRACK\_CHANGES, TEXT, TEXT\_ALLOCATION, DENSITY, RECORD, SIZE, FORMAT, COMPRESS\_DATA, or STATIC\_RFA keyword.
- ▶ File keywords must follow FILE, record keywords must follow RECORD, and key keywords must follow KEY.
- ▶ A keyword and its value must be separated from other keywords and their assigned values by either a carriage return or a semicolon.
- ▶ All keywords and values may be abbreviated. However, they must not be abbreviated to the point to where they cannot be distinguished from other keywords and values. For example, you cannot abbreviate DENSITY to “D” because it cannot be distinguished from the DUPLICATES keyword. Abbreviating DENSITY to “DE,” however, is valid. Similarly, TYPE INTEGER can’t be abbreviated as “TYPE I,” because it cannot be distinguished from the FDL keyword value TYPE INT2. We suggest that you use the full keywords and values for readability.

## XDL keywords

An XDL file must be composed of the following keywords and their values according to the rules listed in [“Rules for XDL keyword files”](#):

### **FILE**

Indicates the beginning of the XDL description. No value should be supplied.

### **NAME** *filename*

The name of the ISAM file you want to create.

*filename*

This value is ignored when the XDL description comes from the OPEN statement. Instead, the filename in the OPEN statement is used. However, this value is used when the XDL description comes from **bldism**. The default extension is **.ism**.

### **ADDRESSING** *file\_size*

(optional) The address length of the ISAM file.

*file\_size*

Possible values are

**32** 32-bit signed file address for a maximum individual file size of 2Gb.  
(default)

**/NO/40** or **/NO/48** 48-bit file address for a maximum individual file size of 256Tb  
(equivalent to TBYTE in the parameter form). When creating a Revision  
4 file, either value generates a 40-bit maximum file size of 1Tb. NO40  
or NO48 prohibits an implicit TBYTE setting.

### **PAGE\_SIZE** *page\_size*

(optional) The size of the index blocks.

*page\_size*

The size, in bytes, of the index blocks. Possible values are

**512**  
**1024**  
**2048**  
**4096**  
**8192**  
**16384**  
**32768**

The default is 4096 for Revision 6 files (and 1024 for Revision 4 files).

**KEYS** *num\_keys*

(optional) The number of keys in the ISAM file.

*num\_keys*

You can specify between 1 and 255 keys. If not specified, the number of keys is calculated by counting the number of supplied key definitions.

**ERASE\_ON\_DELETE** *yes|no*

(optional) Controls what will happen to a record when it's deleted. A value of **yes** causes the record to be erased (i.e., nulled out), and a value of **no** causes records that are deleted to be marked deleted, but their contents remain in the file until the space is physically reused. The default is **no**. ERASE\_ON\_DELETE cannot be used if TRACK\_CHANGES is set.

**NETWORK\_ENCRYPT** *yes|no*

(optional) Specifies whether the network encryption flag is enabled for this file. The default is **no**.

**ROLLBACK** *yes|no*

(optional) Control the rollback function. A value of **yes** allows it, and a value of **no** prohibits it. The default is **yes**.

**STORED\_GRFA** *yes|no*

(optional) Generate and store the CRC-32 part of an RFA to each record header on each STORE or WRITE.

**SIZE\_LIMIT** *max\_size*

(optional) Specifies the size the **.is1** file is allowed to reach. If a STORE or WRITE operation exceeds this limit, an "Output file is full" error (FILFUL) occurs.

*max\_size*

The maximum number of megabytes the **.is1** file is allowed to reach. If a STORE operation exceeds this limit, a "File record limit exceeded" error (RECLIMIT) occurs.

**RECORD\_LIMIT** *recs*

(optional) Indicates that the record limit follows.

*recs*

The maximum number of records allowed in the **.is1** file. If a STORE operation exceeds this limit, a "File record limit exceeded" error (RECLIMIT) occurs.

**TRACK\_CHANGES** *yes|no*

(optional) Specifies whether change tracking is enabled. The default is **no**.

**TEXT** *“text\_string”*

(optional) Specifies text to add to the header of a file being created.

*text\_string*

The text string to add to the file header. To specify larger text strings, use multiple TEXT lines, which will be concatenated together to represent a single logical line. The total amount of text you can specify is unlimited.

```
TEXT  "Text is concatenated when more than one TEXT line is used "  
TEXT  "so a larger text string can be added. IPAR outputs these "  
TEXT  "quoted strings at 50 bytes."
```

**TEXT\_ALLOCATION** *text\_size[K]*

(optional) The amount of space to allocate for user-defined text in the file header, in bytes (rounded to the nearest kilobyte) or kilobyte blocks (if you specify the “K”). For example,

**TEXT\_ALLOCATION=1024** allocates text space of 1KB (1024 bytes).

**TEXT\_ALLOCATION=100** allocates text space of 1KB (1024 bytes).

**TEXT\_ALLOCATION=1025** allocates text space of 2KB (2048 bytes).

**TEXT\_ALLOCATION=2K** allocates text space of 2KB (2048 bytes).

If TEXT is specified but TEXT\_ALLOCATION is not, the default allocation size is either 32 bytes or the size of *text\_string* if the concatenated *text\_string* is larger than 32 bytes.

**DENSITY** *file\_density*

(optional) The packing density of the index blocks. If specified, the indexes are packed to the specified *file\_density* when possible.

*file\_density*

The minimum value is 50. The maximum value is 100. The default value when DENSITY is not specified is similar to the value of 50.

**RECORD**

Indicates the beginning of the record definition. No value should be supplied.

**SIZE** *rec\_size*

The size of the data records.

*rec\_size*

Consult the [“Synergy ISAM capacities and limits” table on page 3-20](#) for the minimum and maximum possible values. In Synergy .NET and 64-bit traditional Synergy, if you create a file with a variable record type and a maximum record size of 0, the records in that file can exceed 64K and are limited only by the amount of memory you have available or 2 GB.



**FORMAT** *rec\_format*

(optional) Specifies the format of the records.

*rec\_format*

Possible values are

<b>fixed</b>	The records are of fixed length. (default)
<b>multiple</b>	The records are of multiple fixed length.
<b>variable</b>	The records are of variable length.

**COMPRESS\_DATA** *yes|no*

(optional) Specifies whether the record data will be compressed. The default is **no**.

**STATIC\_RFA** *yes|no*

(optional) Ignored on REV 6 and higher files because it is set automatically. Specifies whether the records have constant RFAs over the life of the specified ISAM file.

**PORT\_INT** *pos:len*

(optional) Specify nonkey integer data within a record. One or more PORT\_INT keywords and their assigned values may be specified.

*pos*

The starting position of the nonkey integer data within the record.

*len*

The length of the nonkey integer data. Possible values are 1, 2, 4, and 8. The length may not cause the data to overlap any other nonkey integer data sections or keys.

**KEY** *n*

Indicates the beginning of a key definition.

*n*

The key number. This number must start at 0, and the following key numbers must be sequential.

**START** *pos\_1[:pos\_n]*

The 1-based starting position of the key or the first segment of the key. You can specify up to eight positions, separated by colons. You must specify a start position for each segment defined.

*pos\_1*

The starting position of the key or the first segment of the key.

*pos\_n*

(optional) The starting position of additional key segments, if any exist.

**LENGTH** *len\_1[:len\_n]*

The length of the key. You can specify up to eight lengths, separated by colons. You must specify a length for each segment defined.

*len\_1*

The length of the key or the first segment of the key. The key can be up to 254 characters long on Windows and UNIX (251 if the key allows duplicates or 250 if the key allows duplicates and this is a terabyte file), or 255 characters on OpenVMS.

*len\_n*

(optional) The length of additional key segments, if any exist.

**TYPE** *type\_1[:type\_n]*

The type of the key. You can specify up to eight types, separated by colons. You may specify only *type\_1* if all segments are the same type; otherwise, you must specify a type for each segment defined.

*type\_1*

The type of the key or the first segment of the key. Possible values are

<b>alpha</b>	Alphanumeric type (default)
<b>integer</b>	Native integer type
<b>decimal</b>	Zoned decimal type
<b>unsigned</b>	Native unsigned integer type
<b>nocase</b>	Case-insensitive alphanumeric type
<b>sequence</b>	Sequence auto type
<b>timestamp</b>	Timestamp auto type
<b>ctimestamp</b>	Ctimestamp auto type

*type\_n*

(optional) The type of additional key segments, if any exist.

**ORDER** *order\_1[:order\_n]*

(optional) The sorting order of the key data. You can specify up to eight orders, separated by colons. You may specify only *order\_1* if all segments have the same order; otherwise, you must specify an order for each segment defined.

*order\_1*

The sorting order of the key or the first segment of the key. Possible values are

<b>ascending</b>	The sorting order is ascending. (default)
<b>descending</b>	The sorting order is descending.

*order\_n*

(optional) The sorting order of additional key segments, if any exist.

**NAME** *key\_name*

(optional) The named key of reference for Synergy ISAM.

*key\_name*

The name of the key. If the name contains spaces, it must be enclosed in quotation marks.

**DUPLICATES** *yes|no*

(optional) Specifies whether duplicate keys are allowed. The default is **no**.

**DUPLICATE\_ORDER** *dup\_ord*

(optional) Specifies whether records that contain duplicate keys will appear at the end or at the beginning of a list of matching records.

*dup\_ord*

Possible values are

**fifo**      Newer records appear at the end of the list. (default)

**lifo**      Newer records appear at the beginning of the list.

**MODIFIABLE** *yes|no*

(optional) Specifies whether the key is modifiable. This is not allowed on the primary key. The default is **no**.

The packing density for all keys defaults to this *file\_density* value unless a different density value is specified on individual keys.

**NULL** *null\_type*

(optional) Specifies that the key is a null key. In this context, a null key means that when a record is stored and the specified key matches the null key value, no entry is placed in the index, thus saving file space, I/O, and processing. You can specify null keys on alternate keys only, not on a primary key.

*null\_type*

Possible values are

**replicate**      Specifies that *null\_val* is a single character and must match each byte of the specified key. Numeric key segments always match on 0 (binary 0) for unsigned and integer and "0" (decimal zero) for decimal.

**noreplicate**      Specifies that *null\_val* is a string that must match the key, from the beginning of the key and for the length of the string. For numeric keys, the string must represent a numeric value.

**short** Specifies that the key won't be stored if the record does not include the entire key on a STORE or WRITE. The file must be defined to have variable-length records.

#### **VALUE\_NULL** *null\_val*

(optional) The null value for a null key.

*null\_val*

The null value can be specified as either a single character or a string. If the null is replicating, the value refers to alpha segments only. Numeric key segments are always defined as their null or zero value and cannot be changed. If the key is specified as a nonreplicating null key, the allowable value depends on the type of the key:

- ▶ If the key is alphanumeric, an alpha string must be specified for the null value. If that key is segmented, the length of the alpha string must not cause the value to overlap a numeric segment.
- ▶ If the key is numeric and **noreplicate** was specified, a string that represents a numeric value is specified for the null value. The key may not be segmented. The allowable numeric values depend on the type and length of the key.

#### **DENSITY** *key\_density*

(optional) The index packing density for the current key.

*key\_density*

The minimum value is 50. The maximum value is 100. The default value is similar to the value of 50. This *key\_density* value overrides the *file\_density* value (for this key only) if specified.

## Examples

! This is a sample file containing a valid XDL description

#### FILE

```
NAME           sample
ADDRESSING     48
PAGE_SIZE      4096
NETWORK_ENCRYPT no
TRACK_CHANGES no
KEYS           2
TEXT           "APK 3.01"
TEXT_ALLOCATION 1K
```

#### RECORD

```
SIZE           110
FORMAT         fixed
COMPRESS_DATA  yes
ERASE_ON_DELETE yes
STATIC_RFA     yes
PORT_INT       100:4
PORT_INT       104:4
```

```

KEY 0
  START      1:16
  LENGTH     15:4
  TYPE       alpha:integer
  ORDER      ascending:descending
  NAME       "Customer"
  DUPLICATES no
  MODIFIABLE no

KEY 1
  START      31
  LENGTH     30
  TYPE       nocase
  NAME       "Name"
  DUPLICATES yes
  DUPLICATE_ORDER fifo
  MODIFIABLE no

```

Here is the same XDL description in a string format appropriate for using in the OPEN statement. Note that the keywords are separated by semicolons:

```

FILE; NAME sample, ADDRESSING 48; PAGE_SIZE 4096; NETWORK_ENCRYPT no;
TRACK_CHANGES no; KEYS 2; TEXT 2; TEXT_ALLOCATION 1K; RECORD; SIZE 110;
FORMAT fixed; COMPRESS_DATA yes; STATIC_RFA yes; PORT_INT 100:4;
PORT_INT 104:4; KEY 0; START 1:16; LENGTH 15:15; TYPE alpha:integer;
ORDER ascending:descending; NAME Customer; DUPLICATES no; MODIFIABLE no;
KEY 1; START 31; LENGTH 30; TYPE nocase; NAME Name; DUPLICATES yes;
DUPLICATE_ORDER fifo; MODIFIABLE no

```

## Correspondence to FDL keywords

The following table lists the XDL keywords and their corresponding FDL keywords.



Synergy ISAM (Windows and UNIX) recognizes both XDL and FDL forms. OpenVMS, however, does not recognize the XDL forms.

XDL keyword	FDL keyword
FILE	FILE
NAME <i>filename</i>	NAME <i>filename</i>
ADDRESSING <i>file_size</i>	—
PAGE_SIZE <i>page_size</i>	—
KEYS <i>num_keys</i>	—

XDL keyword	FDL keyword
ERASE_ON_DELETE yes no	—
NETWORK_ENCRYPT yes no	—
SIZE_LIMIT <i>max_size</i>	—
RECORD_LIMIT <i>recs</i>	—
ROLLBACK yes no	—
STORED_GRFA yes no	—
TRACK_CHANGES yes no	—
TEXT " <i>text_string</i> "	—
TEXT_ALLOCATION <i>text_size</i> [K]	—
DENSITY <i>file_density</i>	—
RECORD	RECORD
SIZE <i>rec_size</i>	SIZE <i>rec_size</i>
FORMAT <i>rec_format</i>	FORMAT <i>rec_format</i>
COMPRESS_DATA yes no	DATA_RECORD_COMPRESSION yes no
STATIC_RFA yes no	—
PORT_INT <i>pos:len</i>	—
KEY <i>key_num</i>	KEY <i>key_num</i>
START <i>pos_1[:pos_n]</i>	POSITION <i>fdl_pos</i> (unsegmented) or SEG1_POSITION <i>fdl_pos</i> (segmented) SEG <i>n</i> _POSITION <i>fdl_pos</i> where <i>fdl_pos</i> is the starting position of the key or segment.
LENGTH <i>len_1[:len_n]</i>	LENGTH <i>fdl_len</i> (unsegmented) or SEG1_LENGTH <i>fdl_len</i> (segmented) SEG <i>n</i> _LENGTH <i>fdl_len</i> where <i>fdl_len</i> is the length of the key or segment.

XDL keyword	FDL keyword
TYPE ALPHA ORDER ASCENDING TYPE ALPHA ORDER DESCENDING TYPE INTEGER ORDER ASCENDING LENGTH 2 LENGTH 4 LENGTH 8 TYPE INTEGER ORDER DESCENDING LENGTH 2 LENGTH 4 LENGTH 8 TYPE UNSIGNED ORDER ASCENDING LENGTH 2 LENGTH 4 LENGTH 8 TYPE UNSIGNED ORDER DESCENDING LENGTH 2 LENGTH 4 LENGTH 8 (There are no equivalents for the zoned decimal and case-insensitive alphanumeric types or multiple segment types.)	TYPE STRING  TYPE DSTRING   TYPE INT2 TYPE INT4 TYPE INT8  TYPE DINT2 TYPE DINT4 TYPE DINT8  TYPE BIN2 TYPE BIN4 TYPE BIN8  TYPE DBIN2 TYPE DBIN4 TYPE DBIN8
ORDER <i>order_1[:order_n]</i>	—
NAME <i>key_name</i>	NAME <i>key_name</i>
DUPLICATES yes no	DUPLICATES yes no
DUPLICATE_ORDER <i>dup_ord</i>	—
MODIFIABLE yes no	CHANGES yes no

XDL keyword	FDL keyword
NULL <i>null_type</i>	NULL_KEY yes no <b>yes</b> The key is null. The null type defaults to replicate. <b>no</b> The key is not a null key.
VALUE_NULL <i>null_val</i>	NULL_VALUE <i>null_val</i>
DENSITY <i>key_density</i>	INDEX_FILL <i>key_density</i>

## XDL syntax checker utility

The **xdlchk** utility flags all unrecognized keywords. It was created because XDL processing must ignore any keywords it doesn't recognize, because they may be FDL keywords that aren't part of the XDL definition. However, if an unrecognized keyword is actually a misspelled XDL keyword, the ISAM file may be created incorrectly. After you run **xdlchk**, it is your responsibility to distinguish between valid FDL keywords and misspelled XDL keywords.

The **xdlchk** utility scans through the specified file and checks each keyword against the keywords in the XDL definition. It has the syntax

```
xdlchk [-f] filename
```

**Xdlchk** runs in one of two modes, depending on whether or not the **-f** option is specified:

If -f is	xdlchk verifies that
Not specified (default)	The file is a valid XDL file. Any keyword that is <i>not</i> a valid XDL keyword (including keywords in the defined subset of FDL keywords) generates a warning. You can use this mode to verify that any unrecognized keywords are actually FDL keywords that are not in the XDL definition.
Specified	An FDL file is a valid XDL file. This mode verifies only FDL keywords that are part of the XDL definition. It generates an error if it finds any other XDL keyword. It ignores any keyword it doesn't recognize, assuming that it is an FDL keyword that is not part of the XDL definition. You can use this mode to verify that an existing FDL file will pass through the XDL processing without any errors. <b>Note:</b> This mode only checks that you are not using XDL keywords; it does not validate an FDL file! If you are going to share FDL files between OpenVMS RMS and Synergy ISAM, we suggest that you first create the FDL file for use with RMS.



# Moving Database Files to Other Systems

Some of the sections below refer to endian type. If you do not know what endian type your machine is, see [“Big-endian and little-endian”](#) in the “UNIX Development” chapter of your *Professional Series Portability Guide*.

## Moving ISAM database files to an ISAM machine

- ▶ Simply transfer the files between machines (in binary mode, if you’re using FTP).

If the machines have different endian types, you can only transfer files in this manner if your records do not contain integer data. If your records contain integer data, see [“Using integer data in your records and moving between endian machines”](#) below.

## Moving files from RMS to ISAM

1. Unload your files to sequential files using **isload** or CONVERT/FDL.
2. Transfer them to the other machine.
3. Reload the file on the target machine using **fconvert** (on Windows and UNIX) or **isload** or CONVERT/FDL (on OpenVMS).

Do not use FTP in binary mode on the sequential output files.

## Using integer data in your records and moving from ISAM little endian to or from RMS

1. Unload and then reload your files using **fconvert** (on Windows and UNIX) or **isload** or CONVERT/FDL (on OpenVMS) and counted format on both your source and target machines.
2. Transfer the counted output file in binary mode.

## Using integer data in your records and moving between endian machines

You must write the conversion routines yourself. See [%CNV\\_IP](#) and [%CNV\\_PI](#) in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual* for subroutines that enable you to convert integer data to portable form and vice versa.



# 4

## General Utilities

### **The Synergy UI Toolkit Control Panel 4-3**

Describes how to use the Synergy UI Toolkit Control Panel to translate or modify message text.

### **The Synergy DBL Profiler 4-11**

Explains how to use the Synergy DBL Profiler to profile routines in the files being compiled.

### **The Synckini Utility 4-14**

Describes the **synckini** utility, which reports on the location of the **synergy.ini** and **synuser.ini** files that will be used.

### **The Servstat Program 4-15**

Describes the **servstat** program, which monitors the OpenVMS Synergy/DE xfServer.

### **The Monitor Utility for Windows 4-21**

Describes the Windows Synergy/DE xfServer monitor feature.

### **The Monitor Utility for UNIX 4-23**

Describes the UNIX Synergy/DE xfServer monitor feature.

### **The ActiveX Diagnostic Utility 4-31**

Discusses the ActiveX Diagnostic utility, which tests ActiveX controls to see if they load properly.

### **The Synbackup Utility 4-34**

Describes the **synbackup** utility, which provides a way for all cooperating processes to freeze update I/O while Synergy databases are being backed up.

### **The Synergy Prototype Utility 4-39**

Discusses the Synergy Prototype utility, which generates prototypes for classes and their member subroutines, functions, properties, structures, and nested classes.

### **The Variable Usage Utility 4-44**

Describes the Variable Usage utility, which identifies unused variables or variables used by a local routine.

### **The Gennet40 Utility 4-46**

Describes the **gennet40** utility, which generates Synergy classes that wrap the classes defined in a .NET assembly.

### **The Dbl2xml Utility 4-51**

Discusses the **dbl2xml** utility, which processes Synergy DBL source files that include language attributes, parameter modifiers, and comments, and outputs an XML file containing interfaces and methods.

# The Synergy UI Toolkit Control Panel



The Synergy UI Toolkit Control Panel (**synctl**) enables you to translate or otherwise modify message text in Synergy/DE products. All Synergy/DE error and screen messages (including the information line and application titles) reside in the file **syntxt.ism**. The only error message that is hard-coded into the Synergy/DE system is the “No message file found” message, for obvious reasons.

You can modify messages in one of three ways:

- ▶ Editing the messages interactively using the Synergy UI Toolkit Control Panel’s “Interactive mode”
- ▶ Unloading the messages to an ASCII file and then reloading them to an ISAM file
- ▶ Using the Synergy UI Toolkit Control Panel’s command line interface

## Before you begin

Before using the Synergy UI Toolkit Control Panel to change Synergy messages, make a copy of the **syntxt.ism** and **syntxt.is1** files.

We also recommend that you print a list of the messages in **syntxt.ism** before you translate or modify anything, so you’ll have a reference of what you’re changing. To do so,

1. Follow steps 1 through 3 in [“Making major changes: Unloading messages to an ASCII file” on page 4-6](#).
2. Print the file that is created.

If you want to use **syntxt.ism** from a directory other than DBLDIR, set the SYNTXT environment variable to the desired directory.



When customizing error and informational messages:

- ▶ Do not remove any Synergy messages.
  - ▶ Be careful when changing messages that contain “%” followed by one or two characters (for example, c, d, ld, s, or u). These specifiers are replaced when the message is generated. If you plan to change any of these messages, do not remove the “%” specifier. If there is more than one specifier, the order of the specifiers is fixed and cannot be changed.
-

Making minor changes: Editing messages interactively

For small changes, you can use the Modify messages function of the Synergy UI Toolkit Control Panel to interactively modify the text message file **syntxt.ism**. Run Synergy UI Toolkit Control Panel as follows:

On	Do this
Windows	From the Windows Control Panel, select Synergy Control Panel > Synergy UI Toolkit Control Panel or go to a command prompt and type <code>dbr DBLDIR:syntcl</code>
UNIX	Type <code>dbr DBLDIR:syntcl</code>
OpenVMS	Type <code>run DBLDIR:syntcl</code>

To modify one message at a time,

1. Print a list of messages as instructed in “Before you begin” on page 4-3.
2. From the Text messages menu, select Interactive mode.
3. At the Message Library prompt, enter the name of your text message library. (Synergy/DE messages are in **DBLDIR:syntxt**, which is the default.)
4. From the Interactive menu, select Modify messages. (You can also add a new message or delete a message by selecting Insert messages or Delete messages.)

A dialog box is displayed. See [figure 4-1](#).

Enter the name of the facility in which you want to change messages

Enter the number of the message you want to change

Enter the message's mnemonic

Type your changes and press ENTER

Facility Name :

Message Number :

Mnemonic :

----- Message Text -----

Figure 4-1. Modifying text messages.

5. Display the message you want to change in any of the following ways:
  - ▶ Use the Find, Last, Prev, and Next entries in the Search menu.
  - ▶ Enter the facility name in the Facility Name field and press the Find shortcut (CTRL+F). Press F3 or F2 to page forwards or backwards through the messages until you find the one you want to change. Synergy/DE messages have the following facility codes:

<b>Facility Codes</b>	
<b>Application</b>	<b>Facility</b>
UI Toolkit	DTK
Proto	PRO
Composer	CPS
Repository	RPS
ReportWriter	RPT
Report Definition Language	RDL
Compiler	CMP
Linker	LNK
Librarian	LBR
Runtime	RNT

- ▶ Look up the number of the message you want to modify on your printout of messages. Enter the message facility in the Facility Name field and its number in the Message Number field, and press CTRL+F.
  - ▶ Look up the mnemonic of the message you want to modify on your printout of messages. Enter the name of the message facility in the Facility Name field, 0 in the Message Number field, and the desired search string in the Mnemonic field. Then press CTRL+F.
6. Make your changes in the message that's displayed in the Message Text field, and press ENTER.

## Making major changes: Unloading messages to an ASCII file

If you are making major changes or translating text into another language, you can make your changes by unloading **syntxt.ism** to a sequential ASCII file, modifying the file, and then reloading it to an ISAM file.

1. In Synergy UI Toolkit Control Panel, select Unload messages from the Text messages menu.
2. From the Unloads menu, select Sequential file. (If you're unloading Synergy DBL messages—Compiler, Linker, Librarian, or Runtime—select C header file from the Unloads menu.) A dialog box is displayed. See [figure 4-2](#).
3. Enter the desired information in each input field and press ENTER to unload the messages.
4. Using your favorite text editor (for example, Wordpad on Windows), open the data file that's created and modify the messages you want changed. Make sure you save the file as a text file.



Don't ever modify information in the first 18 bytes of the record.

Enter the name of the text message library in which the messages are stored. The default extension is **.ddf** for a sequential file or **.h** for a C header file

Enter the name of the facility you want to select

Enter the name of the sequential data file to unload the messages to

Select whether you want to specify a particular facility (as opposed to all facilities)

*Figure 4-2. Unloading messages.*

5. In Synergy UI Toolkit Control Panel, select Remove messages from the Text messages menu to clear messages from the **syntxt.ism** file.
6. Enter the desired information in each input field in the displayed dialog box, and press ENTER to remove the messages.
7. To load **syntxt.ism** from your modified sequential ASCII file, select Load messages from the Text messages menu.



8. From the Loads menu, select Sequential file.
9. Enter the desired information in each input field in the displayed dialog box, and press ENTER to load the messages.



Your distribution includes the **dtktxt.ddf**, **protxt.ddf**, **wdtxt.ddf**, **rpstxt.ddf**, **rpttxt.ddf**, and **rdltxt.ddf** files, which contain the unloaded Toolkit, Proto, Repository, ReportWriter, and Report Definition Language text messages, respectively. If your **syntxt.ism** file becomes corrupted, follow step 4 using these **.ddf** files as your sequential input files.

---

## Creating your own text message files

If you want to create new messages, you can either add your messages to **syntxt.ism** or you can create a new message library. To create and use your own text message library,

1. In Synergy UI Toolkit Control Panel, select Create new library from the Text messages menu.
2. At the Message Library prompt, enter the name of the library file you'd like to create, and press ENTER. The default extension is **.ism**.
3. Add messages to your new library either by inserting them interactively using the Interactive mode and Insert messages commands or by loading an existing sequential file using the Load messages command.
4. Generate your message definitions to a definition file by selecting Generate defines from the Text messages menu.

A dialog box is displayed. See [figure 4-3](#).

5. **.INCLUDE** the definition file in your application.

You can use the **U\_GETTXT** subroutine to retrieve lines of text from the text message library. For the subroutine syntax, see [U\\_GETTXT](#) in the “Utility Routines” chapter of the *UI Toolkit Reference Manual*.

## General Utilities

### The Synergy UI Toolkit Control Panel

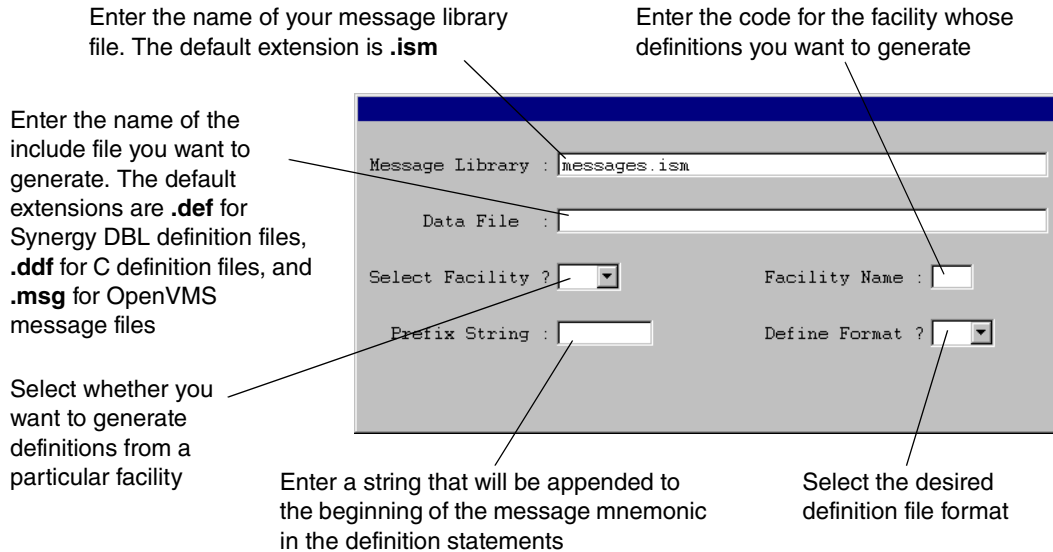


Figure 4-3. Generating definitions to an include file.

## Modifying messages at the command line

You can modify messages at the command line without invoking the Synergy UI Toolkit Control Panel's menu interface by running **synctl** and specifying one or more options. For example, on Windows and UNIX, you'd enter

```
dbr DBLDIR:synctl [option ...]
```

On OpenVMS, add DBLDIR: to your DCL\$PATH search list logical, which is analogous to UNIX's PATH.

### Arguments

*option*

(optional) One or more of the following command line options:

**-?**

Display a help screen of valid command line options.

**-a file**

Append unloaded messages or generated definitions to the end of an existing sequential or C header file, where *file* is the name of the existing output file. If specified, this option must precede the specification of the output file.

**-c library**

Create a message library, where *library* is the name of the library you want to create.

**-d library [-f facility]**

Delete records in a message library, where *library* is the name of the library from which you are deleting records and *facility* is an optional code for the category of messages you are deleting.

**-l library seq\_file [-f facility]**

Load the message library from a sequential file, where *library* is the name of the library to which you are loading messages, *seq\_file* is the name of the sequential file from which the messages are being loaded, and *facility* is an optional code for the category of messages you are modifying.

**-l library -h header\_file [-f facility]**

Load the message library from a C header file, where *library* is the name of the library to which you are loading messages, *header\_file* is the name of the C header file from which messages are being loaded, and *facility* is an optional code for the category of messages you are modifying.

**-u library seq\_file [-f facility]**

Unload messages from a library to a sequential file, here *library* is the name of the library from which you are unloading messages, *seq\_file* is the name of the sequential file to which you are unloading messages, and *facility* is an optional code for the category of messages you are modifying.

**-u library -h header\_file [-f facility]**

Unload messages from a library to a C header file, where *library* is the name of the library from which you are unloading messages, *header\_file* is the name of the C language header file to which you are unloading messages, and *facility* is an optional code for the category of messages you are modifying.

**-g library definition\_file [-f facility] [-p prefix]**

Generate a Synergy DBL definition file, where *library* is the name of the library from which the message definitions are being extracted, *definition\_file* is the name of the Synergy DBL definition file you are generating, *facility* is an optional code for the category of messages you are modifying, and *prefix* is an optional prefix for the message mnemonic in the definition file.

**-g library -h header\_file [-f facility] [-p prefix]**

Generate a C header file, where *library* is the name of the library from which messages are being extracted, *header\_file* is the name of the C header file you are generating, *facility* is an optional code for the category of messages you are modifying, and *prefix* is an optional prefix for the message mnemonic in the header file.

## Discussion

The default filename extensions for the different types of files specified above are as follows:

<b>.ism</b>	Message libraries
<b>.ddf</b>	Sequential files
<b>.h</b>	C header files
<b>.def</b>	Synergy DBL definition files

The facility names for Synergy messages are listed in the [“Facility Codes” table on page 4-5](#).

---

### VMS

Synergy DBL does not use the Synergy UI Toolkit Control Panel directly, and the **syntxt.ism** file distributed with Synergy DBL does not have the linker and librarian errors loaded into it. As a result, when customizing those messages on OpenVMS, you must perform a few extra steps before your changes are reflected. Enter the following at the command line:

```
$ syntcl:==$dbldir:syntcl.exe
$ syntcl -g all_errors.msg -v DBLDIR:syntxt.ism
$ message/nosymbols/obj=all_errors.obj all_errors.msg
$ link/shareable=sys$message:dblmf.exe all_errors.obj
$ install replace sys$message:dblmf/open/header/share
```

---

## Examples

The following example unloads the runtime messages from the Synergy message library, **syntxt.ism**, into a sequential file named **mymsg.ddf**.

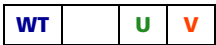
```
dbr syntcl -u DBLDIR:syntxt mymsg -f RNT
```

The next example creates a message library named **mytxt.ism**, loads it with messages from the sequential file **mytxt.ddf**, and generates the definition file **mymsg.def**. It specifies the facility **DTK** and the prefix **DTK\_**.

```
dbr syntcl -c mytxt -l mytxt mytxt -g mytxt mymsg -f DTK -p DTK_
```

When combining operations as in this example, all the arguments for each option must be respecified, or the default values are used. By default, all facilities are used, and no prefix is used.

# The Synergy DBL Profiler



By providing profiling information about your programs, the Synergy DBL Profiler helps you determine where and how you can best optimize them. A profiled routine counts the CPU time it uses (in ticks), including any system calls, and it counts the number of times the routine is XCALLED. It also shows I/O on Windows and both I/O and page faults on OpenVMS.

Depending on which system options you’ve set, profiling can also include any Synergy DBL subroutines that are XCALLED by the current routine. In the latter case, the total CPU time for a program can be counted many times, and the total CPU time is the time taken by the root module, rather than the sum of all the modules.

On UNIX and OpenVMS, the profiler calculates accumulated CPU time. On Windows, you have a choice of (low-granularity) accumulated CPU time or (high-granularity) elapsed CPU time.

To profile your routines,

1. If you are only profiling specific routines (rather than all routines), compile those routines using the profiling compiler option (**-u** on Windows and UNIX or **/profile** on OpenVMS).
2. Compile.
3. Run the resulting program.
4. Decode the **profile.dat** or **lines.dat** file that was created to get the profile results.

## Enabling profiling

You can enable profiling in one of two ways.

- ▶ To enable profiling for all routines, set system option #42. (See “[Setting an Option](#)” in the “System Options” chapter of *Environment Variables & System Options*.)
- ▶ To enable profiling for specific routines, set one of the following system options *and* specify the profiling compiler option (**-u** on Windows and UNIX or **/profile** on OpenVMS) when you compile your programs:

To profile	Use this option
The current routine if the profiling compiler option is specified	#40
The current routine and all routines XCALLED by the current routine if the profiling compiler option is specified	#41
Synergy programs at the line level. (Note that you cannot specify a list of routines to exclude when you use this option.)	#52

Running a program that was compiled with profiling enabled outputs a file called **profile.dat** (or **lines.dat** if system option #52 is set). This file is the data file that is used to get the profile. (See “[Decoding the profile.dat or lines.dat file](#)” below.) On OpenVMS, this file is written to SYS\$SCRATCH:. On other systems, it is written to the current directory.

## Decoding the profile.dat or lines.dat file

Depending on which system option was set when you compiled, either **profile.dat** or **lines.dat** was created.

To decode the **profile.dat** file, enter

```
dbr DBLDIR:profile [-x exclusion_file]
```

The **profile** program interprets **profile.dat** and outputs the results to a file called **profile.lst**. When you run **profile**, you can specify an exclusion file that contains a list of routines to exclude from the profile output. The routine names listed in this text file can be on one or more lines and must be separated by commas. (Although the exclusion file option is available on all platforms, it provides the greatest benefit on Windows, due to the granularity of the measurements.) If the **-x** option is not specified, the profiler uses the default exclusion file, **tkexclude.txt**, which is distributed with UI Toolkit in the directory pointed to by the WND environment variable. This file is provided because when elapsed CPU time is profiled on Windows, Toolkit input routines such as I\_INPUT\_P can consume a large amount of time. Excluding the Toolkit input routines enables you to get a more accurate idea of how much CPU time is actually being used.

To decode the **lines.dat** file, enter

```
dbr DBLDIR:proflines
```

The **proflines** program interprets **lines.dat** and outputs the results to a file called **lines.lst**, which is sorted according to which statements are used most often.

## Keep in mind...

Routine profiling is only as accurate as the CPU times posted to your process by the operating system. Especially on faster systems, misleading results can be generated. On a fast CPU, 10 millisecond ticks encompass a lot of Synergy DBL instructions.

---

### VMS

The DIO count includes any I/O required to write the **profile.dat** file, which can account for DIO counts in routines in which no I/O occurs. In addition, the I/O is only updated every 10 milliseconds.

---

---

## WIN

The Synergy DBL 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. (For more information, see [PROFILE\\_PROCESSOR\\_TIME](#) in the “Environment Variables” chapter of *Environment Variables & System Options*.) 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 or on a multi-processor or hyperthreaded CPU.

---

# The Synckini Utility



The **synckini** utility reports the setting of SFWINIPATH and the location of the **synergy.ini** and **synuser.ini** files that Synergy/DE will access, and prompts you as to whether you want to edit one or both files.

For example:

```
SFWINIPATH = c:\synergyde\dbl
Full path for synergy.ini is:
c:\synergyde\dbl\synergy.ini
file exists and can be opened
Edit c:\synergyde\dbl\synergy.ini (y/n)?
```

To edit the specified **synergy.ini** file, type **y**. The **synckini** utility launches the registered editor for **.ini** files (Notepad by default) with the **synergy.ini** file loaded. It continues to report on **synuser.ini**.

For example:

```
Full path for synuser.ini is:
C:\WINDOWS\Application Data\Synergex\synuser.ini
file exists and can be opened
Edit C:\WINDOWS\Application Data\Synergex\synuser.ini (y/n)?
```

To edit the specified **synuser.ini** file, type **y**. The **synckini** utility launches a new occurrence of the default editor with the **synuser.ini** file loaded.

You can edit the files in Notepad and then save your changes and close Notepad normally.

If the **synergy.ini** file cannot be located or opened, **synckini** reports an error. If the **synuser.ini** file cannot be located or opened, **synckini** gives you the option to create the file by prompting

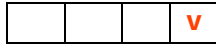
```
Create path\synuser.ini (y/n)?
```

where *path* is the Synergex subdirectory of your local application data directory (Documents and Settings\username\Local Settings\Application Data).

For more information about **synergy.ini** or **synuser.ini**, see “[Synergy initialization files](#)” in the “Environment Variables” chapter of *Environment Variables & System Options*.



# The Servstat Program



The **servstat** program enables the system manager to start, stop, and modify parameters of Synergy/DE xfServer and xfServerPlus on OpenVMS. The program may be run either on the OpenVMS host machine or on a remote machine. You can either run it interactively, letting it prompt you for options and values, or you can specify command line arguments (if it is set up as a foreign symbol).

**Servstat** checks whether the server is running. If not, it asks whether to start the server. If the answer is **no**, the program exits.

The **servstat** program checks for the existence of the connection manager by scanning all running processes for process names of the form RSDMS\$MGR\_*nnnn*, where *nnnn* is the IP port on which the server is listening. This allows more than one server to be running on the host. If more than one connection manager is found, you are prompted for which one to use.

The **servstat** menu offers the following options:

- [1]     Display status
- [2]     Change free pool size
- [3]     Purge free pool
- [4]     Shut down server
- [5]     Show server global logicals
- [6]     Show session server logicals
- [7]     Change cull interval
- [8]     Change pool extend time
- [9]     Display xfServerPlus status
- [10]    Purge xfServerPlus free pool
- [11]    Cycle xfServerPlus log file

For more detailed information about these options and the information they display, see [“Servstat options” on page 4-17](#).

Typing the EOF key sequence or entering any invalid response exits the program.

## Function of the program

**Servstat** allows the system manager to monitor the performance of the server in processing incoming connections. If the peak number of pending servers (processes started but not yet registered with the connection manager) is ever greater than the minimum free pool size, you may want to expand the free pool at the startup command line in the command file **SY\$COMMON:[SYSMGR]SYNERGY\_STARTUP.COM** to speed up connections at peak usage times. The server attempts to optimize the free pool size itself if there is a sudden onslaught of connections. If the number of pending servers ever exceeds the current free pool size, the free pool is enlarged by one process.

When the connection manger increases the free pool in this way, the free pool may stay elevated above the minimum free pool size specified in **SYNERGY\_STARTUP.COM** for the “free pool extend time.” If no connections are processed in this time, the free pool is cut back by one server. This happens until the free pool recedes back to the minimum free pool size. The frequency of free pool size checks is determined by the “Cull interval” period. This defaults to one-fifth of the free pool extend time (unless the free pool extend time is less than 15 minutes, in which case it is one-half of that time).

The free pool extend time is changed using option 8. The cull interval is changed at the same time according to the above rule. You can also change the cull interval using option 7.

## Running servstat with command line arguments

The **servstat** program can accept command line arguments if it's set up as a foreign symbol:

```
$ servstat == $SYNERGYDE$ROOT:[SERVER] servstat
```

To run **servstat** using command line arguments, enter the same values that you would enter if you were running **servstat** interactively. For example, if you want to change the pool extend time, enter

```
servstat 8 "time"
```

where *time* is the length of time the free pool is allowed to be elevated above the minimum free pool size.

The following example, which includes output, runs **servstat** with option 5, to show server global logicals:

```
$ servstat 5
```

```
Synergy server version 9.5.3
      PID      Process name      Image
1) 20402E7C  RSDMS$MGR_2331
$2$DKA0:[SYS0.SYSCOMMON.] [SYSEXE] RSynd.EXE;1
```

```
Option> 5
```

```
(LNM$RSDMS$MGR_2331)

"DBLDIR" = "_$2$DKA0:[SYNERGYDE.] [DBL] "
```

You can enter up to two arguments at the command line. If multiple servers are running, you will need to specify which of the servers are to be modified or inspected. In the following example, six servers are running, and we select the first to do option 5:

```
$ servstat 1 5
Synergy server version 9.5.3
      PID      Process name      Image
1) 21236CBD  RSDMS$MGR_2330 $6$DKA0:[SYS0.SYSCOMMON.] [SYSEXE] RSynd.EXE;1
```

```

2) 212234BE + RSDMS$MGR_2356 $6$DKA0: [SYS0.SYSCOMMON.] [SYSEXE] RSYND.EXE;1
3) 21236CBF S RSDMS$MGR_3200 $6$DKA0: [SYS0.SYSCOMMON.] [SYSEXE] RSYND.EXE;1
4) 212234C2 M RSDMS$MGR_3201 $6$DKA0: [SYS0.SYSCOMMON.] [SYSEXE] RSYND.EXE;1
5) 212234C5+S RSDMS$MGR_3205 $6$DKA0: [SYS0.SYSCOMMON.] [SYSEXE] RSYND.EXE;1
6) 21236CC8+M RSDMS$MGR_3206 $6$DKA0: [SYS0.SYSCOMMON.] [SYSEXE] RSYND.EXE;1

```

Which server (1-6) [1] 1

Option> 5

(LNM\$RSDMS\$MGR\_2330)

"DBLDIR" = "\_\$2\$DKA0: [SYNERGYDE.] [DBL] "

If multiple servers are running and the selected option requires a value, only the server number and option can be specified on the command line, and the value must be specified at the prompt. For example, you will get an error if you enter the following:

```

$ servstat 1 2 3
Too many command line arguments
Usage: servstat cmd
Usage: servstat cmd1 cmd2

```

The correct way to run **servstat** with these arguments is as follows:

```

$ servstat 1 2
Synergy server version 9.5.3
      PID      Process name      Image
1) 20402E7C   RSDMS$MGR_2331 $2$DKA0: [SYS0.SYSCOMMON.] [SYSEXE] RSYND.EXE;1
2) 20402E83 + RSDMS$MGR_2330 $2$DKA0: [SYS0.SYSCOMMON.] [SYSEXE] RSYND.EXE;1

```

Which server (1-2) [1] 1

Option> 2

New pool size: 3

## Servstat options

### [1] Display status

Option 1 displays the current status of the OpenVMS server system and the session server processes as follows:

```

      Synergy Server Manager Status at 30-JUN-2011 12:45:06.93
      -----
Encryption mode:          No
Number of active servers: 3
Peak # of active servers: 4
Servers in use:           1

```

## General Utilities

### The Servstat Program

```
Free pool:                2 - Free pool extend time:    0 00:30:00.00
Minimum free pool:        2
Free servers:             2
Number pending servers:   0
Peak # pending servers:   2
Cull interval:            0 00:06:00.00
Next cull:                30-JUN-2011 12:50:28.43
```

Pid	Status	Time at that status	User	I.P address
21236CC0	Ready	48 21:56:54.19	CSTEST	10.1.1.121
212234C1	Serving	3 03:50:53.60	CSTEST	10.1.1.121
2124A45A	Ready	3 03:50:53.42		

#### [2] Change free pool size

Option 2 manually changes the size of the free pool. When prompted, enter the new pool size as an integer. If the new number is greater than the current number, the connection manger creates new server processes. If the new number is less than the current number, the free pool is trimmed after the free pool extend time has expired. To change the pool size from the command line, enter

```
servstat 2 pool_size
```

Assuming only one server is running, the following example changes the free pool size to 3:

```
servstat 2 3
```

#### [3] Purge free pool

Option 3 purges the free pool back down to the minimum free pool size.

#### [4] Shutdown server

Option 4 closes down the server. By default, serving processes close when their clients disconnect.

#### [5] Show server global logicals

Option 5 displays all logicals from the logical name table LNM\$RSDMS\$MGR\_*nnnn*, where *nnnn* is the port on which the connection manger is listening. This table is shared by all server processes controlled by the connection manager and is used to resolve logical names after the process and job name tables have been scanned. Logicals may be added to this table by setting them in the command file **DBLDIR:SERVER\_INIT.COM**. The server startup command file runs this command file when the server process has been created. This is a more efficient way to set server-specific logicals than putting definitions in **SYNRC.COM**. We advise using the server with the /NOUSE\_SYNRC qualifier in **SYNERGY\_STARTUP.COM**.

## [6] Show session server logicals

If the server is using **SYNRC.COM**, each session server processes **SYNRC.COM** in the default directory of the connection user. The logicals defined in this directory are placed in the process-private logical name table `LNMS$SYNSVR_XXXXXXX`, where `XXXXXXX` is the PID of the session server. This table is used to resolve logicals before the server-wide name table `LNMS$RSDMS$MGR_nnnn`.

## [7] Change cull interval

Option 7 changes how often the free pool size is checked and trimmed. The time must be entered in the form *D HH:MM:SS*. For instance, 0 00:29:59 equals 29 minutes and 59 seconds. The time must be enclosed in quotation marks if you are running from the command line. For example, you would enter the following to change the cull interval to 30 minutes from the command line:

```
servstat 7 "0 00:30:00"
```

## [8] Change pool extend time

Option 8 changes how long the free pool is allowed to be elevated above the minimum free pool size while no connections have been processed. The time must be entered in the form *D HH:MM:SS* and must be in quotation marks if you are running from the command line.

## [9] Display xfServerPlus status

Option 9 displays status information about xfServerPlus. It shows the PID for each xfServerPlus process and specifies whether it is free or in use and by which IP. The output looks like this:

```
Synergy xfServerPlus Manager Status at 30-JUN-2011 12:38:42.11
-----
xfServerPlus is enabled on port 3205 as CSTEST
Encryption mode:      Slave, Cipher=High
Certificate file:      DBLDIR:rsynd.pem
xfServerPlus free pool: 2
Number of active servers: 0
Number of pending servers: 0
Servers in use:        0 - Inactive for          49 00:00:44.48
Free pool:             2 - Free pool extend time: 0 00:30:00.00
Minimum free pool:     0
Free servers:          2
Number pending servers: 0
Peak # pending servers: 2
Cull interval:         0 00:06:00.00
Next cull:             30-JUN-2011 12:44:27.13

  Pid      Status      Time at that status  Port  I.P. address
212234C3    Ready      49 00:00:40.97
212234C4    Ready      49 00:00:40.95
```

If option 9 shows that xfServerPlus is not enabled, refer to the **rsynd** log file for information about what went wrong. By default the **rsynd** log file is named **node\_rsynd\_port.log** and is located in DBLDIR.

#### [10] Purge xfServerPlus free pool

Option 10 causes all processes in the xfServerPlus free pool to be destroyed and the pool to be repopulated with new processes.

#### [11] Cycle xfServerPlus log file

Option 11 closes and then opens a new version of the xfServerPlus log file (**DBLDIR:xfpl.log**, by default). This enables you to examine the log file without shutting down **rsynd**.

# The Monitor Utility for Windows



The Monitor utility for Windows (**synxfmon.exe**) displays information about files opened by xfServer. It tells you which files are open, who opened them, and whether there are locked records within those files. It also indicates whether a connection is active or suspended and enables you to close a connection.

```
synxfmon [option] [> redirect_file]
```

## Arguments

*option*

One or more of the following options:

- k** *remote\_port*      Close (kill) the xfServer connection on the specified remote (client) port.
- n** *host*              Host name of the xfServer machine. The default is **localhost**.
- p** *port*              Port where xfServer is running. The default is 2330.
- v**                    Verbose output.

*redirect\_file*

(optional) The name of a file (including the path) to which output will be sent.

## Discussion

The Monitor utility generates a list of all files opened by users for a specific instance of xfServer and displays status information about those files. Each line of nonverbose output has the following format:

```
[lock_status] username : filename [connection_status]
```

where *[lock\_status]* is one of the following:

- [ ]      There are no locked records in the file
- [L]      One or more records in the file are locked

and *[connection\_status]* is one of the following:

- [ **Active** ]            Connection is active
- [ **Suspended** ]        xfServer is holding the context of a client that is temporarily disconnected.

The Suspended status can display only when xfServer connection recovery is enabled between the client and server. Suspended status indicates that an unexpected socket disconnect has been detected and xfServer has saved the client context awaiting reconnection. The length of time the connection has been suspended will display in HH:MM:SS format (see sample below). If the client successfully reconnects, *connection\_status* will return to Active. You can use the **-k** option to kill a suspended connection and delete the context. (For more information on the connection recovery feature, see “[Using connection recovery](#)” in the “Configuring xfServer” chapter of the *Installation Configuration Guide*.)

Valid port numbers for the **-k** and **-p** options are in the range 1024 through 65535.

The **synxfmon** utility may be run from either the client or the server. When run from a client, specify the server name with **-n**. *Host* must be a Windows server.

The **-v** option returns the remote port number of the client machine along with additional operational information, including the number of READ, STORE, etc., operations that have taken place since the file was opened. For example, **synxfmon** might yield the following output without the **-v** option:

```
[L] tiger\fred: C:\MyFiles\testfile.ism [ Suspended 00:05:22 ]
```

but if **-v** is specified, the output might look like this:

```
User: tiger\fred Port: 24785 [ Suspended 00:05:22 ]
  1 file(s) open
  [L] ISAM: C:\MyFiles\testfile.ism
      File ops:
          READ... 15
          READS... 10
          STORE... 52
```

This information can be used in conjunction with the Windows **netstat** utility on the client and server to assist in client process detection and to close connections when locks are not released.

The **-k** option kills the connection, releases all file and record locks, and deletes any saved client contexts. You must be a member of the administrator group to use this option. To obtain the remote port number to specify for **-k**, first run the Monitor utility with the **-v** option. If the xfServer connection is terminated successfully, a “Successfully closed connection on port *remote\_port*” message is generated; otherwise, you will get an error message. A client that requests I/O from xfServer after **synxfmon** has terminated its connection receives a “Server session has expired or has been terminated” error (\$ERR\_SRVEXP).

Specify a *redirect\_file* if you want output to be redirected to a file, rather than displayed to the screen.

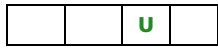


Because the **-v** option can produce a significant amount of output when multiple files are open, we recommend that you redirect output to a file when using this option.

---



# The Monitor Utility for UNIX



You can audit your client/server system's activities with the Synergy/DE xfServer's Monitor feature.

With the Monitor running, you can display the following:

- ▶ Server information
- ▶ Client information
- ▶ Accessed file information
- ▶ Error information (see the note below)

## When to use the Monitor

You may want to run the Monitor whenever you run Synergy/DE xfServer, or run it only when you need to troubleshoot any problems that may be occurring on the system. Depending on your client/server setup, running with the Monitor always turned on may decrease your system's performance.

The primary purposes for running the Monitor are the following:

- ▶ To troubleshoot any existing client/server problems
- ▶ To audit the existing performance so you can determine any actions that could further maximize performance



---

The error information displayed by the Monitor is related to the trappable runtime errors caused by your Synergy application (\$ERR\_BADKEY, \$ERR\_EOF, \$ERR\_FNF, etc.).

---

## Running the Monitor

To run the Monitor, specify one of the following option combinations on the **rsynd** command line:

- ▶ **-m**
- ▶ **-m -l**
- ▶ **-m -t**

These options are also listed in the [“QUERY syntax” table on page 4-25](#).

### -m option

```
rsynd -m
```

The **-m** option

- ▶ starts the Monitor.
- ▶ maintains information on client/server activities of up to 300 accessed files in the xfServer's memory.

### -m -l option

```
rsynd -m -l debugfile
```

The **-m -l** option

- ▶ starts the Monitor.
- ▶ writes detailed client information to the specified text file (*debugfile*) every time a client accesses the server.



You need to keep track of *debugfile*'s size. It can quickly grow very large and might fill your disk.

---

*Debugfile* is located in /usr/tmp (or /var/tmp if /usr/tmp can't be found). If you are using an alternate port, the port number is appended to the filename. For example, **/usr/tmp/rmoncore** is created by the default **rsynd** running on port 2330, and **/usr/tmp/rmoncore.2345** is created for the **rsynd** running on port 2345.

### -m -t option

```
rsynd -m -t minutes
```

The **-m -t** option

- ▶ starts the Monitor.
- ▶ writes summarized statistical information to a binary file (**rmoncore**) at intervals of the specified number of minutes. (See the discussion of query's **-a** option in the [“QUERY syntax” table on page 4-25](#) if you want to specify a different amount of time after starting the Monitor.) If **-t minutes** is not specified, the default is every 20 minutes.

The **-m -t** option is necessary if you plan to use the **-v** option when you query for client/server information. (See the [“QUERY syntax” table on page 4-25](#) for information on the **-v** query option.)



If you use this option, you should keep track of the size of the **rmoncore** file. It could fill your disk if left running for a long period of time.

---

## Displaying Monitor information

Once you have started the Monitor program, you can request information about the system using the QUERY command with any of the available options.

The following table lists the QUERY syntax for each type of query available.

QUERY syntax	
Command	What it gives you
query or query -g	<i>Global information.</i> This is information such as start times (both xfServer and Monitor), the total number of packets that have been received and/or sent, and total bytes that have been received or sent.
query -a <i>min</i>	<i>Alarm value change.</i> The value passed in <i>min</i> is the number of minutes between each write to the <b>rmoncore</b> file. For example, if you enter <code>query -a 60</code> , the Monitor writes to <b>rmoncore</b> every 60 minutes. If <i>min</i> has a value of 0, the timed logging to <b>rmoncore</b> (the <b>-t</b> option) is turned off. If you pass a nonzero value when the timed logging is off, timed logging will be turned on (even if <b>-t</b> was not specified).
query -c	<i>Clear</i> (reset) Monitor global variables.
query -e	<i>Error log.</i> Shows the number of packets that were returned with error status instead of performing the client request.
query -f	<i>File information</i> on the last 300 accessed files.
query -h	<i>Help</i> listing for all these query options.
query -l	<i>Latest</i> information since last reset.
query -n	<i>Names</i> of files accessed and the last accessed time for each listed file.
query -p	<i>Process</i> information (current client information).
query -P <i>port option</i>	<i>Alternate port.</i> Query an xfServer monitor running on a port other than the default (2330). For example, if you enter <code>query -P 2345 -p</code> , you will get process information from the monitor running on port 2345. If you enter <code>query -P 2231 -g</code> , you will get global information from the monitor running on port 2231. If you start <b>rsynd</b> on the default port ( <b>rsynd -m</b> ), you don't have to specify <b>-P port</b> .
query -v	<i>View the rmoncore file.</i> This is available only if <b>-m -t</b> was specified to start the Monitor. (See “ <b>-m -t option</b> ” on page 4-24 for more information.)

## Sample output from the Monitor Utility

query or query -g

```
Server is up for 0 day(s) 0 hour(s) 3 minute(s) and 9 second(s)
Total current clients      : 1
Total current open files  : 1
Total clients' connections : 2
Total files accessed      : 2
Total Data packets        : 2473
Total Data packets size   : 123546
Total packets             : 4968
Total packets size        : 268719
Total error packets       : 2
Total error packets' size : 71
Error packets percentage  : 0.0%
Error pkts size percentage : 0.0%
```

Output	Explanation
Total current clients	The total number of clients currently connected to this instance of xfServer.
Total current open files	The total number of files currently open across all connections.
Total clients' connections	The total number of connections since the Monitor was started.
Total files accessed	The total number of files accessed since the Monitor was started.
Total Data packets	The total number of data packets sent to and from this instance of xfServer. A data packet is defined as a record either sent to xfServer using a STORE or WRITE statement or received from xfServer using a READ statement.
Total Data packets size	The total number of data bytes sent and received. The data bytes refer to the record data only (not including overhead).
Total packets	The total number of communication packets sent and received from xfServer. This includes data and handshaking packets.
Total packet size	The total number of bytes sent and received by xfServer from all packets.
Total error packets	The total number of errors recorded by xfServer. There is no distinction between innocuous and fatal errors here. See <a href="#">“query -e” on page 4-27</a> for an itemized list of errors detected.

Output	Explanation
Total error packets' size	The total number of bytes used in send and receive packets during error conditions.
Error packets percentage	The percentage of error packets against all other packets.
Error pkts size percentage	The percentage of error packet bytes against overall packet bytes.

### query -e

```
err cnt      caused by
-----      -
      1      READS
```

### query -f

```
file name  time  cr co  ac  op  fl  del  stor read  write  pkt#    size
-----
test.ism   16:10  0  0   1   1   0   0   100  1007    0   1107   55298
fred.ism   16:11  1  0   1   1   0   0   1354   12    0   1366   68248
```

Output	Explanation
file name	The name of the file being analyzed. Only the rightmost 17 bytes of filenames accessed will be displayed. Run <b>query -n</b> to identify the full path names.
time	The last time this file was accessed.
cr	The total number of times this file has been created with ISAMC since the Monitor was started.
co	The current number of connections to this file (at the instant <b>query</b> was run).
ac	The total number of times this file has been accessed (with OPEN or ISAMC) since the Monitor was started.
op	The total number of times this file has been opened by an xfServer client since the Monitor was started.
fl	The total number of times a FLUSH has been sent to this file since the Monitor was started.
del	The total number of times a record has been deleted (with DELETE) since the Monitor was started.
stor	The total number of times a record has been stored (with STORE) since the Monitor was started.

## General Utilities

The Monitor Utility for UNIX

Output	Explanation
read	The total number of times a record has been read (with READ or READS) since the Monitor was started.
write	The total number of times a record has been written (with WRITE or WRITES) since the Monitor was started.
pkt#	The total number of packets sent and received while accessing this file since the Monitor was started.
size	The total size of all packets sent and received while accessing this file since the Monitor was started.

### query -l

The output for **query -l** is similar to that of **query** and **query -g** except it may report on totals since the last time Monitor global variables were cleared (with **query -c**).

### query -n

```
date   time           file name
-----
Nov 20 16:43:05      /usr1/data/test.ism
Nov 20 16:11:48      /usr1/data/fred.ism
Nov 20 16:35:02      /usr2/sde64/synergy6/test/rnt/test.ism
```

### query -p

```
User Name           : synuser
Host Name            : aix.synergex.com
Host IP Address      : 134.1.1.140
Log on time          : Nov 20 17:00:05
Clear                : 0
Rename               : 0
Remove               : 0
Current Open file    : 1
Pkts #               : 2622
Size total           : 123498
E_pkt #              : 1
E_Size               : 71
File accessed        : File Name  co del  read  write  store  pkts   size
                       -----
                       test.ism   1    0   1208     0    100    1308   65348
                       data.ism   2    0   1903    123    121   12321  123412
```

Output	Explanation
User Name	The name of the user.
Host Name	The name of the xfServer host machine.
Host IP Address	The IP address of the xfServer host machine.
Log on time	The time at which the user logged on to make the connection.
Clear	The total number of times this connected user has performed a clear operation (XCALL ISCLR) for this connection only.
Rename	The total number of times this connected user has performed a rename operation (XCALL RENAM) for this connection only.
Current open file	The total number of files open by this connected user at this time.
Pkts #	The total number of overall packets sent and received from this connection.
Size total	The total number of bytes for the number of packets for this connection.
E_pkt # and E_Size	The total number of error packets and bytes for this connection.
File accessed	The files currently open by this connection.
co	The total number of times this connection has the file open.
del	The total number of records in this file that have been deleted by this connection.
read	The total number of records in this file that have been read (with READ or READS) by this connection.
write	The total number of records in this file that have been written (with STORE, WRITE, or WRITES) by this connection.
pkts and size	The total number of packets and the size of all packets sent and received while this connection has been accessing this file.

### query -v

	date	time	cp	cf	ct	ft	e_pkt	e_size	d_pkts	d_size	pkts	size
Nov 20	16:12:01		0	0	2	2	2	71	2473	123546	4968	268719
Nov 20	16:35:01		1	1	3	3	0	0	2	2048	10	2872
Nov 20	16:36:01		0	0	3	3	0	0	3	3072	8	3318
Nov 20	16:37:01		0	0	5	4	2	138	0	0	16	915
Nov 20	16:42:01		1	1	6	4	0	0	1	50	8	502
Nov 20	16:43:01		0	0	6	4	0	0	0	0	2	36

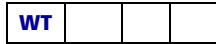
**General Utilities**  
The Monitor Utility for UNIX

```
Nov 20 16:44:07 0 0 7 4 0 0 0 0 8 418
Nov 20 17:01:00 1 1 9 4 4 142 2510 125396 5038 298110
Nov 20 17:17:05 2 2 3 1 0 0 0 0 24 1376
Nov 20 17:21:05 1 1 3 1 0 0 0 0 2 36
```

Output	Explanation
cp	The total number of clients currently connected to this instance of xfServer (same as “Total current clients” from <b>query -g</b> ) since the last interval.
cf	The total number of files that are currently open across all connections (same as “Total current open files” from <b>query -g</b> ) since the last interval.
ct	The total number of connections since the monitor was started. (same as “Total clients’ connections” from <b>query -g</b> ) since the last interval.
ft	The total number of files accessed since the monitor was started (same as “Total files accessed” from <b>query -g</b> ) since the last interval.
e_pkt and e_size	The total number of errors recorded by xfServer and the total number of bytes used in send and receive packets during error conditions (same as “Total error packets” and “Total error packets’ size” from <b>query -g</b> ) since the last interval.
d_pkts and d_size	The total number of data packets sent to and from this instance of xfServer and the total number of data bytes sent and received (same as “Total Data packets” and “Total Data packet size” from <b>query -g</b> ) since the last interval.
pkts and size	The total number of communication packets sent and received from xfServer and the total number of bytes sent and received by xfServer from all packets (same as “Total packets” and “Total packets’ size” from <b>query -g</b> ) since the last interval.



# The ActiveX Diagnostic Utility



The ActiveX Diagnostic utility (**axutl.exe**) enables you to register or test an ActiveX control. Testing a control provides answers to the following questions:

- ▶ Can a given ActiveX control be accessed?
- ▶ If it can be loaded, what are its basic parameters?
- ▶ If it cannot be loaded, why not?

If you cannot load an ActiveX control from your Synergy program, you can use the ActiveX Diagnostic utility to determine whether the control can even be accessed on your system. This eliminates guesswork as to whether the problem lies within your application or in the ActiveX control itself. For example, the ActiveX Diagnostic utility can tell you if a DLL is missing, if the control has not yet been registered, or if the control is not licensed, thus making it inaccessible to your application.

To run the ActiveX Diagnostic utility,

- ▶ Enter **axutl.exe** at the command prompt.

The ActiveX Diagnostic utility dialog box is displayed. (See [figure 4-4](#).)

## Registering an ActiveX control

1. In the ActiveX Diagnostic utility dialog box, click the Add Control button.

The Open dialog box is displayed.

2. Select the control you want to add, and click the Open button.



---

You can also use the Windows regsvr32 utility (**regsvr32.exe**) to register and unregister ActiveX Controls (and DLLs as well). This file is distributed with Windows and is also available on the Microsoft website. See [“Registering an ActiveX control”](#) in the “Synergy ActiveX API” chapter of the *Synergy DBL Language Reference Manual* for more information.

---

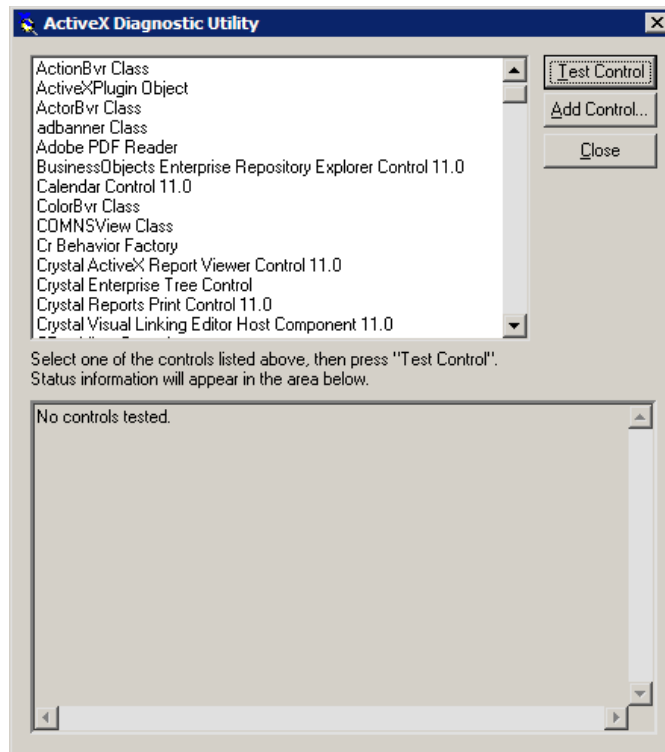


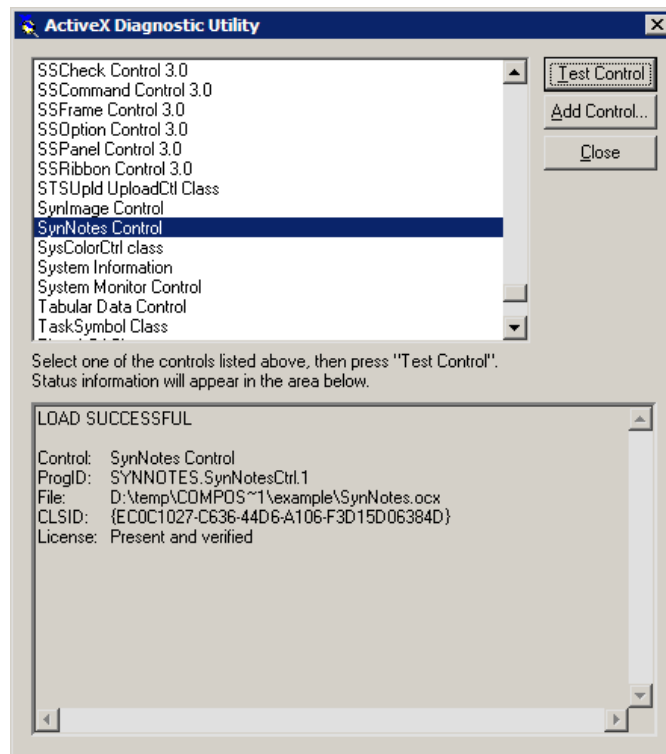
Figure 4-4. The ActiveX Diagnostic Utility dialog box.

## Testing an ActiveX control

1. If the control you want to test is not listed in the ActiveX Diagnostic Utility dialog box, add and register it by following the instructions in [“Registering an ActiveX control”](#) on page 4-31.
2. In the ActiveX Diagnostic Utility dialog box, select the ActiveX control you want to test, and click the Test Control button.

Diagnostic information about the ActiveX control is displayed. (See [figure 4-5](#).)

If the control can be loaded, the message “LOAD SUCCESSFUL” and some additional information is displayed. The ProgID entry is the *control\_name* string that is to be used by `%AX_LOAD` to load that particular control. If this control requires a license, your system is licensed (in other words, the `.lic` license file is present), and a runtime key is available, a LicKey entry is also displayed. The LicKey entry is the *license* string that should be passed to `%AX_LOAD` if you cannot legally distribute the license file to your end users.



*Figure 4-5. Diagnostic results.*

If the control cannot be loaded, or if no **.lic** file is present and licensing is required, the message "LOAD FAILED" is displayed. If the load failed for licensing reasons, you also get the message "License: Runtime key required."

# The Synbackup Utility



Today's 24-7 customer environments require that backups must be able to occur without having to shut down applications. The **synbackup** utility provides a means for all cooperating processes to freeze I/O during a Synergy system backup.



A cooperating process is defined as any Synergy program (**dbr**, **dbs**, **rsynd**, **isutl -r**, **fconvert**, and any programs that use them) that is started after the backup mode feature is enabled. Therefore, make sure you start any other Synergy products (such as xfServer or xfODBC) *after* you configure the backup feature. See [“Configuring the backup mode feature on Windows” on page 4-36](#) and [“Configuring the backup mode feature on UNIX” on page 4-38](#) for configuration instructions.

## How is synbackup used?

The primary use of **synbackup** is to ensure that ISAM files (each made up of two files) are correctly synchronized. The most common problem that plagues an ISAM file is index corruption, where the index file (**.ism**) and data file (**.isl**) are not in sync. Copying an ISAM file while it's performing an update operation is almost certain to make a copy that's out of sync.

**Synbackup** has four backup modes:

- ▶ **Pending.** Backup is about to be performed. The **isutl -r** or **fconvert** utility will not begin operations on a file if this mode is set. All other cooperating processes are unaffected.
- ▶ **On.** Backup can be performed. All I/O by cooperating processes is frozen, and the **isutl -r** or **fconvert** utility will not begin operations on a file if this mode is set.
- ▶ **Off.** Backup cannot be performed. All cooperating processes behave normally.
- ▶ **Not Allowed.** Backup cannot be performed. The **isutl -r** or **fconvert** utility sets this mode when it is in the process of creating a file. When the utility exits (regardless of whether it succeeds or fails), it resets the backup mode to Off. All other cooperating processes are unaffected.

You can use **synbackup** in several different ways:

- ▶ You can use **synbackup** to block ISAM and relative file updates from an application that is completely unaware of the **synbackup** mechanism itself. By freezing updates with **synbackup**, update operations will be blocked until **synbackup** restores updates. You don't need to add anything to your application to accomplish this. Using **synbackup** in this manner ensures your ISAM files are synchronized and backed up correctly.

- ▶ The second use, which requires some application changes, is to use **synbackup** to warn of an impending stoppage of updates. Making some minor code changes allows your application to detect when **synbackup** has issued a Pending state. So, before starting a transaction that may span multiple files, the application can detect the Pending state and choose to wait until after updates have been restored. If the application doesn't wait, the files linked to the transaction might not contain all the changes applied to the transaction. Using **synbackup** in this manner can ensure your transactions are preserved, provided enough time is given between the Pending state and the On state for users to complete a transaction.
- ▶ Finally, in addition to the changes made in the second option above, you can make your application log when a user has started a transaction and when that transaction has been completed. In cooperation with other user processes, this information can be made available to the administrator performing the backup operation. Transactions started after the Pending state has been issued should wait. Transactions started before the Pending state are allowed to complete. Once all active transactions have been completed, **synbackup** can be used to freeze updates. Using **synbackup** in this manner ensures your transactions are preserved.

The following **synbackup** sequence of events is typical in a backup scenario:

1. Put **synbackup** into a Pending state (**synbackup -b**) for a short period of time prior to performing a backup.
2. Freeze Synergy updates by setting the backup mode to On (**synbackup -s**) until the backup is complete.
3. Unfreeze Synergy updates by setting the backup mode to Off (**synbackup -x**).

When the backup mode is Pending or On, DELETES, STORES, and WRITES operations are frozen, and **isutl -r** and **fconvert** operations are not allowed. You can change the way your application handles the freezing of update operations using the `%SYN_SETSTATE` function. (See `%SYN_SETSTATE` and `%SYN_GETSTATE` in the “System-Supplied Subroutines, Functions, and Classes” chapter of the *Synergy DBL Language Reference Manual*.) The default behavior is to suspend application execution.

To reduce the stoppage time that an application may wait for a backup to occur, consider copying the data files to another location first, which is faster than actually backing them up to a tape device. Then, after restoring updates with **synbackup**, the backup to tape can be done on the copies.

To display usage information (or help) for **synbackup**, run it without any options.

Two **synbackup** modes of operation are supported. If the data files are on the same machine as the user applications, **synbackup** is administered from that machine. If the data is located on a data server, **synbackup** is administered from the server and any remote applications must use xfsServer.



**Synbackup** cannot be run from a Synergy/DE Client installation. If it is, a “Backup feature not allowed across network drive” error will be generated. The backup mode feature also is not supported on systems running multiple versions of Synergy.

---

## Synbackup on Windows

You must be a member of the Administrators or Backup operators group to run **synbackup**. **Synbackup** only works with data on a file server using xfServer or with local data accessed via clients using Terminal Services.

**Synbackup** has the following syntax:

```
synbackup [-c] [-b|-s|-x [-w seconds]] [-d] [-q]
```

**-c**

Create the file **DBLDIR:synbackup.cfg**. The backup mode is set to Off initially. (Only a member of the Administrators group can use this option.)

**-b**

Set backup mode to Pending, which means that **synbackup** will next be run with the **-s** option. Pending mode gives applications an opportunity to defer update operations until the pending backup is complete.

**-s**

Set backup mode to On, which freezes all requests for update operations.

**-w**

Specify the maximum number of seconds to wait for an active **isutl -r** or **fconvert** to complete before setting the backup mode. Specifying a value of -1 tells **synbackup** to wait indefinitely until the backup can be performed. The **-w** option can only be specified in conjunction with **-b** or **-s**.

**-x**

Set backup mode to Off, which unfreezes any requests for update operations.

**-d**

Delete the **DBLDIR:synbackup.cfg** file (in other words, disable the backup mode feature). (Only a member of the Administrators group can use this option.)

**-q**

Display the current backup mode.

A file mapping of a physical file (**DBLDIR:synbackup.cfg**) is created that contains the backup mode (Pending, On, Off, or Not Allowed). Only cooperating processes that are local to that file are affected. The backup mode feature is disabled when **DBLDIR:synbackup.cfg** does not exist.

## Configuring the backup mode feature on Windows

- ▶ Initialize the shared memory by running the **synbackup** utility with the **-c** option.

## Synbackup on UNIX

Only a system administrator whose effective user ID is root can use **synbackup** to create or change the backup mode. Otherwise, an error is generated. Any user is allowed to display the current backup mode using the **-q** option.

**Synbackup** has the following syntax:

```
synbackup [-c] [-b|-s|-x [-w seconds]] [-d] [-q]
```

**-c**

Create the shared memory segment for all cooperating processes. The backup mode is set to Off initially.

**-b**

Set backup mode to Pending, which means that **synbackup** will next be run with the **-s** option. Pending mode gives applications an opportunity to defer update operations until the pending backup is complete.

**-s**

Set backup mode to On, which freezes all requests for update operations.

**-w**

Specify the maximum number of seconds to wait for an active **isutl -r** or **fconvert** to complete before setting the backup mode. Specifying a value of -1 tells **synbackup** to wait indefinitely until the backup can be performed. The **-w** option can only be specified in conjunction with **-b** or **-s**.

**-x**

Set backup mode to Off, which unfreezes any requests for update operations.

**-d**

Disable **synbackup** by removing the **synbackup.cfg** file and releasing shared memory.

**-q**

Display the current backup mode as well as the number of current processes that are using **synbackup**.

The current backup mode (Pending, On, Off, or Not Allowed) is maintained in a shared memory segment on the system. The **synbackup** utility is used to initialize and maintain this shared memory segment, as well as to set 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 SYNBACKUP environment variable. If SYNBACKUP is set, the runtime opens the **synbackup.cfg** file to retrieve the base address. If SYNBACKUP is not set, the backup mode feature is disabled.

## Configuring the backup mode feature on UNIX

1. Uncomment the SYNBACKUP=1 line in the distributed **setsde** script.
2. Run **synbackup -c** to initialize the shared memory segment. This shared memory segment must be reinitialized after every system reboot.
3. Ensure that all processes wishing to cooperate source the **setsde** script so the uncommented SYNBACKUP line is honored. (This includes **rsynd**.)



# The Synergy Prototype Utility



The Synergy Prototype utility (**dblproto**) generates prototype files (**.dbp**), which can be used for two purposes:

- ▶ To strongly prototype Synergy code, which ensures that calls match their routine definitions and enables you to detect more problems at compile time rather than waiting until runtime
- ▶ To make declarations within a namespace available to other Synergy source files via the **IMPORT** statement



Strong prototyping is required for object-oriented code, and it happens automatically within a compilation unit. However, outside a compilation unit, such as when you use multiple **dbl** commands, prototype checking is not automatic, and you will need to generate prototypes.

The Synergy Prototype utility has the following syntax:

```
dblproto [options] sourcefile [...]
```

## Arguments

*options*

(optional) One or more of the following. The equal sign is optional; you can use a space instead.

- expdir=directory** Specify the export directory for the generated prototype file, where *directory* is either a full directory specification or an environment variable that contains a directory location.
- N** Process decimal type arguments as numeric (equivalent to setting system option #28).
- out=file** Specify the name of the prototype file to generate. If **.dbp** is not the extension, it is added.
- platform=type** Specify 32- or 64-bit object file creation, where *type* is **x86** for 32-bit platforms or **x64** for 64-bit platforms. If not specified, defaults to the bitness of the machine.
- qdefine:identifier1=value1[, identifier2=value2,...]**  
Set compile-time defines, where *identifier* is the name of the identifier being defined and *value* is the replacement text.

**-qdefns=namespace[;namespace;...]**

Specify namespaces to be imported automatically. This option overrides the setting for SYNDEFNS. The main purpose of this option when prototyping is that if your Synergy files do not have namespaces, they will be assigned to the default namespace, which is the first specified by **-qdefns**.

**-qimkdir=importdir [,importdir ...]**

Specify directories that the compiler will search for IMPORTed namespaces. This option does not override the SYNIMPDIR setting. Rather, if SYNIMPDIR is also specified, the **-qimkdir** directories are searched first, then the SYNIMPDIR directories.

**-quserdef=file**

Specify a file to include when compiling. This option overrides the setting for SYNUSERDEF.

**-qrelaxed=opt[:opt]**

Set **-qrelaxed** options, where *opt* is **end** or **extf**. The following **-qrelaxed** options are set by default and cannot be changed: **allowdup**, **deprecate**, **interop**, **local**, **param**, **path**. With the exception of **allowdup** (which will cause an error), you can put any of these on the **dblproto** command line, though it would be pointless to do so. Note that **-qrelaxed=end** is the same as the **-qrelaxedend** option; consequently the only real use for **-qrelaxed** is to set **extf**. (See “[Relax strong prototyping and error checking](#)” on page 1-14 for details about the effect these options have.) You can substitute a colon (:) for the equal sign (=).

**-qrelaxedend**

Change the behavior of the END statement to clear .DEFINES at the end of the routine instead of at the end of the file.

**-qrntcompat=value**

Target an earlier version of the Synergy runtime. See “[Targeting an earlier runtime version \(-qrntcompat\)](#)” in the Discussion for valid *value* options. (Windows, UNIX only)

**-qvariant=value**

Define an internal compiler variable that can be retrieved with ^VARIANT.

**-single**

Prototype multiple source files (for example, **-single \*.dbl** or **-single a.dbl b.dbl c.dbl**) one file at a time instead of as a compilation unit. This option is deprecated and should not be used.

**-? or -h**

Display **dblproto** command line options and usage information.

#### *sourcefile*

The name of one or more source files to be prototyped. The default filename extension is **.dbl**, and wildcard characters are valid.

You can alternatively specify a redirected input file in the format *<file*, where *file* contains a list of files to be prototyped. Wildcard characters are not valid with this format. For Windows

and UNIX, **-T** is valid (as described in “[Redirecting compiler commands from a file](#)” on [page 1-23](#)). Within the input file, continuation lines are determined by a trailing minus sign (-) on the previous line.

## Discussion

The Synergy Prototype utility creates a single prototype file that contains prototypes for all subroutines, functions, and classes in the source files being prototyped.

When you change your source code, you should regenerate prototypes. For best results, delete the original **.dbp** file(s) before regenerating.

If **-out** is not specified, the file is named with the name of the first source file on the command line. The extension is **.dbp**. If you use a wildcard to process multiple **.dbf** files in a directory, they are processed in alphabetical order, and the **.dbp** file will be named with the filename of the first file processed. (This means the prototype filename could change next time you run **dblproto** if you add or remove source files.)



The **.dbp** files are binary and cannot be moved from one endian system to another.

---

The export directory is determined by the following precedence order:

- ▶ The location designated by the **-expdir** option, if **-expdir** is specified
- ▶ The location designated by the SYNEXPDIR environment variable, if SYNEXPDIR is defined
- ▶ The current directory

The Synergy Prototype utility supports a number of options that correspond to compiler options (**-qdefine**, **-qvariant**, etc.). This is because in order to generate prototypes, **dblproto** must do some of the same processing that the compiler does. Generally speaking, these options should be set the same way you set them when compiling (though this may not be possible for **-qrelaxed**, since **dblproto** sets a number of **-qrelaxed** options for you).

The Synergy Prototype utility ignores blocks of source code between a **.NOPROTO** directive and either a matching **.PROTO** directive or the end of the source file. It automatically encloses all prototypes between a pair of **.NOLIST**-.**LIST** directives to prevent prototypes from being sent to the listing file.



If errors occur during **dblproto** processing, they are reported but prototypes are still generated. We recommend that you fix these errors to avoid future problems when compiling or linking.

---



Prototyping a large number of files may exceed the memory capacity of 32-bit machines. If you need to output 32-bit **dbp** files, we recommend using the 64-bit version of **dblproto** (on a 64-bit machine) and specifying **-platform=x86** to generate 32-bit files.

---

### Targeting an earlier runtime version (-qrntcompat)

(Windows, UNIX only) The **-qrntcompat=value** option enables the compiler to target an earlier version of the runtime. If you will use this option when compiling the code that uses the **.dbp** file, then you must also use it when generating that **.dbp** file. Note that the prototypes generated with **-qrntcompat** cannot be used with an earlier version of the compiler; they must be compiled with the current compiler. See [“Targeting a specific runtime version” on page 1-20](#) for additional information on this feature. The table below shows valid values for *value* and the Synergy version they corresponds to.

Value	Synergy Version
90501	9.5.1
90503	9.5.3
100101	10.1.1
10010101	10.1.1a
10030100	10.3.1
10030101	10.3.1a
10030102	10.3.1b
10030103	10.3.1c

### Implementing strong prototyping

If you generate prototypes for your source files and tell the compiler where to find them (by setting SYNIMPDIR), prototype checking will take place automatically when you compile. As of 10.3.3, you do not need to IMPORT prototypes to use this feature.

1. If you want to specify a default namespace to use for prototypes that do not have a namespace, set SYNDEFNS to the desired namespace. (Or specify it with **-qdefns** on the **dblproto** command line.) If a default namespace is not specified, synglobal will be used.
2. Set the SYNEXPDIR and SYNIMPDIR environment variables to the same path. The prototype file will be generated to the SYNEXPDIR directory and then the compiler will search in the SYNIMPDIR directory. (You don't *have* to make these two environment variables point to the same directory, but it is more efficient to do so.) Optionally, you can specify the export directory on the command line with **-expdir** and the import directory with **-qimkdir**.
3. Run **dblproto** on your **.dbl** files. For the **-out** value, specify a unique filename. For example,

```
dblproto -out=Fred myfile1.dbl myfile2.dbl
```

Or, use a wildcard character to prototype all the files in the directory:

```
dblproto -out=Fred *.dbl
```

The output file, **Fred.dbp**, will contain prototypes of all the functions, subroutines, structures, and classes in **myfile1.dbl** and **myfile2.dbl** (or, in the second example, all **.dbl** files in the directory). It will be created in the directory specified with SYNEXPDIR.

4. Compile your program. The compiler will look in the SYNIMPDIR directory and perform prototype checking against any source files in the compilation unit for which there is a prototype.

### Handling source files in multiple directories

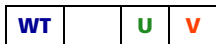
When prototyping sources in multiple directories, you have two options:

- ▶ Place each prototype file in its respective source directory. Do not set SYNEXPDIR; rather, use the **-expdir** option to specify the directory for each file. Then, set SYNIMPDIR to point to all the directories where files were created.
- ▶ Place all the prototype files together in a separate directory. Specify this directory with SYNEXPDIR and set SYNIMPDIR to point to the same directory. This method is simpler in that SYNIMPDIR needs to refer to only one directory, but you must take care that the **.dbp** files have different filenames. If you use this method, we recommend you always explicitly specify a filename with **-out**.

### See also

- ▶ [“Prototyping”](#) in the “Welcome to Synergy DBL” chapter of the *Synergy DBL Language Reference Manual*.
- ▶ [IMPORT](#) in the “Synergy DBL Statements” chapter of the *Synergy DBL Language Reference Manual*.
- ▶ [SYNEXPDIR](#), [SYNIMPDIR](#), and [SYNDEFNS](#) in the “Environment Variables” chapter of *Environment Variables & System Options*.
- ▶ [.NOPROTO-.PROTO](#) in the “Preprocessor and Compiler Directives” chapter of the *Synergy DBL Language Reference Manual*.

# The Variable Usage Utility



By default, the Variable Usage utility identifies unused local variables in each routine. It can also identify the global variables, labels, and include files that are no longer referenced in each routine or the primary source file, as well as those used by one or more local (CALLED, not XCALLED) routines. The variable usage level compiler option designates which items are reported.

To use this utility, compile using the variable usage compiler option and optionally the variable usage level compiler option, as follows:

## WIN, UNIX

---

```
dbl -qvar_review[=file] -qreview_level=n source_file
```

---

## VMS

---

```
dibol /VAR_REVIEW[=file] /REVIEW_LEVEL=n source_file
```

---

## Arguments

*file*

The name of the generated output file. The default filename is the name of the primary source file with a **.unu** extension.

*n*

The sum of one or more of the following bit flags:

- 0**      Unused local variables in each routine (default)
- 1**      Unused global and local variables in each routine
- 2**      Unused labels and local variables in each routine
- 4**      Unused include files and local variables in each routine
- 8**      Unused local variables defined in the primary source file only

## Discussion

Valid values for *n* are 0 through 31. For example, using a value of 5 (1 + 4) reports on unused global and local variables and unused include files in each routine. A value of 11 (1+2+8) reports on unused labels and unused global and local variables defined in the primary source file only.

## Sample output

```
VARIABLE USAGE REPORT FOR c:\dev\test.dbl

ROUTINE ADDRECORD
The following files are included in this routine, but are no longer
referenced:
testdata.def                C:\dev\test.dbl Line: 23
testinfo.def                C:\dev\test.dbl Line: 43

The following local variables are no longer referenced:
IX                          C:\dev\test.dbl Line: 50
CUSTNO                      C:\dev\test.dbl Line: 63

ROUTINE UPDATERECORD
LOCAL ROUTINE LBL
The following local variables are referenced in this local routine:
FF                          C:\dev\test.dbl Line: 348
GG                          C:\dev\test.dbl Line: 349
HH                          C:\dev\test.inc Line: 15
KK                          C:\dev\test.inc Line: 18
No referenced global variables found in this local routine

ROUTINE DELETERECORD
No un-referenced include files found.
No un-referenced local variables found.
```

# The Gennet40 Utility



The **gennet40** utility generates Synergy classes that wrap the classes defined in a .NET assembly.

To generate wrappers for all classes in an assembly, use this syntax:

```
gennet40 -output output_file -log log_file assembly [assembly...]
```

To generate wrappers for specific classes in an assembly, use this syntax:

```
gennet40 -output output_file -log log_file -s xml_file [xml_file...]
```

## Arguments

**-output** *output\_file*

The path and filename of the generated Synergy source file. If an extension is not specified, **.dbl** will be added. You may use a full directory specification or an environment variable that contains a directory location. If *output\_file* contains an environment variable (for example, **MYDIR:myout.dbl**), **.INCLUDEs** in the generated file will also use that environment variable. You can abbreviate **-output** to **-o**.

**-log** *log\_file*

The path and filename of the generated log file. You can abbreviate **-log** to **-l**.

*assembly*

The names of one or more assemblies used for input. If *assembly* is contained in the GAC, do not specify a filename extension; in all other cases, you should include the extension. *Assembly* can be a full directory specification or an environment variable that contains a directory location. See [“Specifying an assembly”](#) in the Discussion for more information.

**-s** *xml\_file*

An XML file that specifies individual classes within an assembly for which you want to generate wrappers. You can also specify a complete assembly in the XML file. See [“Specifying classes within an assembly \(-s\)”](#) in the Discussion for more information.



## Discussion

**Gennet40** generates wrappers for one or more .NET assemblies so you can use them in your Synergy applications.



Never run **gennet40** twice for a single linked unit. If you run it twice and then link the resulting files together, a “Class CRC mismatch in module %s” error (CLSCRC) will occur. If you need to rerun **gennet40**, delete all the existing **gennet40** output files first.

---

Due to .NET Framework security restrictions, **gennet40** should not be run against DLLs over a network share.

Compilation time and memory usage can be significant for the wrapper routines generated by **gennet40**. For large assemblies and assemblies with a large number of dependencies (e.g., .NET Framework assemblies), consider using **-s** to limit processing to only the classes you will use. See “[Specifying classes within an assembly \(-s\)](#)” on page 4-49.

To use **gennet40** follow these steps:

1. Run **gennet40**, specifying all of the assemblies (including dependent assemblies) that you want to use in your Synergy application.

```
gennet -o gen_output.dbl MyAssembly.dll
```

This will produce at least two .dbl files (named **gen\_output.dbl**, **gen\_output1.dbl**, **gen\_output2.dbl**, etc.) and an .inc file (**gen\_output.inc**). The latter is no longer needed, but if your setup already uses it, you can continue to do so. The *filename.dbl* file will contain one or more .INCLUDE statements for the other, numbered .dbl file(s). The exact number of files generated depends on the amount of input. If the input is large, **gennet40** breaks the output into multiple files.

See “[Restrictions and adaptations](#)” on page 4-49 for information on how various aspects of class wrapper generation are handled.

2. Prototype the generated file (the unnumbered one; **gen\_output.dbl** in our example) using **dblproto**. See “[The Synergy Prototype Utility](#)” on page 4-39 for more information.

```
dblproto -out=proto_output.dbp gen_output.dbl
```

3. Compile the Synergy output file that was created by **gennet40** using **-qrelaxed:interop** to create a .dbo file. See “[Restrictions and adaptations](#)” on page 4-49 for information on the requirement to use **-qrelaxed:interop**.

```
dbl -qrelaxed:interop -o gen_output.dbo gen_output.dbl
```



Do not try to build a wrapper routine generated by **gennet40** in debug mode.

---

4. Create a **gen\_output.elb** file from the **.dbo** file you just created:

```
dblink -l gen_output.elb gen_output.dbo
```

We recommend that you use OLBs for **gennet40**-generated classes when they are used in conjunction with ELBs that contain routines or methods that call the **gennet40**-generated classes. When linking an ELB with these OLBs, use the **-r** and **-R** switches.

5. Compile your Synergy program that uses the wrappers. Make sure the compiler has access to prototypes generated in [step 2](#).

```
dbl -o MyProgram.dbo MyProgram.dbl
```

6. Link in the **.elb** file you created in [step 4](#) to your Synergy program and create a **.dbr**:

```
dblink -o MyProgram.dbr MyProgram.dbo gen_ouput.elb
```

7. Run your Synergy program:

```
dbr MyProgram.dbr
```



If an assembly is loaded into **gennet40** as a dependency and later at runtime has one of its types loaded before the original loading assembly, it will not be able to find the DLL outside of the GAC or the directory your program is running in.

---

### Specifying an assembly

If *assembly* contains an environment variable (for example **MYLOC:myassembly.dll**) the generated wrapper will use that environment variable to load the assembly at runtime. Assemblies in the .NET global assembly cache (GAC) should not be specified with an environment variable. If neither an environment variable nor a physical path is specified, **gennet40** searches for the assembly in the following order:

- ▶ The GAC
- ▶ The current directory
- ▶ The dbl\bin directory
- ▶ The windows\system32 directory

The **gennet40** utility does not support search paths. If you specify an environment variable that is set to multiple directory paths, **gennet40** will use only the first one.

If an assembly dependency fails to load from the GAC or the current **gennet40.exe** directory, **gennet40** will try to load from each of the directories from which it has successfully loaded other assemblies. This list of directories is processed in order of appearance.

When running **gennet40** on a DLL, the DLL filename and the assembly name must match. This restriction also applies to dependencies.

## Specifying classes within an assembly (-s)

To specify a subset of the classes in an assembly, use the **-s** option. This option allows you to specify only the classes you want to build wrappers for, instead of creating a large number of unused classes. You do this by creating an XML file that specifies only the desired classes. Do not also specify *assembly* on the command line when you use the **-s** option; if you want to generate wrappers for all the classes in one assembly and only some of the classes in another assembly, specify both in the XML file.

Create an XML file with the following format:

```
<?xml version='1.0'?>
<assemblies>
  <assembly Name="D:\test\MyClassLibrary.dll">
    <class Name = "MyClassLibrary.Class2"/>
    <class Name = "MyClassLibrary.Class4"/>
  </assembly>
  <assembly Name="D:\test\YourClassLibrary.dll">
    </assembly>
</assemblies>
```

The resulting **.dbf** file will have wrappers for only Class2 and Class4 in MyClassLibrary and for all classes in YourClassLibrary.

## Errors

**Gennet40** may generate the following errors. In each of these cases, **gennet40** will terminate with the error status and set the shell ErrorLevel to 1.

Error	Cause
Cannot find specified Assembly <i>name</i>	The specified assemblies cannot be opened and loaded as .NET assemblies.
Error opening output/log file	<i>Output_file</i> or <i>log_file</i> cannot be opened for output.

## Restrictions and adaptations

As far as possible, the generated classes all have the same type information (methods, properties, fields, etc.) in Synergy/DE as they have in .NET, and they will use the DotNetObject methods described in the “[Synergy .NET Assembly API](#)” chapter of the *Synergy DBL Language Reference Manual* to call their .NET equivalents.

Any features of .NET classes that cannot be emulated in Synergy/DE are omitted from the generated classes. Omitted features, types, and methods are reported in the log file. The following are known restrictions and adaptations:

- ▶ The **-qrelaxed:interop** compiler option is required when compiling wrapper classes because it truncates identifiers longer than 30 characters, which is the Synergy/DE limit. **Gennet40** does not truncate identifiers, but it does detect any collisions resulting from the 30-character limit, omits entities involved in collisions, and logs these omissions to the log file. If a collision results in the omission of a class, methods and properties that require that class are also omitted.
- ▶ The classes generated by **gennet40** are derived from `DotNetObject`, and a `ToString()` method with the `NEW` modifier is generated for each generated class. Whichever type your instance variable is declared as determines which `ToString()` will be used. Therefore, a call to `ToString()` will run `DotNetObject`'s `ToString()`.
- ▶ Generics and class names containing invalid Synergy/DE identifier characters are not generated.
- ▶ An “m” is added to the beginning of any identifier that starts with a leading underscore (`_`) to make the name legal. Additionally, identifiers that are reserved words in Synergy classes (such as **this**, **parent**, **and**, **or**, etc.) are prefixed with “m\_”.
- ▶ Interfaces are folded into the generated classes when possible but may present conflicts.
- ▶ Parameters and return values that collide with Synergy types in the System namespace are automatically converted to the Synergy type. If you are using `System.Collections.ArrayList`, modifications made to a returned array are not visible to the corresponding .NET object unless the array is copied back through a method or property invocation. In addition, delegates that have parameters defined as `System.Collections.ArrayList` are marked `INOUT`.
- ▶ Public members of `System.Object` that are not included in the Synergy implementation of `System.Object` are omitted from all generated classes.
- ▶ Any member that expects or returns a pointer type is omitted.
- ▶ All objects contain their full .NET constructors with parameters, including default constructors.
- ▶ Inheritance is flattened so all methods are visible.
- ▶ Reference parameters in the .NET assemblies are converted to `INOUT` arguments.
- ▶ An event is turned into a nested class. A member of that type is placed as an instance of the event, which allows you to use the `AddHandler`, `RemoveHandler`, and `RaiseEvent` methods. The nested class for the event is prefixed with “e\_”.
- ▶ The `Clone`, `Equals`, `CreateInstance`, and `GetObjectData` methods are not generated on any generated classes.

See the “[Synergy .NET Assembly API](#)” chapter in the *Synergy DBL Language Reference Manual* for more information.

# The Dbl2xml Utility



The **dbl2xml** utility processes Synergy DBL source files that include language attributes, parameter modifiers, and comments, and outputs an XML file containing interfaces and methods. This XML file is then used to update the Synergy Method Catalog (SMC), via the Method Definition Utility’s import facility, for use with xfServerPlus and xfNetLink.

Many users find updating the SMC manually by entering data in the Method Definition Utility (MDU) to be not only tedious, but also subject to error because changes to source code oftentimes require a change to the SMC definition as well. By attributing your code and running **dbl2xml**, you can automate the process and achieve greater accuracy. See “[Using Attributes to Define Synergy Methods](#)” in the “Defining Your Synergy Methods” chapter of the *xfNetLink & xfServerPlus User’s Guide* for details on attributing your code.

The **dbl2xml** utility has the following syntax:

`dbl2xml [options] sourcefile [...]`

## Arguments

### *options*

(optional) One or more of the following. The equal sign is optional; you can use a space instead.

- |  |   |
|--|---|
| <b>-out=</b> <i>output_file</i>                                      | Specify the path and filename for the XML output file. You can use a logical or the full path. The extension <b>.xml</b> will be appended to the filename if no extension is specified. By default, the file is named with the first interface encountered during processing and placed in the current directory. |
| <b>-qdefine:</b> <i>identifier1=value1[, identifier2=value2,...]</i> | Set compile-time defines, where <i>identifier</i> is the name of the identifier being defined and <i>value</i> is the replacement text.   |
| <b>-qrelaxedend</b>  | Change the behavior of the END statement to clear .DEFINES at the end of the routine instead of at the end of the file.   |
| <b>-qvariant=</b> <i>value</i>                                       | Define an internal compiler variable that can be retrieved with ^VARIANT.   |
| <b>-single</b>   | Process multiple source files (for example, <b>-single *.dbl</b> or <b>-single a.db1 b.db1 c.db1</b> ) individually instead of as a compilation unit.   |
| <b>-?</b> or <b>-h</b>   | Display <b>dbl2xml</b> command line options and usage information.  |

#### *sourcefile*

The name of one or more source files to be processed. The default filename extension is **.dbl**, and wildcard characters are valid.

You can alternatively specify a redirected input file in the format `<file`, where *file* contains a list of files to process. Wildcard characters are not valid with this format. For Windows and UNIX, **-T** is valid (as described in [“Redirecting compiler commands from a file” on page 1-23](#)). Within the input file, continuation lines are determined by a trailing minus sign (-) on the previous line. If the input file contains multiple lines without the use of continuation characters, **dbl2xml** treats each line separately, and multiple XML files will be produced.

## Discussion

The **dbl2xml** utility creates a single XML file from one or more source files. Before running this utility you must attribute your code as described in [“Using Attributes to Define Synergy Methods”](#) in the “Defining Your Synergy Methods” chapter of the *xfNetLink & xfServerPlus User’s Guide*.

If the routines included in a single interface are located in more than one source file, you should process all those source files at the same time. This will make it easier to update the SMC, as you can then import the entire interface at once, replacing the existing interface (if there is one). See [“Importing and Exporting Methods”](#) in the “Defining Your Synergy Methods” chapter of the *xfNetLink & xfServerPlus User’s Guide* for instructions.

The **-single** option causes files to be processed one at a time rather than together in a compilation unit. We suggest that you use **-single** if you have a lot of **.dbl** files with no interdependencies and you do not want to consume a large amount of memory or processing time.

In addition to errors generated by **dbl2xml**, you may see compiler errors because **dbl2xml** runs the compiler before creating the XML file. We recommend that you compile and fix any errors before running **dbl2xml**.



When you change your source code, you should run **dbl2xml** to regenerate the XML file and then re-import it into the SMC. You may want to add the commands to run **dbl2xml** and to import the XML file to your build script.

---

## Examples

The first example processes two source files and creates an XML file named **fred.xml**. The second example uses the **-single** option and a wildcard character to process all **.dbl** files in the current directory and creates an XML file named **fred.xml**:

```
dbl2xml -out=c:\work\fred myfile1.dbl myfile2.dbl
```

```
dbl2xml -out=c:\work\fred -single *.dbl
```

## See also

[“The Method Definition Utility”](#) in the “Defining Your Synergy Methods” chapter of the *xfNetLink & xfServerPlus User’s Guide*.

# 5

## Error Messages

### **About Synergy DBL Errors 5-2**

Discusses the effect of trappable and fatal errors and explains error literals and error message variables.

### **Runtime Errors 5-4**

Lists error literals, numbers, and messages for the trappable, informational, fatal, success, debugging log, and window runtime errors, along with a brief explanation of each message.

### **Compiler Errors 5-52**

Lists error literals and messages for the nonfatal, informational, fatal, and warning compiler errors.

### **Linker Errors 5-117**

Lists error literals and messages for the fatal, informational, and warning linker errors.

### **Librarian Errors 5-125**

Lists error literals and messages for the fatal and warning librarian errors.

### **Synergy DBMS Errors 5-128**

Lists the error numbers and messages for Synergy DBMS.

### **List of Runtime Error Numbers 5-132**

Lists the runtime error numbers and literals in consecutive numerical order, as they are listed in the file `syntxt.ism`, to give you an alternative way to look up error numbers.

## About Synergy DBL Errors

Synergy DBL uses the following types of error messages:

- ▶ Trappable or nonfatal warning errors
- ▶ Informational messages that describe the error in more detail
- ▶ Fatal errors that cause the program to abort
- ▶ A success message that indicates the program completed normally
- ▶ Window manager errors

Some of the error messages contain a variable, such as “%s” or “%d”. These variables are replaced with an actual value, character, or string at runtime. The variables and their corresponding replacement values are as follows:

This variable	Is replaced with this value
%s	alpha string
%d	signed decimal value
%u	unsigned decimal value
%ld	signed longword value
%c	single alpha character

## Trapping runtime errors

Trappable runtime errors are those from which your program can recover. You can trap these errors using the **ONERROR** statement, end-of-file labels in some of the I/O statements, and I/O error lists that can be associated with any I/O statement.

When establishing error traps, your program indicates errors that are to be trapped, as well as the label(s) to which execution control is to be transferred. When your program traps an error, program control transfers to the appropriate error handling code as if a **GOTO** statement had occurred. For more details, see **ONERROR** in the “Synergy DBL Statements” chapter of the *Synergy DBL Language Reference Manual* and “**Error Trapping**” in the “Error Handling” chapter of that same manual.



Every trappable error has an error literal or a mnemonic that begins with `$ERR_`. For example, the error literal for `ARGSIZ` is `$ERR_ARGSIZ`, and the error literal for `ILLCHN` is `$ERR_ILLCHN`. You can use these error literals with `ONERROR` statements, in I/O error lists, and wherever you can use a literal. For example:

```
onerror ($ERR_IOFAIL, $ERR_DIGIT) proc_err1, ($ERR_EOF) proc_err2  
  
read(1, data, rec_id) [$ERR_IOFAIL=proc_err1]
```

## Fatal errors

Fatal, nontrappable errors result in immediate, unconditional termination of program execution. When a program is abnormally terminated because of a nontrappable error (or a trappable error for which no error trapping was established), an error message and a traceback of the program are generated. The traceback is a record of how your program reached the line of code that encountered the error.

The traceback appears beginning at the line at which the error was detected, and proceeds backward until all `CALLs` and `XCALLs` have been displayed. The following is an example of traceback:

```
%DBR-F-CHNUSE, Channel is in use  
%DBR-I-ATLINE, at line 584 in routine M_PROCESS_P  
    Called from line 814  
    Called from line 462  
%DBR-I-ATLINE, at line 335 in routine M_PROCESS  
%DBR-I-ATLINE, at line 156 in routine TXTUTL
```

If you're running your program in the debugger and a fatal error is encountered, the debugger generates the fatal error message with its traceback and break at the line that caused the fatal error. This feature enables you to remain in the debugger for post-error debugging.

## Using error literals instead of numbers

In addition to a literal, each error message also has an associated number. *We strongly recommend that you use error literals instead of error numbers to make your code easier to read and maintain.*

## Customizing error messages

Error numbers and messages are defined in the **syntxt.ism** message library (as shown in [“List of Runtime Error Numbers” on page 5-132.](#)) You can use the Synergy UI Toolkit Control Panel to translate or otherwise customize these messages if you'd like.

See [“The Synergy UI Toolkit Control Panel” on page 4-3](#) for information about using the Synergy UI Toolkit Control Panel to extract and modify error messages.

# Runtime Errors

## Runtime error messages

The following error messages are those that can be trapped and from which your program can recover. For a list of errors sorted by error number, see [“List of Runtime Error Numbers” on page 5-132](#).

\$ERR_ADDR Siz	160	Invalid address size	The subroutine address passed to XSUBR is of incorrect size.
\$ERR_ALCOMPAT	626	ArrayList compatibility issue. See the 9.1.5 release notes	You’ve attempted to run a program that uses the ArrayList class without recompiling first. You will only potentially see this error after upgrading to 9.1.5 or higher, and once you have recompiled, you will not see it again. However, you must still either change all references to ArrayList to Synergex.SynergyDE.Collections.ArrayList or convert all ArrayList index references to be 0-based. If you don’t do one of these two steps, you will probably encounter future problems. More detailed information is provided in the 9.1.5 release notes.
\$ERR_ALPHARG	519	Alpha argument required	The subroutine requires an alpha variable to pass back a text value.
\$ERR_AORDXP	82	Alpha or decimal variable expected	Either an alpha or decimal variable is required by the operation.
\$ERR_AORIARG	522	Integer or alpha argument required	A call being made using %DLL_CALL has passed an argument that is neither alpha nor integer data type.
\$ERR_ARGDIG	151	Numeric digit(s) expected in argument	You must pass an argument that contains only numeric characters (0 – 9).
\$ERR_ARGDIGPT	144	Numeric digit(s) and at most one decimal point expected	You must specify an argument that contains only numeric characters (0 – 9) and a maximum of one decimal point.
\$ERR_ARGMIS	87	Argument missing	You did not pass an argument that was expected by the external subroutine.

\$ERR_ARGORD	77	Arguments out of order for PAK or UNPAK The fields passed in the PAK or UNPAK subroutines are not in ascending order.
\$ERR_ARGREC	78	PAK/UNPAK fields not in record A field not contained in the specified record for either the PAK or UNPAK subroutine was found.
\$ERR_ARGSIZ	31	Argument specified with wrong size You passed an argument whose length was not within the prescribed limits to a system-supplied external subroutine. Often, this error means the argument was too short.
\$ERR_ARRAYBND	619	Index is outside the bounds of the array The array index does not lie within the specified array.
\$ERR_AXERR	421	Error HRESULT ( <i>number</i> ) while processing an ActiveX control  An error state occurred in one of the ActiveX API routines but was not specifically recognized. For example, a uniquely ActiveX-related error status on %AX_CALL, an unexpected error state on %AX_CREATE, or a missing COM interface on %AX_LOAD could cause this error. In almost all cases, additional qualifying information regarding the exact error state is displayed in the debug log if AXDEBUG is set to yes. <i>Number</i> is the returned HRESULT number, if any, from the ActiveX control
\$ERR_AXNOFIND	425	ActiveX parameter not found The second argument to %AX_GET, %AX_GETINT, %AX_SET, %AX_BIND, or %AX_CALL is not defined.
\$ERR_AXNOLOAD	422	Could not load ActiveX control %AX_LOAD could not load the referenced ActiveX control.
\$ERR_AXNOSUB	423	Could not find subroutine or function %AX_BIND could not find the Synergy DBL routine that was specified as an argument.
\$ERR_AXUNSUP	424	Unsupported feature The ActiveX control being loaded uses a feature that the Synergy ActiveX API doesn't support. Additional information is displayed in the debug log if AXDEBUG is set to yes.

## Error Messages

### Runtime Errors

\$ERR_BACKPEND	13	Backup mode is On An attempt was made to DELETE, STORE, or WRITE to a Synergy ISAM file when backup mode was On.
\$ERR_BACKUPMODE	41	Backup mode error The shared memory segment on a UNIX system cannot be attached to. Use the system error code reported (or %SYSERR if the error is trapped) to debug your shared memory problem.
\$ERR_BADADDR	531	Bad address detected: %s Returned by the Synergy DBL debugger when asked to examine an invalid address.
\$ERR_BADDATATYP	335	SQL: Invalid data type for this operation See <a href="#">“Trapping runtime error messages in SQL Connection programs”</a> in the “Error Logging and Messages” chapter of the <i>SQL Connection Reference Manual</i> .
\$ERR_BADDBGPORT	608	Invalid debug port number: %s. Must be an integer within the range 1024 to 65535, inclusive The port number specified in the <b>dbf -rd</b> command is not in the correct range. It must be between 1024 and 65535, inclusive.
\$ERR_BADDBGTMOT	609	Invalid remote debug timeout value: %s The timeout value specified in the <b>dbf -rd</b> command is either negative, 0, or alpha. It must be a positive numeric value.
\$ERR_BADELB	532	Bad ELB detected: %s An ELB was specified that was invalid or corrupted. This error also could be caused by duplicate global commons, global literals, global data sections, or static records when the ELB was loaded, if the sizes of the duplicate records are different.
\$ERR_BADFONTID	542	Invalid font ID specified: %d The specified font handle was passed to one of the %U_WNDFONT subfunctions, and the handle does not correspond to any known font.
\$ERR_BADFONTNAM	539	Invalid font name specified: %s An invalid font name was passed to one of the %U_WNDFONT subfunctions. A font name must be an identifier (case insensitive, containing the characters a–z, 0–9, _, or \$, and beginning with a–z) and is limited to 60 characters.

\$ERR_BADFORMAT	525	Bad format string Bad argument translation string passed to %DLL_SUBR or %DLL_CALL.
\$ERR_BADHANDLE	526	Bad DLL handle Invalid DLL handle passed to %DLL_SUBR, %DLL_CALL, or %DLL_CLOSE.
\$ERR_BADHOST	325	Unknown host “%s” in server spec The server host specified in a client program was unknown or unreachable.
\$ERR_BADKEY	52	Illegal key specified One of the following conditions has occurred: <ul style="list-style-type: none"><li>▶ The specified key name does not match a key.</li><li>▶ An implied key specification does not match a key (wholly or partially).</li><li>▶ The specified key index is not in the range defined for the ISAM file.</li><li>▶ On Windows and UNIX, an explicit key of reference specified on a READ or a FIND statement exceeds the number of keys in the file.</li></ul>
\$ERR_BADUSER	326	Bad username, login rejected on %s An invalid username or password caused the Synergy/DE xfServer to reject a client login.
\$ERR_BADWNDID	549	Window %d bad or no longer open The specified window ID, which was passed to one of the windowing API routines (W_), is invalid.
\$ERR_BDIGXP	145	Binary digits expected in argument (%s) You must specify an argument that only contains binary digits (0 or 1).
\$ERR_BIGALPHA	14	Alpha temporary result exceeds 65535 The results of an alpha string concatenation are greater than 65535 bytes on a 32-bit system.

\$ERR_BIGNUM	15	Arithmetic operand exceeds maximum size	<p>Either an operand or the result of an arithmetic operation exceeds the allowable size for its data type. Possible causes are as follows:</p> <ul style="list-style-type: none"><li>▶ During evaluation of an arithmetic expression, the number of significant digits in the final or some intermediate result exceeded 28 for a decimal variable or the whole number part of an implied-decimal variable.</li><li>▶ The INCR statement caused the value of a variable to overflow its field width. (For example, the highest number a <b>d2</b> field can hold is 99.)</li><li>▶ A decimal value that exceeded the documented limit was passed to a system-supplied external subroutine.</li><li>▶ A decimal value exceeds the minimum 64-bit value.</li><li>▶ You specified a decimal value greater than 65,535 as the control value in a SLEEP statement or as the number of seconds to wait in the WAIT external subroutine.</li></ul>
\$ERR_BITOPP	599	File built with opposite bit size: %s	<p>An attempt was made to load an ELB built with the wrong architecture size (32- or 64-bit) for this runtime.</p>
\$ERR_BLKSIZE	115	Invalid value specified for BLKSIZ	<p>The BLKSIZ value specified in an OPEN statement is outside the permitted range of values.</p>
\$ERR_CATCH	900	(Internal use only)	<p>\$ERR_CATCH is passed to the ONERROR statement to enable the catching of errors in called routines that do not have an ONERROR statement.</p>
\$ERR_CHNEXC	33	Too many files open	<p>You've attempted to open more channels than this system supports. For each file you need to open, you must close a previously opened file to avoid this error. You can also reconfigure your operating system to support more open channels.</p>
\$ERR_CHNUSE	16	Channel is in use	<p>An OPEN statement specified the number of a channel that's currently in use.</p>

\$ERR_CLNTERR	319	Client server error, host %s An error occurred during communications with the Synergy/DE xfServer.
\$ERR_CLSMTCH	603	Class mismatch between routines The class being referenced doesn't match the same class referenced in a prior routine. This occurs when a class is changed (class members are added, removed, or changed) and the prototype hasn't been rebuilt or the modules referencing the class hasn't been recompiled.
\$ERR_COMPATISAM	632	Incompatible ISAM file - unsupported by this Synergy version The ISAM file you're attempting to use contains a new feature that that changed the file's structure, which makes it incompatible with older versions of Synergy.
\$ERR_CURSERR	334	ID must be a non select cursor ID not SELECT cursor Invalid cursor ID See <a href="#">"Trapping runtime error messages in SQL Connection programs"</a> in the "Error Logging and Messages" chapter of the <i>SQL Connection Reference Manual</i> .
\$ERR_DATACRYPT	433	Error encrypting data field: %s Error decrypting data field: %s The specified error occurred during encryption or decryption processing via OPENSSL.
\$ERR_DBGCLOSED	613	Remote debug client closed the connection; continuing without debug The debug client closed the connection. All breakpoints and watchpoints have been cancelled, and program execution continues as if debug were not enabled.
\$ERR_DBGNOCONN	611	No debug client connection was established A debug client did not connect to the port within the specified timeout period. No debug client connection was established, and program execution continues as if debug were not enabled.

## Error Messages

### Runtime Errors

\$ERR_DBGNOSOCK	610	Unable to attach to remote debug port The port specified for debugging was already in use when xfServerPlus attempted to launch the runtime or when the runtime attempted to listen on that port. The xfServerPlus session or program execution continues as if debug were not enabled.
\$ERR_DBGSOCKER	612	Remote debug socket error; continuing without debug xfServerPlus was able to launch the runtime and the runtime was able to listen on the specified debug port, but some other error occurred when attempting to accept a connection. More detailed information on the error that occurred is appended to the log entry. The xfServerPlus session or program execution continues as if debug were not enabled.
\$ERR_DEADLOCK	535	Operation would cause deadlock A READ from a file that is locked by a process waiting for a lock that the reading process holds would cause a deadlock condition.
\$ERR_DECXPT	153	Decimal expected An argument was passed, but it was not type <b>d</b> as required.
\$ERR_DELREC	318	Deleted record The record you are attempting to access with an RFA has been deleted or moved. (For more information, see “ <a href="#">Static RFAs</a> ” in the “Synergy DBL Statements” chapter of the <i>Synergy DBL Language Reference Manual</i> .)
\$ERR_DEVNOTRDY	107	Device not ready The device accessed by an I/O statement was off-line or otherwise not ready (the modem has hung up, the device is a network device that has been disconnected, and so forth), or one of the following OpenVMS system errors was received: SS\$_DEVOFFLINE, SS\$_HANGUP, SS\$_DISCONNECT, or SS\$_DEVINACT.
\$ERR_DEVUSE	37	Device in use An OPEN statement attempted to open a nonsharable device that was in use.
\$ERR_DIFDIMS	620	Arrays must have the same number of dimensions The System.Array method Copy requires that the number of dimensions match exactly.



\$ERR_DIGIT	20	Bad digit encountered  The alpha value being converted to a numeric value contained a character that is not a digit 0 through 9, a decimal point, a space, or a sign character (+ or -).
\$ERR_DIVIDE	30	Attempt to divide by zero  An arithmetic operation attempted to divide by zero.
\$ERR_DLLCLSERR	529	DLL could not be closed  An error occurred while %DLL_CLOSE was closing a DLL.
\$ERR_DLLOPNERR	528	DLL could not be opened: %s  An error occurred while %DLL_OPEN was opening the specified DLL. Possible causes are as follows: <ul style="list-style-type: none"><li>▶ The .dll file could not be found.</li><li>▶ Your application does not have read access to the .dll file.</li><li>▶ Not enough virtual memory is available to load the .dll file.</li><li>▶ You have exhausted the maximum number of handles on your system.</li><li>▶ The DllMain function of the DLL caused an error.</li><li>▶ The DLL attempted to load another DLL, which failed for any one of the above reasons.</li></ul>
\$ERR_DLLOPNMOD	548	Associated DLL not in path or not found  When a DLL was being opened, another DLL required by the first could not be found.
\$ERR_DUPFONTNAM	540	Duplicate font name specified: %s  Internally, an attempt was made to create a font with a name already assigned to some other font.
\$ERR_EOF	1	End of file  You've attempted to access information beyond the logical or physical end of a file. The physical end of a file was detected by a READS or READ, or the end-of-file indicator (which varies according to operating system) was entered from a character-oriented device during an ACCEPT or READS.

## Error Messages

### Runtime Errors

\$ERR_EXCACT	120	Too many activation characters	Either the ACCHR (or ACESC) subroutine attempted to define additional activation characters but exceeded the limit of 10, or you specified an invalid activation character to the DACCHR (or DAESC) subroutine.
\$ERR_EXCEPT	616	Exception of type '%s'	<p>An exception was created that doesn't map to a \$ERR_ error. For example, if your code contains</p> <pre>exception_handle = new SynException()</pre> <p>the generic \$ERR_EXCEPT error is used because there's no associated \$ERR_ text. The text of the error message (which in this case would be "Exception of type 'SYNERGEX.SYNERGYDE.SYNEXCEPTION'") is specified in the exception_handle.Message property.</p>
\$ERR_EXECF1	590	Cannot execute: %s	An attempt to execute a program using the RUNJB subroutine was unsuccessful. The file may not be present or this may be an incorrect command.
\$ERR_EXQUOTA	106	Exceeded quota	The runtime failed because it exceeded some process limit imposed on it by the system.
\$ERR_FILFUL	25	Output file is full	All space allocated for a file has been filled, and the file cannot be extended. This includes running out of physical disk space or reaching some system-level artificial file size limit (like ULIMIT on UNIX).
\$ERR_FILOPT	21	Invalid operation for file type	You've issued an I/O statement that was not allowed by the mode in which the file was opened. For example, you would get this error if you attempted to write to a file that was opened in input mode.
\$ERR_FILORG	103	Invalid file organization	You've opened a file whose organization is different than that specified in the OPEN mode.
\$ERR_FILSPC	17	Bad filename	A file specification contains a syntactical error.

\$ERR_FINUSE	38	File in use by another user
	One of the following has occurred:	
	<ul style="list-style-type: none"> <li>▶ The file specified in an OPEN statement is in use by another user and is not available as a shared file.</li> <li>▶ The file specified in a call to RENAM or DELET is currently open by another user.</li> <li>▶ The file specified in a call to COPY is open in update, output, or append mode by another user.</li> </ul>	
	Verify that the file is closed before attempting to rename or delete it, or that it is either closed or open in input mode before attempting to copy it.	
\$ERR_FNF	18	File not found
	You've attempted to locate a file that doesn't match a specified filename. This error usually occurs on an OPEN statement. The FNF error can also occur on an LPQUE statement, a STOP statement, or the processing of some system-supplied external subroutines, such as DELET and RENAM. It can also occur on a %SYN_SYSERRTXT call if %SYN_SYSERRTXT does not immediately follow an ONERROR statement.	
\$ERR_FNOTFOUND	523	Function not found
	The DLL function that you've called with %DLL_CALL routine doesn't exist. Check the documentation for the DLL to find out why the function isn't there.	
\$ERR_FONTINUSE	541	Font %d in use, cannot delete
	Internally, upon shutdown, an attempt was made to delete a font that is currently in use.	
\$ERR_HDIGXP	146	Hexadecimal digits expected in argument (%s)
	You must specify an argument that only contains hexadecimal digits (0 – 9 and A – F).	
\$ERR_HKNOOPS	628	No I/O hook operations specified; at least one is required
	No mask values (neither pre nor post) were specified on the constructors. One or more IOEventMask values is required when using an I/O hooks constructor that takes pre- and post-mask values.	
\$ERR_HKNOTACT	629	I/O hooks no longer active on channel
	The IOReset method was called on an I/O hooks object that has already been released. (For example, closing the handle causes this to occur.)	

## Error Messages

### Runtime Errors

\$ERR_HKNOTIMP	630	Referenced I/O hook operation(s) not implemented: %s An operation specified via pre-mask or post-mask resolved to an IOHooks virtual method whose override was not implemented.
\$ERR_HKACTIVE	631	Attempted I/O on hooked channel from within hook routine The I/O handler hook routine performed I/O on or attempted to close the channel on which an event was raised.
\$ERR_HNDCORUPT	625	Handle has been modified; possible subscripting violation An invalid object handle was detected, and its contents resemble the result of a CLEAR exceeding the boundary of a single data variable.
\$ERR_HSIZERR	161	Map outside bounds of field or handle A ^M reference caused a data reference outside of the declared bounds of the handle or data field.
\$ERR_IDPARMREQ	605	Implied-decimal parameter required An implied argument was passed to a routine that declared the parameter type as <b>n</b> instead of <b>n.</b> .
\$ERR_IDXP	158	Implied data type required A system function was passed an argument with an incorrect data type.
\$ERR_ILLCHN	10	Illegal channel number specified You've specified a channel number that is less than 1 or greater than 1024. Channel numbers appear as part of I/O statements and as arguments to some system-supplied external subroutines.
\$ERR_INCPTCLS	600	Incompatible classes An object handle was used that does not match (or is not an ancestor of) a required class for a particular operation, or if you're using the FOREACH statement, the elements of the specified collection cannot be cast to the type of the loop variable. This error is always accompanied by the "Class <%s> is not an ancestor of <%s>" informational error.
\$ERR_INTARG	521	Integer argument required A system function was passed an argument with an incorrect data type.
\$ERR_INTISM	302	Internal Isam error - File may be corrupt An internal ISAM error occurred whose cause may be due to a file that is corrupted. This error is always accompanied by an informational error that provides additional information.

\$ERR_INTLCK	303	Unexpected system locking error An attempt to acquire a file or record lock failed due to an unexpected system error.
\$ERR_INTRPT	98	Interrupt character detected Program execution was terminated because the user entered the interrupt character. You can disable the aborting action of the interrupt character by using the FLAGS subroutine to set runtime option flag 8 or by specifying the system option #10.
\$ERR_INVACT	309	Invalid action for XCALL FATAL You specified an invalid action in the FATAL subroutine. Valid values are 0 – 3.
\$ERR_INVALRFA	317	Invalid record's file address One of the following occurred: <ul style="list-style-type: none"><li>▶ You specified an invalid RFA on an I/O operation.</li><li>▶ The size of the alpha argument to the GETRFA or RFA qualifier was something other than 6 or 10.</li><li>▶ GETRFA:<i>global_rfa</i> was used on the FIND statement, on the READ, READS, and WRITE statements for file types other than ISAM and relative, or on the WRITES statement for file types other than relative.</li><li>▶ RFA:<i>global_rfa</i> was used on the FIND, READ, and WRITE statements for file types other than ISAM and relative.</li><li>▶ The GETRFA and RFA qualifiers on the same READ statement had variables of different sizes. If a READ statement has both a GETRFA and an RFA qualifier, the variables for must both be either 6 bytes (RFA size) or 10 bytes (global RFA size).</li></ul>
\$ERR_INVARG	420	Invalid argument A parameter is invalid for a given usage. If this error occurs in an ActiveX API routine, additional information regarding the exact error state is displayed in the debug log if AXDEBUG is set to yes.
\$ERR_INVATIME	633	Invalid timestamp key exceeds current time The timestamp autokey in an ISAM file is later than the current time. This may occur, for example, if you've set your server time back or moved the file to another system where the current time is in the past relative to the original machine.

## Error Messages

### Runtime Errors

\$ERR_INVCAST	617	Invalid cast operation	The type used to cast a variable is not the class or an ancestor class of the object stored in the variable, or it does not have an explicit conversion operator.
\$ERR_INVCLLSEQ	546	Invalid calling sequence	You specified a Windows printing API function or operation that requires previously set values. (For more information about valid calling sequences, see <a href="#">“Recommended calling sequence”</a> in the “Synergy Windows Printing API” chapter of the <i>Synergy DBL Language Reference Manual</i> .)
\$ERR_INVCLSHND	574	Invalid class handle	An invalid class handle was used while referencing a class object. (If the runtime recognizes that the handle is corrupt, it generates an \$ERR_HNDCORUPT error.)
\$ERR_INVDATE	527	Invalid date	A system date routine was passed an illegal or badly formed date.
\$ERR_INVDIM	223	Invalid number of dimensions	You’ve passed a dimensioned array as an XCALL argument, and the associated argument within the subroutine was either not dimensioned or did not have the same number of dimensions.
\$ERR_INVDSR	568	Invalid descriptor	An invalid descriptor was passed in a Synergy argument. If you get this error, please contact Synergy/DE Developer Support for assistance. (See <a href="#">“Product support information” on page x</a> for details about contacting Support.)
\$ERR_INVEXFTYP	417	Invalid external function data type	An incorrect data type was declared in or passed to an external function, or a subroutine whose first parameter is an alpha was called as a function.
\$ERR_INVFORENT	157	Invalid entry to FOR loop	A FOR loop was entered without being initialized (in other words, using a GOTO statement).
\$ERR_INVHDL	159	Invalid memory handle	An invalid memory handle was used in a memory allocation or ^M statement.

\$ERR_INVNAMHND	573	Invalid namespace handle A namespace ID that was not a valid memory handle was passed in one of the %NSPC_ routines. Correct the namespace ID.
\$ERR_INVNETHND	571	Invalid network handle A network connection ID that was not a valid memory handle was passed to xfNetLink Synergy. Correct the network connection ID.
\$ERR_INVOPER	627	Invalid operation: %s An invalid operation has occurred. The following are some examples of invalid operations: <ul style="list-style-type: none"><li>▶ A collection was modified while a FOREACH statement was being executed.</li><li>▶ An invalid operation occurred during a Select.</li><li>▶ You specified a Where clause in the Select that contains a join, with a reference to one of the inner tables. A Where expression can only reference the driving (or outermost) table. To apply a filter condition to an inner table, move it to one of the On expressions.</li></ul>
\$ERR_INVKEY	341	Invalid partial key A key was read or written that didn't include all or part of the key. Integer and decimal keys cannot be partially read or written, but they can be entirely suppressed. Autokeys must be read or written in their entirety, and they cannot be suppressed.  When one of the new autokeys gets read or written, their read or write buffer needs to include at least enough to encompass all defined autokeys (which incidentally are integer types under the hood.) The difference here being their buffer length needs to entirely include an autokey, unlike integer and decimal which is ok if it doesn't include any of the key. They both don't allow buffer length to end in the middle of these keys.
\$ERR_INVPNHAND	545	Invalid pen handle An invalid pen memory handle was passed during a pen operation (%WPR_INFO( <i>report_handle</i> , DWP_GETPEN)) in the Windows printing API. Correct the pen handle.
\$ERR_INVPRC	224	Invalid fractional precision During evaluation of an arithmetic expression, the number of digits in the fractional portion of an implied-decimal variable exceeded 28.

## Error Messages

### Runtime Errors

\$ERR_INVRCBHND	570	Invalid RCB handle	An invalid memory handle was specified in the <i>rcbid</i> parameter of an RCB call.
\$ERR_INVRPTHND	544	Invalid report handle	An invalid report memory handle was used during a report operation in the Windows printing API.
\$ERR_INVWNDHND	572	Invalid window handle	A bad window handle was passed to the %W_INFO(WIF_USRMEM) or W_FLDS( <i>id</i> , WF_USER) routine.
\$ERR_IOFAIL	22	Failure during I/O operation	<p>A system error that indicates the data transfer was incorrect was returned during an I/O operation. One possible reason is that the I/O statement tried to access a terminal device that wasn't ready. (Perhaps the device was offline, the modem hung up, the device was a network device that had been disconnected, and so forth.)</p> <p>To access the underlying system error that caused \$ERR_IOFAIL, we recommend that you capture %SYSERR <i>immediately</i> after the error is trapped and use that value in the diagnosis process.</p>

---

### WIN, UNIX

For ISAM files, see the *stv* argument of the ERROR routine for further clarification. (See [%ERROR](#) in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual*.) In most cases, an \$ERR\_IOFAIL error indicates ISAM file corruption. Use **isutl -vi** to detect corruption and follow the repair recommendations. If the problem persists and you are not aborting programs abnormally and have not had system crashes, contact Synergy/DE Developer Support. (See [“Product support information” on page x](#) for details about contacting Support.)

---

### VMS

You might get this error with the SS\$\_DATAOVERUN system error code if the type-ahead buffer is filled and the terminal is set **nohostsyc** (a normal OpenVMS error condition). To avoid this error, either include an error list in your ACCEPT statements or ensure that the terminal is set **hostsyc**. If the terminal cannot be set **hostsyc**, you can set the terminal **altype** to reduce the occurrence of this error. You can retrieve additional information for RMS files using the *stv* argument of the



ERROR routine. (See [%ERROR](#) in the “System-Supplied Subroutines and Functions” chapter of the *Synergy DBL Language Reference Manual*.)

---

\$ERR_IOMODE	108	Bad mode specified  In an OPEN statement, you specified a mode or submode that is invalid or that conflicts with the file organization.
\$ERR_IORDXP	154	Only integer and decimal operands allowed  The round value (operand on the right) that you specify to the rounding operator (#) must be integer or decimal data type.
\$ERR_IRCSIZ	316	Invalid record size  You specified an invalid record size. This error occurs primarily on STORE, WRITE, or WRITES operations and may occur when you output to an ISAM or relative file and the buffer you’re passing is larger than the record size.
\$ERR_IRNDVAL	156	Invalid round value for integer operand: %d  The round value (operand on the right) that you specify to the rounding operator (#) for an integer value to round (operand on the left) must be less than or equal to 28.
\$ERR_JOINENUM	647	Invalid enumerator type for Select object with Join  The Select object being enumerated contains an InnerJoin or LeftJoin. Prior to enumerating, the Select object must be converted to a JoinSelect object first using the Select.Join() method.
\$ERR_JOINISMREQ	646	ISAM file required for inner table: %s  You attempted to specify a non-ISAM file as an inner table. Only ISAM files are allowed as inner tables.
\$ERR_JOINKEYREQ	642	Inner table requires key reference: %s  A column specified in an On expression object that references an inner table must be a key or partial key.
\$ERR_JOINONREQ	645	Valid On expression required  An On expression is missing. Each Join requires an On class expression that defines how two files are to be joined via common fields.

## Error Messages

### Runtime Errors

\$ERR_JOINOPER	644	Invalid Join predicate operator	The first (leftmost) expression in the On class (known as the “Join predicate”) does not contain an equality operator (==, .EQ., or .EQS.). On requires at least one equal expression that references the righthand inner table and a lefthand outer table (in scope).
\$ERR_JOINOUTREF	643	Outer table reference required for inner table: %s	No outer table reference exists for the specified inner table. Columns specified in an On expression must either explicitly or implicitly reference the appropriate From record used in the current or “outer” Join (where “outer” Join refers to a Join occurring earlier in a nested Join scenario, not, for example, a “Left Outer Join”).
\$ERR_KEYNOT	53	Key not same	<p>The specified key value doesn’t match an existing record in the file. This error is related to I/O operations involving ISAM files and occurs for any of the following reasons:</p> <ul style="list-style-type: none"><li>▶ If the key field specified in a READ or FIND statement is at least as long as that defined for the ISAM file, Synergy DBL assumes the READ or FIND is requesting a record whose key value exactly matches the value of the designated key field. If no record’s key field begins exactly as specified by the key value, this error occurs, and the first record with a higher key value is returned, and if the file is open for update, the record is locked. (Even though you get an error, the I/O is completed.)</li><li>▶ During a WRITE operation, the key value does not match the value of the stored record exactly, and the index does not allow modification.</li></ul>
\$ERR_LIBMAX	330	Maximum open libraries exceeded	You’ve attempted to open more than 256 libraries at one time.
\$ERR_LOCKED	40	Record is locked	You’ve attempted to access a record or group of records that is being used by another user.
\$ERR_LPQERR	256	LPQUE failed	An error occurred on the LPQUE statement. Use %SYSERR to obtain (or access) system-specific error codes.
\$ERR_MAXIF	141	Too many input files open	You’ve exceeded the maximum number of indirect command input files.

\$ERR_MAXPRC	308	Too many processes The attempt to create a new process failed because the maximum number of allowed processes was reached.
\$ERR_MISSFLD	427	Field/Type/Property/Event not found The specified field name does not represent a public or public static field defined in the assembly associated with the specified object, or the specified field, type, property, or event could not be found.
\$ERR_MISSMETH	426	Method/Delegate not found The specified method name does not represent a public or public static method defined in the assembly associated with the specified object, or the specified method or delegate could not be found.
\$ERR_MRGERR	226	Merge error An error occurred during the processing of the MERGE statement.
\$ERR_MSGFAIL	324	SEND/RECV message failure Some failure has caused a message not to be sent or received. The most common causes for this error are that either you've exceeded the maximum message size of 4080 bytes or the Synergy message manager is not running.  This message is normal on a RECV statement if many messages are queued up and the message manager has not yet processed them all. In such a case, the RECV has queued another message in sequence, but the message manager has not processed it in the time the runtime has allotted for a reply. You should retry the RECV if you're sure the message manager is actually running. This error protects the program from hanging if the message manager aborts; however, the time to wait depends on the processing activity and speed of the system, so you must determine if a retry is required.
\$ERR_NDNOPP	598	File built with opposite 'endian': %s An attempt was made to load an ELB built with the wrong byte order for this machine.
\$ERR_NETCONFIG	331	Local network configuration error A TCP socket call failed.
\$ERR_NETCRYPT	432	File requires network encryption An OPEN statement referenced a file with the network encryption flag set via a network path specification (NFS or Windows network mapped drive), and encryption has not been enabled on the server.

## Error Messages

### Runtime Errors

\$ERR_NETPROB	320	Network problem reaching server %s	A problem was detected while trying to communicate with Synergy/DE xfServer or xfServerPlus. Either an attempt was made to make a call on a disconnected socket, or the socket connection was lost during the call. Try closing all your channels and reopening them.
\$ERR_NOCHAIN	101	Stop chain not allowed in callback on Windows 7 or Windows X64	A Windows callback attempted to perform a STOP chain on Windows 7 or higher or on 64-bit Windows platforms. Because these systems prevent exceptions from being thrown across the kernel (user32.dll) boundary, a Windows callback (for example, a .NET assembly API delegate called from a WinForm or an ActiveX event called from a UI control) cannot perform a STOP chain without causing problems. Therefore, this action is not allowed.
\$ERR_NOCURR	61	No current record	You haven't established a current record, and one is required for the I/O operation you're attempting. This error can occur when during a DELETE or WRITE operation, a FIND, READ, or READS operation does not logically precede the update attempt.
\$ERR_NODBGPORT	607	Debug port number not specified: %s	A port number was not specified on the <b>dbf -rd</b> command. The number of the port on which you want the debug server to listen as a Telnet server for the debug client must be specified.
\$ERR_NODOTNET	430	Could not load the .NET CLR	The .NET common language runtime (CLR) could not be loaded for any one of a variety of reasons, including that it is not installed.
\$ERR_NODUPS	54	Duplicate key specified	An output I/O operation attempted to store a duplicate key value in an ISAM file that doesn't allow duplicates.
\$ERR_NOFDL	533	Invalid open mode for FDL usage	An FDL qualifier was specified in an OPEN for output statement.
\$ERR_NOFORK	311	Cannot fork	A system fork call failed.
\$ERR_NOLOAD	428	Could not load assembly	The assembly could not be loaded.

\$ERR_NOMEM	9	Not enough memory for desired operation  This operation could not be performed with the available memory. This error only occurs after all memory has been reorganized and all unnecessary segments are freed. If you get this error, either decrease the size of your routine or increase the amount of available RAM.
\$ERR_NOMETHOD	162	Method's routine not found  The method you called is not a member of a class.
\$ERR_NOMORECURS	333	SQL: No more available open cursors  See <a href="#">“Trapping runtime error messages in SQL Connection programs”</a> in the “Error Logging and Messages” chapter of the <i>SQL Connection Reference Manual</i> .
\$ERR_NONETSHR	634	Network share is not allowed with this file  You've attempted to open a file that contains a timestamp key in update mode on a mapped drive. Files with timestamp keys cannot be opened in update mode across a network share or NFS drive. Use xfServer instead.
\$ERR_NOOBJ	601	No object for handle  An object handle that does not contain an object instance was used.
\$ERR_NOOPEN	11	Channel has not been opened  You've attempted an I/O operation on a channel that has not been activated by the OPEN statement.
\$ERR_NORETURN	622	Leaving local scope where a CALLED subroutine is still active  A routine was CALLED from within a compound statement that defines handles locally, and the compound statement was exited before a RETURN was executed.
\$ERR_NOSERVER	321	Synergy server on %s is not running or has been shut down  A client attempted to access a server on a host where the server has not been started or has been shut down.
\$ERR_NOSPAC	24	No space exists for file on device  On an OPEN statement in output mode, the operating system indicated that the device didn't have enough space left for the file to be opened or the record to be stored. This error can occur either because fewer storage blocks are available than are requested during preallocation of the file or because the directory structure for the device cannot accommodate more files.

The Novell operating system allows user log-ins to be limited in the amount of disk space that the user can allocate. Novell version 3 and higher allows directories to have disk space restrictions as well. Such restrictions can cause \$ERR\_NOSPAC errors, even if hundreds of megabytes are available on the disk. The network administrator can use the Novell **dspace.exe** utility to check and change user and directory restrictions if such errors occur. Your users may want to consider eliminating these restrictions for certain directories and users.

\$ERR_NOSQL	80	SQL Connection installation error or DBLOPT 48 not set You've called an SQL Connection routine before the Connection was initialized. SQL Connection is initialized by setting system option #48 using either DBLOPT or %OPTION.
\$ERR_NOTAVL	19	Device not available The device you've attempted to access is not available.
\$ERR_NOTCLSHND	584	Handle is not a class handle The handle is a valid memory handle, but it is not a class handle.
\$ERR_NOTISM	56	Not an ISAM file One of the following has occurred: <ul style="list-style-type: none"><li>▶ You've attempted to open a file that is not recognized as an ISAM file.</li><li>▶ You've specified a channel in the GETRFA, POSRFA, ISKEY, or ISSTS subroutine that was not opened to an ISAM file.</li><li>▶ You've specified a file in the ISCLR subroutine that is not recognized as an ISAM file.</li><li>▶ You've attempted to open an ISAM file whose index and data headers don't match, indicating that either another instance or revision of the file was copied over just one of the files (.ism or .is1), or one of the files has become corrupted.</li></ul>
\$ERR_NOTNAMHND	583	Handle is not a namespace handle The memory handle passed as a namespace ID in one of the %NSPC_ routines is a valid memory handle, but it is not a handle to a namespace. Correct the namespace ID.
\$ERR_NOTNETHND	581	Handle is not a network handle This is an xfNetLink Synergy error. The memory handle passed as a network connection ID is a valid memory handle, but it is not an ID to a network connection. Correct the network connection ID.

\$ERR_NOTOHND	602	Both source and destination must be object handles One of the operands in an assignment operation on object handles is not an object handle.
\$ERR_NOTPNHAND	589	Handle is not a pen handle The memory handle passed as <i>report_handle</i> in <i>%WPR_INFO(report_handle, DWP_GETPEN)</i> is a valid memory handle, but it is not a pen handle. Correct the <i>report_handle</i> parameter.
\$ERR_NOTRCBHND	580	Handle is not an RCB handle The memory handle passed in the <i>rcbid</i> parameter of an RCB call is a valid memory handle, but it is not a handle to a routine call block. Correct the <i>rcbid</i> parameter.
\$ERR_NOTRPTHND	588	Handle is not a report handle The memory handle passed during a report operation in the Windows printing API is a valid memory handle, but it is not a handle to a report operation. Correct the <i>report_handle</i> parameter.
\$ERR_NOTWNDHND	582	Handle is not a window handle The memory handle passed as a user data set ID in the <i>%W_INFO(WIF_USRMEM)</i> or <i>W_FLDS(id, WF_USER)</i> routine is a valid memory handle, but it is not an ID to a user data set in a Synergy window. Correct the user data set ID.
\$ERR_NOXCAL	254	Undefined XCALL referenced At least one of your program's referenced ELBs was modified to refer to a new routine that is undefined. When you ran your program, it attempted to invoke a routine that referenced the undefined subroutine or function.
\$ERR_NULARG	322	Improper use of null argument You passed a null argument to a data reference operation, intrinsic function, or system-supplied external subroutine that requires a nonnull argument. Null arguments are passed either by not specifying an argument in the call or by specifying an argument that is itself an argument that was not passed when the current routine was called.
\$ERR_NULLREF	429	Invalid use of NULL object An attempt was made to dereference a null object reference.
\$ERR_NUMXP	166	Numeric argument expected A nonnumeric argument was passed to a subroutine that required a numeric argument.

## Error Messages

### Runtime Errors

\$ERR_OBJPASSED	623	Unexpected object handle passed as argument An object handle was passed as an argument to a routine, but the routine does not declare the parameter as the appropriate object handle type.
\$ERR_ODIGXP	147	Octal digits expected in argument (%s) You must specify an argument that contains only octal digits (0 – 7).
\$ERR_OHNDCPY	606	Invalid copy of an object handle An object handle was overwritten or cleared in such a way that the runtime was unaware of its type. This can be caused by oversubscripting a field in a record and changing the contents of another field declared as an object handle.
\$ERR_OHNDREQ	604	Object handle required A parameter declared as an object handle was not passed an object handle.
\$ERR_OLDELB	595	Old ELB file format%s detected: relink %s The ELB file format that you linked with is no longer supported. You must relink with a newer version. (We recommend linking with the current version.)
\$ERR_ONLYWR	12	Attempt to open output device in input mode An OPEN statement attempted to open an output-only device, such as a line printer, using input (I) mode.
\$ERR_OPNERR	95	OPEN error An error occurred in an OPEN statement. To access the underlying system error that caused \$ERR_OPNERR, we recommend that you capture %SYSERR <i>immediately</i> after the error is trapped, and use that value in the diagnosis process.
\$ERR_OPTINV	547	Invalid option You specified an invalid option.
\$ERR_OUTRDO	39	Output to read-only device An I/O statement attempted to perform output to a device that is write-locked.
\$ERR_OUTRNG	104	Value out of range A statement or subroutine argument or qualifier was outside the permitted range of values.



\$ERR_PROTEC	62	Protection violation You've attempted to access a resource that is protected.
\$ERR_PRTOBJHND	576	Protected object handle cannot be deleted You've attempted to use %MEM_PROC(DM_FREE) on a memory handle allocated with the Synergy XML API or xfServerPlus. Only the XML API or xfServerPlus can delete these memory handles.
\$ERR_PURGE	530	DCL purge error An error occurred in the purge.
\$ERR_QUEUENOTAV	122	Invalid queue specified on LPQUE The queue specified in an LPQUE statement was either not available or invalid.  \$ERR_QUEUENOTAV is a catchall for most print/batch symbiont errors. You should use the %SYSERR intrinsic function to retrieve the system error code to decode these errors further.
\$ERR_RCBCALL	904	RCB: Call in progress  An attempt was made to modify an RCB argument block when an RCB call that uses the same RCB argument block is being executed. It can occur when using any of these functions: %RCB_CREATE, RCB_DELARG, RCB_DELETE, RCB_INSARG, RCB_SETARG, RCB_SETARGS, RCB_SETFNC, and RX_SETTRMTFNC.
\$ERR_RCBDYN	338	RCB: ^M variable still bound on dynamic memory deletion%s  ^M memory (%MEM_PROC) is released, but some or all of the memory is still set as an argument for %RCB_CALL. The checking for this condition only occurs when the routine has been compiled with <b>-qcheck</b> .
\$ERR_RCBOBJ	905	RCB: Class variable still bound/defined on destruction%s  A variable that is part of a class was specified as an RCB argument, and the class instance containing the variable has been released.
\$ERR_RCBSTACK	339	RCB: Stack variable still bound on routine exit%s  Stack record memory is still set as an argument for %RCB_CALL when returning from a routine. The checking for this condition only occurs when the routine has been compiled with <b>-qcheck</b> .
\$ERR_RECBLK	301	Record must be a multiple of block size  The record size specified in a block I/O statement was not a multiple of 512 bytes.

## Error Messages

### Runtime Errors

\$ERR_RECEXTCAL	5	Recursive XCALL or .NET CALL exceeds 64 An external subroutine called itself or, in Synergy .NET, more than 64 nested local CALLs occurred.
\$ERR_RECLIMIT	639	File record limit exceeded A STORE operation was performed to a file with a defined record limit that would cause the number of records to exceed the specified limit.
\$ERR_RECLNG	86	Invalid record length Either a record that exceeds the length of the specified record was encountered during SORT processing or you've passed an invalid record to the ISAMC subroutine.
\$ERR_RECNOT	431	Record not same A record accessed via a READ statement (or a WRITE statement for relative files) using the RFA qualifier with a GRFA no longer matches the original record at the time the GRFA was created.
\$ERR_RECNUM	28	Illegal record number specified A READ, GET, WRITE, or PUT statement specified a record number that was either less than one or greater than the number of records present in the file currently associated with the I/O channel.
\$ERR_RELREC	313	Invalid relative record You've attempted to read a record that hasn't been written yet or a record that is corrupted.
\$ERR_REPLAC	32	Cannot supersede existing file You've attempted to overwrite a file that has been protected against deletion. A possible cause of this error is that the OPEN statement, or the RENAM or ISAMC subroutine attempted to create a new file that already exists. This error condition is only detected if the FLAGS subroutine runtime option flag 3 is set.
\$ERR_REVUNSUP	635	Specified file revision not supported (ISAMC_REV) You've attempted to create an ISAM file with ISAMC_REV set to a value less than 4. Only values of 4 or higher are supported.
\$ERR_RMSERROR	100	Unexpected RMS error This error only occurs on OpenVMS. Synergy DBL cannot access the file it's attempting to access.

\$ERR_RNDVAL	155	Invalid round value: %d The round value (operand on the right) that you specify to the rounding operator (#) must be less than or equal to 28.
\$ERR_RNF	64	Record not found The specified record does not exist in a relative file, or a MATCH:Q_EQ was performed on an ISAM file and the key does not exist.
\$ERR_RTNNF	511	Cannot access external routine %s You attempted to either execute a routine (using XSUBR or ^XADDR) that cannot be found, or to execute a routine (with or without XSUBR or ^XADDR) that calls a routine that cannot be found. This error may occur if you remove a routine from an ELB that is still referenced in the main routine or linked ELBs. The best way to avoid this problem is to always relink main routines when you change ELBs, or for OPENELB, XSBUBR, or ^XADDR, use the <b>-r</b> and <b>-l dblink</b> options to ensure symbols are resolved when you link your ELBs.
\$ERR_SALTIV	434	Error getting salt/initialization vector: %s An SSL error occurred as a result of calling DATA_SALTIV.
\$ERR_SDMSERR	591	SDMS error An problem occurred when using the %ISINFO routine. This could be because a null string was requested but no null string exists for the specified key, or some other %ISINFO error.
\$ERR_SEQRDS	615	Sequential read caching error You are using the xfServer prefetch feature, and one of the instances where it is not allowed has occurred. Immediately try doing a keyed READ to reset your current position. If this doesn't work, turn off prefetching for this file. (See <a href="#">“Prefetching records to improve performance with xfServer”</a> in the “Synergy DBL Statements” chapter of the <i>Synergy DBL Language Reference Manual</i> for more information.)
\$ERR_SETTYP	323	SET data types must be the same The variables you specified in the SET statement did not have the same data type.
\$ERR_SINGLEDIM	618	Array is not a one-dimensional array Some methods in System.Array (GetValue, SetValue, IndexOf, and LastIndexOf) require a single-dimension, or pseudo, array.

## Error Messages

### Runtime Errors

\$ERR_SIZLIMIT	638	File size limit exceeded  A STORE or WRITE operation either caused an ISAM file created with the SIZE_LIMIT option to extend beyond the specified limit or caused a file not configured with the TBYTE option to exceed the 2 GB limit.
\$ERR_SMERR	131	SORT or MERGE error  An error occurred during processing of a SORT or MERGE statement.
\$ERR_SQLDYN	337	SQL: ^M variable still bound/defined on dynamic memory deletion  You used %SSC_BIND, %SSC_DEFINE, or %SSC_OPEN using a variable defined with a memory handle (^M), and the cursor was not closed before the dynamic memory was deleted. The checking for this condition only occurs when the routine has been compiled with <b>-qcheck</b> .
\$ERR_SQLERR	332	Initialize Synergy SQL by calling %INIT_SSQ first  Initialize Synergy SQL by setting DBLOPT 48  Synergy SQL ERROR: uninitialized system called  Synergy SQL ERROR: Licensing error. Demo period expired  Synergy SQL ERROR: Licensing error. Maximum users exceeded  Synergy SQL ERROR: Licensing error. Product not installed  See <a href="#">“Trapping runtime error messages in SQL Connection programs”</a> in the “Error Logging and Messages” chapter of the <i>SQL Connection Reference Manual</i> .
\$ERR_SQLSTACK	336	SQL: Stack variable still bound/defined on routine exit  You used %SSC_BIND, %SSC_DEFINE, or %SSC_OPEN using a variable defined in a stack record, and the cursor was not closed before the routine exited. The checking for this condition only occurs when the routine has been compiled with <b>-qcheck</b> .
\$ERR_SRTFAI	243	SORT failure  An error occurred while Synergy DBL was processing a sort (for example, if you were attempting to sort in a read-only directory).
\$ERR_SRVCONRTY	637	Server connection retry failure  A client process failed to reconnect to the server and the RetryTime has expired. Mapped to NetworkException class.

\$ERR_SRVFILCL	640	Remote file has been closed An xfServer client attempted to access a file opened with an exclusive SHARE on the server that was closed after the connection recovery KeepLocks time expired.
\$ERR_SRVREXP	636	Server session has expired or has been terminated. A client process is trying to connect to a client context that has expired and been discarded. Mapped to ServerNotRunningException class.
\$ERR_SRVLICERR	536	Licensing error on server %s The server is not licensed on the host specified.
\$ERR_SRVRLICNS	534	Server license limit reached on %s The server on the specified host has reached the maximum number of licensed connections.
\$ERR_SRVLICTIMO	537	Licensing timed out on server %s The server license has expired on the specified host.
\$ERR_SRVNOTSUP	563	Unsupported server version/Feature not available Either the version of xfServer is lower than the version for the client system, or the feature you attempted to use is not supported in the version of the server requested.
\$ERR_STRMTCH	624	Structure mismatch between routines A structure was passed as a structfield to a routine, or a boxed structure in an object handle doesn't match the same structure definition referenced in a prior routine.
\$ERR_SUBSCR	7	Invalid subscript specified A value specified as a subscript is outside the allowable range of values, which could indicate one of the following situations: <ul style="list-style-type: none"><li>▶ The starting position for a variable in a subscripted, ranged, or indexed expression is less than or equal to zero.</li><li>▶ The ending position in a ranged expression is less than the starting position.</li><li>▶ The memory area referenced through a subscripted, ranged, or indexed expression is outside the bounds of either the routine's local data area, a calling routine's data area (in the case of parameters), or a global data area (in the case of global variables).</li><li>▶ A real array is referenced with [0].</li></ul>

## Error Messages

### Runtime Errors

\$ERR_SYNSOCK	569	Synsock error %d	xfNetLink Synergy has experienced a socket failure. For additional information about the error, use the <code>RX_GET_ERRINFO</code> subroutine, which stores the socket error level that would be returned as a status if you made the socket calls directly yourself. Restart your system and retry the operation. A recurrent error generally indicates a problem with your network.
\$ERR_TIMEOUT	111	Terminal input operation timeout	One of the following occurred: <ul style="list-style-type: none"><li>▶ You specified the <code>WAIT</code> qualifier with a value on an <code>ACCEPT</code> or <code>READS</code> statement to await input from a terminal, and the user didn't enter the required input before the specified amount of time expired.</li><li>▶ The xfNetLink Synergy client timed out after waiting specified length of time for call results. You can either extend the time-out at runtime using <code>%RX_RMT_TIMEOUT</code>, optimize the called routine, or check with your network administrator.</li><li>▶ The message manager mailbox specified in a <code>RECV</code> statement exists, but the message manager is not running.</li></ul>
\$ERR_TOOBIG	23	Input data size exceeds destination size	You've attempted to read data into a destination that is not large enough to contain the complete data transfer. The full record is read and the data is truncated to the size of your buffer.
\$ERR_TOOLKIT	614	Toolkit error	Toolkit has encountered an error that is not a standard Synergy DBL error, or there has been an explicit call to <code>U_ABORT</code> . Use <code>%ERR_TRACEBACK</code> to retrieve associated messages. See <a href="#">U_ABORT</a> in the "Utility Routines" chapter of the <i>UI Toolkit Reference Manual</i> for more information.
\$ERR_UNDEFERR	327	Undefined error	You've attempted to signal an error that is unknown to Synergy DBL.
\$ERR_UNHANDLED	621	Unhandled exception: %s	A <code>THROW</code> of an exception was not caught by an outer <code>TRY/CATCH</code> .
\$ERR_UPDNFD	27	Update of non-file device	You've attempted to open a nonfile device in update mode.

\$ERR_WINRSRC	538	Windows resource exhausted	An attempt to allocate a Windows resource failed. This error should be followed by the system error message generated by Windows, which provides additional information.
\$ERR_WNDERR	329	Window Manager error	A window error occurred. The specific window error message should be displayed below this error. See <a href="#">“Window error messages” on page 5-49</a> .
\$ERR_WNFCERR	543	Windows API function failure: %s	A call to the Windows API failed. The name of the failing function should be displayed along with the Windows system error number. (For more information about this error, refer to Microsoft’s system error documentation.)
\$ERR_WROARG	6	Incorrect number of subroutine arguments	The number of arguments passed to an external subroutine or intrinsic function is different than the number of arguments expected by the subroutine or intrinsic function.
\$ERR_WRONGTHREAD	701	.NET xfServer client I/O on wrong thread	Open channels were not explicitly closed after a call to S_SERVER_THREAD_INIT. When S_SERVER_THREAD_INIT is used to initialize a thread for a new xfServer connection, you must close all channels opened on that thread after the call to S_SERVER_THREAD_INIT, before the thread is destroyed, and never allow the channels to be closed on the garbage collector thread.
\$ERR_WRTLIT	8	Writing to a literal or missing argument	You’ve attempted to change the value of an alpha, decimal, implied-decimal, or integer literal. This error normally occurs because you’ve passed a literal or an expression to a system-supplied external subroutine or intrinsic function that is expecting a variable, and the subroutine tried to modify the argument.
\$ERR_XFBADARRAY	556	Error mapping array element	An xfNetLink Synergy error occurred while mapping array elements. For additional information about the error, use the RX_GET_ERRINFO subroutine. Contact Synergy/DE Developer Support if you need assistance. (See <a href="#">“Product support information” on page x</a> for details about contacting Support.)

## Error Messages

### Runtime Errors

\$ERR_XFBADMTHID	551	Method ID too long  The specified method ID exceeds 31 characters. For additional information about this xfNetLink Synergy error, use the RX_GET_ERRINFO subroutine.
\$ERR_XFBADPKT	553	Packet format error  A parsing error has occurred in xfServerPlus. Because this error is sometimes caused by network noise, try the operation again. If you get the same error a second time, contact Synergy/DE Developer Support for assistance. (See <a href="#">“Product support information” on page x</a> for details about contacting Support.)
\$ERR_XFBADPKTID	550	Incorrect packet identifier  The parser cannot parse the return response due to an invalid character in the packet type field. Because this xfNetLink Synergy error is sometimes caused by network noise, try the operation again. If you get the same error a second time, contact Synergy/DE Developer Support for assistance. (See <a href="#">“Product support information” on page x</a> for details about contacting Support.) If you are connecting a newer server to an older client, you may need to upgrade the client.
\$ERR_XFBADTYPE	554	Invalid parameter type  The argument type didn’t correspond to the definition in the Synergy Method Catalog. For additional information about this xfNetLink Synergy error, use the RX_GET_ERRINFO subroutine, and then check your routine call against the definition in the Synergy Method Catalog.
\$ERR_XFHALT	561	Fatal error occurred on server  xfServerPlus encountered a fatal, untrapped error in one of the Synergy routines being called remotely. For additional information about the error, use the RX_GET_HALTINFO subroutine. To solve the problem, check the routine for untrapped errors. Be sure to check the number and type of parameters. Then restore the system as required and restart the session.
\$ERR_XFINCALL	592	Remote call already in progress  An %RXSUBR call has timed out and you’ve attempted to make a second %RXSUBR call before receiving the return packet for the first call. Use %RX_CONTINUE to complete the timed-out %RXSUBR call before making another remote call. Or, you can use RX_SHUTDOWN_REMOTE to shut down the session.



\$ERR_XFIOERR	557	File I/O error occurred on server  A file I/O error occurred in a program on the xfServerPlus machine. For additional information about the error, use the RX_GET_ERRINFO subroutine. To solve the problem, correct your code or change the file attributes as indicated by the specific I/O error.
\$ERR_XFMETHCRYPT	596	Method requires encryption  The method is marked for encryption in the SMC, but the client sent clear data.
\$ERR_XFMETHKNF	558	Method key not found  xfServerPlus could not find the method ID. For additional information about the error, use the RX_GET_ERRINFO subroutine, and then check your routine call against the definition in the Synergy Method Catalog. Remember that the method ID is case sensitive.
\$ERR_XFNOCALL	593	No current call in progress  You have attempted to call %RX_CONTINUE when no %RXSUBR call has timed out. %RX_CONTINUE can be used only when a remote call has timed out.
\$ERR_XFNOCDT	565	Unable to open method catalog file  xfServerPlus could not open the catalog file of the Synergy Method Catalog ( <b>cdt.ism</b> ), most likely because it could not find it, the file is corrupted, or it is in a format prior to 8.3. If the SMC is not in the default location, make sure XFPL_SMCPATH is set correctly.
\$ERR_XFNOCMPDT	566	Unable to open method parameter file  xfServerPlus could not open the method parameter file of the Synergy Method Catalog ( <b>cmpdt.ism</b> ), most likely because it either could not find it or the file is corrupted. If the SMC is not in the default location, make sure XFPL_SMCPATH is set correctly.
\$ERR_XFNOCONN	560	No connection to remote host  This is an xfNetLink Synergy error. The connection to the host was lost. Restart the session. If the transaction was interrupted in mid-stream, you may need to restore the system to a valid state before restarting.
\$ERR_XFNOELB	567	Unable to open ELB file  xfServerPlus could not locate or open the specified ELB. Make sure you are using the correct ELB name in the SMC and that the logicals you are using to point to ELBs are correctly defined.

## Error Messages

### Runtime Errors

\$ERR_XFNOINIT	562	RX_DEBUG_START called without RX_DEBUG_INIT This is an is xfNetLink Synergy error. RX_DEBUG_START was called before a corresponding RX_DEBUG_INIT call was made.
\$ERR_XFNUMPARMS	552	Invalid parameter count An invalid number of arguments was passed to xfServerPlus from xfNetLink Synergy. For additional information about the error, use the RX_GET_ERRINFO subroutine, and then check your routine call against the definition in the Synergy Method Catalog.
\$ERR_XFREQPARM	555	Required parameter not sent A required argument was not passed to xfServerPlus from xfNetLink Synergy. For additional information about the error, use the RX_GET_ERRINFO subroutine, and then check your routine call against the definition in the Synergy Method Catalog.
\$ERR_XFRTNNF	559	Cannot access remote routine xfServerPlus could not find the method in the specified ELB or shared image. For additional information about the error, use the RX_GET_ERRINFO subroutine. Make sure the correct ELB is specified in the Synergy Method Catalog and that logicals are set correctly to find the ELB.
\$ERR_XFSERVNOSEC	597	Encryption not enabled on server Encryption was not detected on the server while connecting to an xfServerPlus server and requesting encryption “/ENCRYPT”.
\$ERR_XFUNKERR	564	Unknown error reported by xfServerPlus The server returned an error that was not recognized by the client. Check the <b>xfpl.log</b> file, which records the error even though the client is unable to receive it. This error usually happens when an older xfNetLink Synergy client is communicating with a newer xfServerPlus server. To solve this problem, update your client version. If your versions already match, call Synergy/DE Developer Support. (See <a href="#">“Product support information” on page x</a> for details about contacting Support.)

## Informational error messages

The following errors provide additional information about other errors.

ACCVIO	1001	Access violation You tried to index a variable outside of the data area.
--------	------	---

ALITXP	1002	Alpha literal expected You specified a value in the mode option for the OPEN statement that wasn't an alpha literal.
AMBKWD	1218	Ambiguous XDL keyword: %s An XDL keyword was abbreviated to the point that it could not be distinguished from another XDL keyword. For example, you cannot abbreviate DENSITY to "D" because it cannot be distinguished from the DUPLICATES keyword.
AMBVAL	1224	Ambiguous %s value: %s An XDL keyword value was abbreviated to the point that it could not be distinguished from another possible value.
ARGWAS	1003	Argument number was %d An error occurred when processing the specified argument.
ATLIN	1193	At line %d in routine %s An error occurred at the specified line in the specified routine.
ATLINE	1195	At line %s in routine %s An error occurred at the specified line in the specified routine.
BADDSCR	1217	Corrupted descriptor: type = %d, class = %d All Synergy DBL variables are referenced by descriptor, which contains a pointer to the variable's data and its length. The Synergy runtime has encountered an invalid descriptor.
BADIND	1011	Bad index: %d You specified this illegal index value.
BADRNG	1012	Bad range value: %d,%d You specified this illegal range value.
BADRNGR	1013	Bad range value: %d:%d You specified this illegal range value.
BADXDLF	342	Bad XDL file An invalid XDL keyword file was specified. See <a href="#">"ISAM Definition Language" on page 3-89</a> for a list of rules that apply to XDL keyword files. If any of these rules are broken, this message could be generated.

## Error Messages

### Runtime Errors

BADXDLS	343	Bad XDL string An invalid XDL string was specified. See <a href="#">“ISAM Definition Language” on page 3-89</a> for a list of rules that apply to XDL keyword files. If any of these rules are broken, this message could be generated.
CALFRM	1196	Called from line %s The detected error was called from the specified line.
CALFRO	1194	Called from line %d The detected error was called from the specified line.
CHNWAS	1021	Channel specified: %u An I/O error occurred on this channel.
CHRSPC	1022	Character specified: %c You specified this character as the end value in a MERGE statement.
COLEQL	1029	Colon or equal sign expected A syntax error occurred either in the qualifier specifications, in the OPEN statement, or in the KEY specification of the OPTION qualifier in the SORT or MERGE statement.
CONSUP	1216	Please contact your Synergy/DE supplier If you get this error, contact Synergex or the company that provides your Synergy/DE products.
CREFIL	1030	Error creating file An error occurred during file creation. Files are created when you specify output mode on an OPEN statement and when you use the ISAMC subroutine.
DBLDIR	1040	DBLDIR not set You did not set the DBLDIR environment variable, which is required to be set to the directory that contains your Synergy/DE distribution.
DCMPER	1041	Data compression/uncompression error You’ve attempted to READ(S) or WRITE a record that is corrupted in an ISAM file with compressed data records.

DECXP	1042	Decimal expected	<p>A decimal variable or literal was expected in one of the following situations:</p> <ul style="list-style-type: none"><li>▶ In the OPEN statement's SIZE or RECSIZE specification</li><li>▶ In the KEY specification of the SORT or MERGE statement's OPTION string</li></ul>
DELFIL	1043	Error deleting file	<p>An error occurred while the file specified in the LPQUE statement was being deleted.</p>
DEVFUL	1211	Device full	<p>You've attempted to extend a file when the device contains no free space.</p>
DIMEXP	1230	Dimensions of passed argument: %d	<p>The specified dimension value was deemed out of range.</p>
DIMSPC	1229	Dimension specified: %d	<p>A dimension was specified that does not lie in the range declared for an array.</p>
DINCON	1046	Data incongruity	<p>Your ISAM data file appears to be corrupted. A pointer to the index file points to an invalid area of the data file.</p>
DRCSIZ	1047	Destination record size: %d	<p>The data area you specified in an I/O statement is this size.</p>
EQLEXP	1055	Equal sign expected	<p>An equal sign was expected after a qualifier in the SORT or MERGE statement.</p>
ERTEXT	1052	%s	<p>This informational error displays different text for different situations, including operating system errors. It explains an accompanying error in more detail.</p>
ERTEXTT	1228	%s	<p>This informational error is used for generic information.</p>

## Error Messages

### Runtime Errors

ERTXT2	1053	%s %s	This informational error displays different text for different situations, including operating system errors. It explains an accompanying error in more detail.
ERTXTN	1054	%s %d	This informational error displays different text for different situations, including operating system errors. It explains an accompanying error in more detail.
EXECF	1056	Cannot execute: %s	An attempt to execute a program using the EXEC or RUNJB subroutine was unsuccessful.
EXPDEMO	1213	This system has timed out	Your 14-day demo period has expired. Please contact Synergex or your Synergy/DE supplier for a configuration key.
EXUSER	1214	Exceeded concurrent user capacity	The maximum license capacity in the License Manager has been reached. (In other words, the number of log-ins on your system is greater than the licensed number of users.) Either contact your Synergy/DE supplier for another configuration key so you can increase the number of users, or wait until someone logs out.
FILWAS	1061	File specification was %s	The specified file is involved in the error.
FLSPCW	216	File specification was %s	The specified file is involved in the error.
FRCSIZ	1063	File record size: %d	The record size on a READ exceeded the destination size, and this informational error displays the record size. This error only occurs on OpenVMS.
IINCON	1070	Index incongruity	Your ISAM index file appears to be corrupted.
INTCON	1212	Internal consistency failure	An error in runtime processing, usually associated with the License Manager, has occurred.

NUMSPC	1134	Number specified: %ld  The number of CR/LF pairs in the FORMS statement exceeds 9999, or the item passed in the DISPLAY statement is a numeric expression less than 0. (This message is usually accompanied by an \$ERR_BIGNUM error.)
INVAVAL	1222	Invalid %s value: %s  An invalid alpha value was specified for an XDL keyword. Some keywords have a defined set of possible values. For example, the FORMAT keyword requires a value of either “fixed” or “variable.” If any other value is specified, this error is generated.
INVBUFF	1227	Alpha return argument expected  The <i>buf</i> argument to SDMS_ISINFO (which returns an alpha value) was not passed.
INVCMD	1076	Invalid I/O command: %s  You specified this invalid command.
INVDVAL	1223	Invalid %s value: %d  An invalid decimal value was specified for an XDL keyword. Some keywords have a defined set of possible values. For example, the PAGE_SIZE keyword requires a value of 512, 1024, 2048, 4096, 8192, 16384, or 32768. If any other value is specified, this error is generated.
INVIVAL	1226	Numeric return argument expected  The <i>ival</i> argument to SDMS_ISINFO (which returns a numeric value) was not passed.
INVKVAL	340	Invalid key value  The ISAM key value is not valid for the declared data type. For example, if you declare a decimal ISAM key and attempt to STORE a record with an alpha value for that key, this error will be generated.
INVSMD	1079	Invalid OPEN submode  You specified an invalid submode in an OPEN statement.
INVSW	1078	Invalid switch  You specified an invalid switch in the KEY specification of a SORT or MERGE statement.
INVVAL	1080	Invalid value for %s  You specified this invalid value in the LPQUE statement.

## Error Messages

### Runtime Errors

IOERR2	1088	Channel %d, open mode %s This is the channel number and the open mode you specified.
IOOPN	1084	Cannot open %s You specified this indirect filename that cannot be opened.
KEYSPC	1101	Could not locate key with identifier %s You specified an invalid ISAM key of reference.
KEYSPEC	1225	Key specified: %d When an error occurs within a key definition section of an XDL file, this message is generated in addition to the error message to indicate the key definition in which the error occurred.
MAXSIZ	1120	Maximum record size is %u You can specify this maximum record size on a SORT or a MERGE statement.
MLTKWD	1219	Keyword specified multiple times: %s An XDL keyword that is supposed to occur only one time in the file attribute section of the XDL file was found more than once. (The keywords that can only occur once are FILE, NAME, KEYS, ADDRESSING, PAGE_SIZE, RECORD, SIZE, COMPRESS_DATA, FORMAT, and STATIC_RFA.)
MSGBIG	1207	Message exceeds maximum size During a SEND and RECV, you specified a message to send that exceeded 4080 bytes.
MSGEXP	1208	Message communication timeout You attempted to SEND or RECV a message with system option #7, and the Synergy DBL daemon is not running.
NOEOFC	1132	No EOF character found. Physical EOF was used The physical EOF was encountered before the END value you specified in a SORT or MERGE statement.
NOLMD	1210	Cannot access Synergy License Manager The Synergy DBL daemon is not running.
NOTCONF	1215	Synergy Runtime license is not configured Your runtime is not configured. See the <a href="#">“Configuring License Manager”</a> chapter of the <i>Installation Configuration Guide</i> for assistance.



NOVAL	1221	No value supplied with keyword: %s The specified XDL keyword requires a value, but none was assigned.
NUMWAS	1163	Record number: %ld You specified this invalid record number.
OPNFIL	1140	Cannot open file A file could not be opened.
OPTSPC	1142	Option specified %s You specified this option.
OPTWAS	1077	Invalid option: %s You specified this invalid option.
OPWCRE	1149	Operation was ISAMC The error occurred on file creation, either on an ISAMC external subroutine or on an OPEN statement.
OPWDEL	1146	Operation was DELETE The error occurred on a DELETE statement.
OPWFND	1144	Operation was FIND The error occurred on a FIND statement.
OPWRDS	1145	Operation was READS The error occurred on a READS statement.
OPWRED	1143	Operation was READ The error occurred on a READ statement.
OPWSTO	1147	Operation was STORE The error occurred on a STORE statement.
OPWWRI	1148	Operation was WRITE The error occurred on a WRITE statement.
RBKXP	1160	Right bracket expected A closing right bracket was expected on the SIZE specification in an OPEN statement.
READER	1162	Cannot read input file An error occurred while an input file was being read.

## Error Messages

### Runtime Errors

RECWAS	1164	Record size specified: %u You specified this record size.
RENFIL	1166	Error renaming file This error occurred while renaming a file.
REQKWD	1220	Missing required keyword: %s A required XDL keyword is missing. Each XDL description must contain one FILE and one SIZE keyword. In addition, each key definition must contain exactly one LENGTH and one START keyword.
RORKXP	1168	Record or key expected Neither a record nor a key was specified in a SORT or MERGE statement.
RPEXP	1169	Right parenthesis expected A closing right parenthesis was not found in a SORT or MERGE OPTIONS string.
SAMOP	132	Operands must be both alpha or both numeric One operand is alpha and the other operand is numeric. Comparison operations are only allowed when the operands are either both alpha or both numeric.
STKTRC	1231	in %s:line %s This message describes the stack trace of a caught exception.
SYSFLT	1209	System fault (%d) During a SEND or RECV or a call to LM_LOGIN or LM_LOGOUT, the Synergy DBL daemon experienced a system fault while processing your request.
TTSBMD	1185	Submode ignored for terminal open You specified a submode that is invalid when opening a terminal.
VALRNG	1192	Value range is %d to %d This is the valid range for the invalid value that you specified.
VALSPC	1191	Value specified is %ld You specified this value.
WRTFIL	1205	Cannot write to file The system write to a file failed.

## Fatal error messages

The following errors cause your program to abort immediately.

BADCMP	29	Compile not compatible with execution system The <b>.dbr</b> file was generated with an incompatible compiler or linker.
BADSYS	513	License management problem The runtime system has become invalid.
CMDBIG	515	Command line too long Your <b>dbr</b> command line exceeded the maximum limit of 2000 characters.
GBLNF	512	Cannot access named global %s You attempted to reference a global data section or external common that is not defined. This error occurs on entry to a routine that references the global data section or external common. It may also occur if variables defined as common in an external routine (which therefore default to external common) do not have a corresponding global common declaration within the main routine.
GBLSIZ	641	Reference exceeds allocated size of global data: %s The scope of the specified variable's defined data area exceeds the end of the containing psect (either common or global). This error only occurs when running in debug mode or if the environment variable BOUNDS_CHECK_FATAL is set.
INVFATERR	500	Invalid fatal error number for XCALL FATAL You requested an invalid error number on the FATAL subroutine.
INVOPT	516	Invalid option You specified an invalid option on the command line.
LMFAIL	514	Licensing failure During runtime start-up, some condition caused license validation to fail.
MSGNOTFND	400	Error message number %d not found or internal failure The specified error message could not be found.
NOCALL	2	Return with no CALL or XCALL A RETURN statement was executed without a corresponding CALL or XCALL statement.

## Error Messages

### Runtime Errors

NOTDBR	503	<code>%s</code> is not a DBR file You specified a non- <b>.dbr</b> file in the runtime command line.
OLDDBR	594	Old DBR file format <code>%s</code> detected: relink <code>%s</code> The DBR file format that you linked with is no longer supported. You must relink with a newer version. (We recommend linking with the current version.)
OPENF	509	Cannot open <code>%s</code> The specified file could not be opened.
RCBREL	903	RCB: Variable bound to local record on memory reclaim Routine: <code>%s%s</code> Local record memory is still set as an argument for <code>%RCB_CALL</code> when the routine and its memory are reclaimed. The checking for this condition only occurs when the routine has been compiled with <b>-qcheck</b> .
RUNERR	102	Internal runtime failure: <code>%s</code> This Synergy DBL system error is always accompanied by a qualifying error description. Such an error represents conditions in Synergy DBL that rely on the documented performance of operating system features. The error indicates that an operating environment is in an unexpected state. Please call us at 800.366.3472 or 916.635.7300 if you get this error, and make sure you mention not only the error mnemonic, but also the qualifying error description.
SIGNAL	508	Signal trap A fatal signal that required the Synergy DBL system to halt was caught.
SQLOBJ	901	SQL: Class variable still bound/defined on destruction You used <code>%SSC_BIND</code> , <code>%SSC_DEFINE</code> , or <code>%SSC_OPEN</code> using a variable defined in object instance data, and the cursor was not closed before the object destructor was called.
SQLREL	902	SQL: Variable bound/defined to local record on memory reclaim Routine: <code>%s</code> You used <code>%SSC_BIND</code> , <code>%SSC_DEFINE</code> , or <code>%SSC_OPEN</code> using a variable defined in a stack record, and the cursor was not closed before the segment was reclaimed. The checking for this condition only occurs when the routine has been compiled with <b>-qcheck</b> .

STKOVN	506	Runtime stack overflow	<p>The internal control stack for the runtime has overflowed. This generally occurs as the result of extremely deep subexpression nesting in arithmetic expressions and/or use of large stack records which cause the stack to be used up quicker. Use of stack records on their own cannot cause this problem. Recode the routine to avoid the problem. The default stack size is 256K.</p> <p>If you also get the informational error “System Stack exhausted - recursive call,” more than 1500 levels (or the maximum for the system stack if it is less) of method, subroutine, or function calls have occurred. You can increase this limit using the MAXRECURSELEVEL environment variable if the system stack allows.</p> <p>In Synergy .NET, all records (including those in main routines) default to STACK records for performance reasons. This means that large data divisions (for example, 8000a2000) can cause stack overflow errors. Use STATIC if appropriate, or change to use dynamic memory.</p>
UNSUP	507	Unsupported command	<p>You attempted to execute a Synergy DBL statement, a system-supplied subroutine or function, or an option to a system-supplied subroutine or function that is not supported on this operating system.</p>
VMSERROR	67	Unexpected VMS system error	<p>An OpenVMS system error occurred.</p>
VMSRMS	128	Unexpected VMS or RMS error	<p>An OpenVMS or RMS error occurred.</p>
WRTErr	520	write failure	<p>An unexpected failure occurred during a WRITE to an ISAM file.</p>

## Warning message

The following message is just a warning.

ELBREF	44	Undefined global data reference ‘%s’	<p>A global reference from within an ELB could not be found. Add a module that defines the global.</p>
--------	----	--------------------------------------	--

## Success message

The following message indicates that your program completed successfully.

STPMMSG	510	STOP
---------	-----	------

The program completed normally.

## Debugging log messages

The following error messages may be written to the **rd.log** file when the debugger is running remotely (i.e., in a client/server configuration).

DBGNOSOCK	610	Unable to attach to remote debug port
-----------	-----	---------------------------------------

The port specified for debugging was already in use when xfServerPlus attempted to launch the runtime or when the runtime attempted to listen on that port. The xfServerPlus session or program execution continues as if debug were not enabled.

DBGNOCONN	611	No debug client connection was established
-----------	-----	--

A debug client did not connect to the port within the specified timeout period. No debug client connection was established, and program execution continues as if debug were not enabled.

DBGSOCKER	612	Remote debug socket error; continuing without debug
-----------	-----	---

xfServerPlus was able to launch the runtime and the runtime was able to listen on the specified debug port, but some other error occurred when attempting to accept a connection. More detailed information on the error that occurred is appended to the log entry. The xfServerPlus session or program execution continues as if debug were not enabled.

DBGCLOSED	613	Remote debug client closed the connection; continuing without debug
-----------	-----	---

The debug client closed the connection. All breakpoints and watchpoints have been cancelled, and program execution continues as if debug were not enabled.

## Window error messages

Window error messages are displayed as part of an ERTEXT informational error. They are always preceded by a “Window Manager error” (WNDERR). The following is a list of window error numbers and their accompanying error text, as well as a description of what may have caused the error. (The character *n* in these messages represents a variable number.)

- |    |  |
|----|--|
| 2  | The window number <i>n</i> is invalid<br><br>The specified window ID is outside the range defined in the MAXWINS constant, or the window is not active.  |
| 3  | Window doesn't fit on screen<br><br>You tried to place a window whose display area won't fit on the screen.  |
| 4  | Not enough memory (needed <i>n</i> , had <i>n</i> )<br><br>The amount of memory you specified in the POOLSIZE constant wasn't sufficient for window processing.  |
| 5  | Value <i>n</i> not in range <i>n</i> to <i>n</i><br><br>The function requires a decimal argument, and the argument you specified was out of the possible range. For example, this error is generated if you specify a palette number of 17, because only palette numbers 1 through 16 are valid. |
| 6  | Function <i>n</i> not in range <i>n</i> to <i>n</i><br><br>You used an invalid function in an XCALL statement. For example, this error is generated if you try to use a WD_POS function in the W_PROC subroutine.  |
| 7  | Not enough arguments<br><br>You didn't supply all the arguments required for the XCALL statement.  |
| 8  | Wrong data type<br><br>An XCALL argument list contains an alpha argument where a decimal argument is required, or vice versa.  |
| 9  | Window name already defined<br><br>A WP_CREATE function uses a name that has already been used for another window.   |
| 10 | Too many windows (max of [maxwins])<br><br>You tried to create a window, but the maximum number of windows has been created.   |
| 11 | Input field not completely visible<br><br>You tried to obtain input from a portion of a window that is partially or completely occluded by another window.   |

## Error Messages

### Runtime Errors

- 12      No field set currently defined  
You called the W\_FLDS subroutine, and no field set has been created yet with the W7CREATE function.
- 13      Transfer area too small  
During a WI\_XFR transfer function, the destination field wasn't large enough for the data being transferred.
- 14      No user data set currently defined.  
A user data set operation (for example, WF\_UGET or WF\_UPUT) was attempted on a window that doesn't have a user data set.
- 15      One of the following error messages:
- ▶ Drag bar border necessary for SYSMENU  
The program attempted to create a system menu on a window that does not have a frame that can support one. A system menu is created when you associate a close method with a window. If the window frame doesn't have a drag bar, the system menu cannot be created. By default, Synergy windows usually have a drag bar, except in the following situations:
    - ▶ The “.BORDER off” or “.BORDER DRAGBAR(off)” script commands are used.
    - ▶ The W\_BRDR(id, WB\_OFF) or W\_BRDR(id, WB\_DRAGOFF) commands are used at runtime.
    - ▶ The window is too large to display a border inside the application window.
    - ▶ The window is only one row high and the drag bar has not been explicitly turned on.
  - ▶ Restoring Synergy Window: uncompression failure  
A W\_RESTORE operation failed (on any platform) because the format of the compressed data was corrupt. This error is most likely to occur in an I\_LDINP, M\_LDCOL, or U\_LDWND routine in the Toolkit, but it can also occur if you call W\_SAVE/W\_RESTORE directly. If this error occurs, something external to the window system has interfered with the window data (for example, perhaps an external application modified the data stored in a window library).
  - ▶ Window *n* not placed  
The ID of an unplaced window was passed to the WIF\_UNDER subfunction of %W\_INFO.



17      Toolbar error [%s]

One of the following toolbar errors has occurred:

- ▶    Button %s already loaded  
     A TBB\_LOAD was attempted using a button name that already exists on this toolbar.
- ▶    Group button %s is not loaded  
     A TBB\_GROUP was attempted using a button name that has not been loaded on this toolbar.
- ▶    Button %s not loaded  
     A TBB\_STATE or TBB\_SELECT was attempted using a button name that has not been loaded on this toolbar.
- ▶    Invalid toolbar ID: %d  
     The ID of a toolbar passed to TB\_BUTTON or TB\_TOOLBAR (for other than TB\_CREATE) is not the ID of any known toolbar.
- ▶    Function name too long, MAX 30  
     A function name passed as the method argument on a TBB\_LOAD is longer than 30 characters.

# Compiler Errors

## Fatal, nonfatal, and warning error messages

The messages below can be reported as nonfatal errors (E-), fatal errors (F-), or warnings (W-). For example, ABSIMP can be invoked as DBL-E-ABSIMP, DBL-F-ABSIMP, or DBL-W-ABSIMP. A compiler error that's reported as a warning by the **dbl** compiler with the **-qnet** option will be reported as an error in the Synergy .NET compiler.

If an error is reported, you must fix it before you can proceed. Some errors may abort the compiler before compilation is complete, and others will not, but either way, no output file will be created.

If a warning is encountered, the compiler still creates an output file. However, we recommend that you investigate the cause and fix the problem. On OpenVMS warnings cause runtime warnings, so all warnings *must* be fixed on OpenVMS.

Note that the **-W** compiler option (**/WARNINGS** on OpenVMS) enables you to control which warning levels will be displayed. (See [“Warnings” on page 1-17](#) in the “Compiler Options” table for details.) You can disable individual warnings with the **-WD** option. Warnings can also be forced to errors with the **-qerrwarn** option.

ABSIMP	766	ABSTRACT or EXTERNAL routine cannot have statements One of two problems has occurred:
		<ul style="list-style-type: none"> <li>▶ You declared the ABSTRACT modifier on a method and also provided an implementation for that method.</li> <li>▶ You declared a local external function and also provided an implementation for that function.</li> </ul>
ABSRTN	765	ABSTRACT %s must be in an ABSTRACT class You declared the ABSTRACT modifier for a method but did not declare the ABSTRACT modifier for its containing class.
ABSSEAL	772	Method cannot be ABSTRACT or VIRTUAL in a SEALED class  A method in a class defined with the SEALED modifier has been marked ABSTRACT or VIRTUAL. These modifiers cannot be specified in the same method definition.
ABSTINST	756	Cannot instantiate abstract class %s You've attempted to create an instance of a class that was defined with the ABSTRACT modifier, which means it cannot be instantiated.

ACCESS	669	%s is inaccessible	<p>The member you have tried to access is not accessible. If the specified member has PROTECTED access, access is limited to the containing class or types derived from the containing class. If it has PRIVATE access, access is limited to the containing type. Accessibility is determined as follows:</p> <ul style="list-style-type: none"> <li>▶ Inside an enclosing type, the accessibility of a member is determined by evaluating the accessibility of that member, regardless of the accessibility of the enclosing type. (For example, when using a field within the class in which it was declared, only the accessibility of the field, not the accessibility of the class, is evaluated.)</li> <li>▶ Outside an enclosing type, the accessibility of a member is determined by first evaluating the accessibility of the enclosing type and then evaluating the accessibility of the member. (For example, if a field has public accessibility and its class has private accessibility, the accessibility of the field is public inside the class and private outside the class.)</li> <li>▶ The accessibility of an inherited type member is based on the accessibility of that member as determined by the instance variable used to access it.</li> </ul>
ACCIGNORED	910	Accessibility on %s %s ignored	<p>An access modifier (PUBLIC, PROTECTED, or PRIVATE) was used on routine data. The modifier was ignored. (Level 3)</p>
ALGFLD	945	Alignment performed on %s	<p>This message is a warning to let you know that the specified fields have been aligned. (Level 3)</p>
ALIGN	364	Invalid .ALIGN value (%d)	<p>The boundary position that you specified for the .ALIGN compiler directive is invalid. The boundary must be one of the following values: BYTE, WORD, LONG, QUAD, PAGE, or an expression in the range of 0 through 9. This error is only generated by the <b>dbl8</b> compiler.</p>
ALPHARG	519	Alpha argument required	<p>The compiler requires an alpha argument.</p>
ALPHAXP	416	Alpha expression expected	<p>The compiler requires an alpha expression.</p>

ALPHDIM	732	Alpha not allowed for dimension specification You have attempted to pass an alpha type as a dimension value for an array.
ALPHXP	402	Alpha operands required (%s) The compiler requires an alpha expression.
ALRINPR	723	Already in a .NOPROTO section The .NOPROTO compiler directive was specified with a previous .NOPROTO directive already pending.
AMBIGUOUS	419	Path specification is ambiguous (...{ %s}) You have specified two or more variables with the same name, without specifying unambiguous paths for those variables. When you reference a non-unique variable, you must specify a unique path for that variable.
AMBOPT	104	Ambiguous option %s A compiler command line qualifier was too short to differentiate it from other similar qualifiers.
AMBSYM	668	Ambiguous symbol %s One of the following scenarios has occurred: <ul style="list-style-type: none"><li>▶ A routine call cannot be resolved unambiguously to a routine declaration.</li><li>▶ The specified locally defined or class identifier is not unique within a scope.</li><li>▶ The specified class identifier is not unique within multiple imported namespaces.</li><li>▶ The specified symbol path is not unique within the routine.</li><li>▶ A property and a field have the same name with different uppercasing, but the calling path doesn't exactly match the case of either member.</li><li>▶ A method call involving a parameterized type cannot be resolved to a method with specific non-generic types.</li><li>▶ When calling an extension method, more than one extension method was found in the same namespace.</li></ul> To make a class identifier unique, you can qualify it with its namespace. For example, a class called File could be identified as UserNS.Class1.File.

ARGMSMCH	791	Parameter must be declared with the MISMATCH modifier You attempted to use an inappropriate ^ARG function to access a parameter value without specifying the MISMATCH modifier for that parameter. For example, the MISMATCH modifier enables you to use ^ARGA on a <b>d</b> value passed to an <b>n</b> parameter. (Level 1)
ARGNUM	329	Invalid number of arguments You have specified an invalid number of arguments.
ARGPASS	837	Parameter passing convention does not match method declaration for argument %s in routine %s The method you called passes an argument with ^VAL or ^REF syntax, but the parameter was not declared as either in the method declaration,
ARGREQ	536	Field not an argument { %s } The argument passed to ^ARGNUM is not a valid parameter.
ARGTYP	341	%s type argument expected (%s) The compiler expected the specified data type, but the argument's data type was different.
ARRAYBOUNDS	680	Array dimension out of bounds for { %s } You attempted to access an array using an integer compile-time expression that is not greater than 0 or that is greater than the upper bound of the array (with bounds checking on).
ARRLCLSTR	848	Cannot use a local structure in a dynamic array You attempted to reference a structure that was declared inside a method, subroutine, function, or main routine in a dynamic array.
ASMLOAD	873	Unable to load assembly %s: %s Either the name of an existing .NET assembly was not provided when compiling with the <b>-ref</b> option, or an assembly that was not signed was referenced when an assembly was built with a specified keyfile.
ASNCERR	991	Async error: %s One of the following problems occurred: <ul style="list-style-type: none"> <li>▶ A hoisting error occurred within the body of an ASYNC method.</li> <li>▶ The method returned an invalid return type.</li> <li>▶ The version of the .NET Framework being used is lower than 4.5. ASYNC and AWAIT support is only available when compiled and run against Framework 4.5 or higher.</li> </ul>

An ASYNC method cannot contain any of the following:

- ▶ A field that is a member of group or a named record
- ▶ A field that overlays or is overlaid by another field
- ▶ A parameter that is BYREF, OUT, INOUT, or a parameter group
- ▶ A return type other than void, Task, or Task<>

ASNWR	942	Assignment within write not allowed for %s The first parameter of an assignment is a string. This is not permitted.
ATOS	979	Passing alpha to structure %s If you get the message “Passing alpha to structure causes conversion to structure,” you passed an alpha field to a structure parameter, which causes the alpha to be converted to a structure. If you get the message “Passing alpha to structure requires alpha of at least structure size,” the passed alpha field is smaller than the structure size, but it must be the same size or larger than the structure. (This is an error in Synergy .NET but a warning in traditional Synergy.) To avoid these errors/warnings, convert the alpha to a structure before passing the structure. (Level 3)
ATRUNC	794	Alpha expression too long for %s. Truncation may occur Truncation may occur when a source that is an alpha expression is moved to a smaller destination. (Level 4)
ATTRERR	970	%s error: %s An error occurred on the specified attribute. For example, if the error text says “ParamArrayAttribute error: parameter not a dynamic array,” the error occurred because the parameter type must be a dynamic array. If it says “Missing device license attribute,” you compiled with the <b>-device</b> option but your application doesn’t contain the SynergyDeviceLicenseAttribute attribute.
ATTRTARG	869	Attribute not valid on this target The attribute is not valid for the item that it is used with. For example, you would get this error if xfMethod or xfParameter were used on something that is not a method or parameter.

AWTERR	989	Await error: %s One of the following problems occurred: <ul style="list-style-type: none"><li>▶ The version of the .NET Framework being used is lower than 4.5. ASYNC and AWAIT support is only available when compiled and run against Framework 4.5 or higher.</li><li>▶ The path to which AWAIT is applied is not Task or Task&lt;0&gt;, and no GetAwaiter method was found within the expression's class or interface.</li><li>▶ AWAIT is being used in a method that's not declared as ASYNC.</li></ul>
BADACCESS	759	Inconsistent accessibility One of the following accessibility rules has been violated: <ul style="list-style-type: none"><li>▶ The accessibility of the parent class (PUBLIC, PROTECTED, or PRIVATE) must be the same as, or greater than, the accessibility of the child class.</li><li>▶ The types used in the signature of any member must be as accessible as the member itself. For example, an argument type or return type cannot have private accessibility in a publicly accessible method.</li><li>▶ Accessibility cannot be changed on overridden virtual members. The accessibility of an overridden method or property must be the same as the virtual member it overrides in the parent class.</li></ul>
BADCDIR	197	Invalid compiler directive: %s The specified compiler directive was not recognized by Synergy DBL. Double check your spelling of the compiler directive.
BADCNVCLS	780	Cannot convert %s class within class hierarchy The argument type and return type of the conversion operator must not be within the same class hierarchy
BADCNVTYP	781	Conversion operator must have a return or parameter type that is the enclosing class The conversion operator method must have either the argument type or the return type be the enclosing class type.
BADCONSTLOC	855	CONST field not allowed in non-CONST record A CONST field was specified in a record that was not declared with the CONST modifier.

## Error Messages

### Compiler Errors

BADCPATH	847	Complex path not supported  Certain complex paths for resolution are not supported. (A path is a list of IDs that may have array access, index access, method calls, and object casts.) You may need to break up the path and simplify it.
BADDSTRCTR	703	Destructor %s does not match class %s  A destructor method must have the same name as the class name but with a tilde (~) character at the beginning. The specified destructor name does not match the class name.
BADFLDPOS	681	Bad field position for %s  The field offset is invalid.
BADGENTYP	974	Unconstructed generic type not allowed in %s  A non-CLS structure or record cannot contain an unconstructed generic type (e.g., MyGenericClass<T>), only a constructed generic type (e.g., MyGenericClass<int>).
BADOPDECL	710	Cannot declare operator %s  The following .NET-reserved operator methods are not supported: op_Implicit, op_False, op_AddressOf, op_PointerDereference, op_SignedRightShift, op_UnsignedRightShift, op_UnsignedRightShiftAssignment, op_MemberSelection, op_Modulus, op_PointerToMemberSelection, op_LeftShift, op_RightShift, op_Comma, op_RightShiftAssignment, op_LeftShiftAssignment, op_ModulusAssignment, op_Assign, op_MultiplicationAssignment, op_SubtractionAssignment, op_AdditionAssignment, op_BitwiseAndAssignment, op_BitwiseOrAssignment, op_DivisionAssignment
BADPROP	785	Improper use of property %s  A property was used as the destination in a clear equate statement (for example, myproperty = <i>destination</i> ).
BADRNG	373	Invalid range specified  The lower-range entry was greater than the upper-range entry on a USING...RANGE statement.
BADROLOC	856	READONLY field not allowed in non-READONLY record  A READONLY field was specified in a record that was not declared with the READONLY modifier.



BADSLD	769	SEALED modifier can only be used on %s marked as OVERRIDE  You specified the SEALED modifier on a method or property that was not declared as OVERRIDE. The SEALED modifier can only be specified in conjunction with the OVERRIDE modifier.
BADSTATIC	778	Non-static reference not allowed in this context  You cannot use a non-STATIC reference in an initializer.
BADSTATLOC	779	Static field has improper location  You declared a field as STATIC in a common, literal, structure, or nonstatic record.
BADSYS	520	License management problem - %s  You have encountered a licensing problem. Make sure your version of Synergy DBL is installed and fully licensed.
BCSEAL	707	Cannot extend a sealed class  The class you tried to extend was defined with the SEALED modifier, which means it cannot be extended.
BDIGXP	145	Binary digits expected in argument (%s)  You must specify an argument that only contains binary digits (0 or 1). (Synergy .NET only)
BDINITVAL	506	Too many initial values  You have specified more initial values for a variable than there are instances of that variable.
BDQUOT	25	Unmatched quotes  You have specified an opening or closing quotation mark without its matching closing or opening quotation mark.
BIGALPHA	14	Alpha temporary result exceeds 65535  The results of an alpha string concatenation are greater than 65535 bytes on a 32-bit system.
BIGIDEN	316	Identifier too long: %s  You've specified an identifier that is longer than 30 characters in traditional Synergy on Windows and UNIX, 31 characters on OpenVMS, or 511 characters in Synergy .NET. Identifiers are only significant up to the 30th, 31st, or 511th character, respectively. (Level 2)

## Error Messages

### Compiler Errors

BIGNUM	15	Arithmetic operand exceeds maximum size The number is larger than allowed. For example, this error is generated if you provide an <b>a11</b> or larger record ID on a WRITE to a relative file.
BIGPATH	323	Path name too long The file path name specified in the .INCLUDE compiler directive was too long.
BIGSIZ	886	Size of %s %s too large The size of the specified arrayed field exceeds 256 MB. Reduce the field size to less than 256 MB.
BLTID	863	Potential name conflict with built-in type A class or structure has the same name as a built-in type (for example, <b>a</b> , <b>a30</b> , <b>i4</b> , etc.). (Level 4)
BMRET	943	%s cannot be used as method return type You've specified an invalid return type for a method. For example, ^VAL is not allowed as a return type for a method.
BNDXP	32	Bind name expected You've specified the bind compiler option without specifying the name of a routine to bind.
BOXLCLSTR	846	Boxing or unboxing of local structures not allowed You've attempted to cast the value type of a structure whose declaration was local to a method to or from System.Object.
BOXSTROBJ	864	Cannot box structure %s that contains a handle You've attempted to box a structure that contains an object handle.
BOXSTRSTRCT	994	Cannot save a string to a boxed non-CLS structure You've attempted to box a string as a non-CLS structure field.
BSTMTCH	715	Best match for %s has an argument type or quantity that doesn't match The compiler was unable to successfully resolve a method call; the closest one differed from the call by either the number of parameters passed or the type of one or more parameters. The signature of the closest match is displayed to aid in determining which parameters were incorrect.

CHGCASE	958	Case changed on override to match %s  An override method or property's identifier case didn't match the base method or property it's attempting to override (which is not allowed in .NET), so the case for the override method was changed to match the base method. In traditional Synergy, you will only get this error when using the <b>-qnet</b> (or <b>/NET</b> ) compiler option. (Level 3)
CIRC	1012	Circular %s not allowed  You've declared a circular interface declaration. For example,  <pre>interface iface1 extends iface2 interface iface2 extends iface1</pre>
CIRCBASE	728	Circular base class not allowed  When defining a class, you have declared a parent class that either forms an inheritance loop (for example, class A inherits from B and class B inherits from A) or is the same as the class being defined.
CIRCCONS	924	Circular constraint dependency  The constraint that you declared forms a circular dependency with the other type parameters in the declaration.
CIRCINIT	782	Circular initializer not allowed  The initializer that you specified has created a circular reference. The following is an example of a circular initializer:  <pre>method mychild     p1, i4     this(p1) proc endmethod</pre>
CLOSIN	37	Cannot close: %s  The specified source code output file could not be closed.
CLSREFRQD	718	Class type required: %s  You must specify a class name.
CLSYN	317	Invalid command line syntax: %s  The specified list option, which controls program listings and overrides compilation flags set by the .START, .LIST, or .NOLIST compiler directives, is invalid. Valid list options are +/NO/LIST, +/NO/COND, +/NO/SUMMARY, and +/NO/OFFSET.

COLLECTTYP	909	Incompatible collection type %s in FOREACH The type specified in a FOREACH statement is not valid for collection variables. A collection variable must be one of the following types: <ul style="list-style-type: none"><li>▶ A dynamic system array of rank 1</li><li>▶ A real Synergy array of rank 1</li><li>▶ System.Collections.ArrayList or a descendant</li><li>▶ Synergex.SynergyDE.Collections.ArrayList or a descendant</li></ul>
CONSARGS	702	Constructor requires argument list You must use the syntax below to instantiate an accessible class and pass parameter values to parameterized constructors: <i>VariableName</i> = new <i>FullClassName</i> ([ <i>param_values</i> ]) Note that the parentheses are required, even if you don't specify any <i>param_values</i> .
CONSBOTH	932	Cannot have both %s and %s constraints You cannot have both a structure constraint and a constructor constraint, or a structure constraint and a class constraint, on a type parameter.
CONSDUP	928	Duplicate constraint %s You cannot repeat an interface name in a constraint on a type parameter.
CONSERR	1011	Type constraint error: %s The specified type is not an interface, a non-sealed class, or a type parameter. The type on a type constraint must be one of those types.
CONSONE	929	Cannot have more than one %s constraint You declared more than one class, structure, or constructor constraint on a type parameter.
CONSREQD	933	%s constraint required You tried to create an instance of the type without declaring a constructor restraint on the type parameter.
CONSSEAL	927	Cannot use sealed class %s in constraint You declared a sealed class in a class constraint.
CONSSPC	926	Cannot use special class %s in constraint You declared one of the following special classes in a class constraint: System.Array, System.Delegate, System.Enum, System.ValueType.

CUSTATTR	872	Cannot create custom attribute You have declared a class with a base class of Attribute, thereby attempting to create a customer attribute class, which is not permitted in traditional Synergy.
DDAPIERR	530	Internal DDapi error %s %s An error occurred while reading the repository. There are two main reasons for this error: <ul style="list-style-type: none"><li>▶ The repository files are no longer available (for example, if they are on a network drive and the network goes down).</li><li>▶ The repository is invalid, and you need to run the Verify Repository utility on the files.</li></ul> This error is only generated by the <b>dbl8</b> compiler.
DDAPIERR2	752	Internal DDapi error %s %s An error occurred while reading the repository. The main reasons for this error are as follows: <ul style="list-style-type: none"><li>▶ The repository files are no longer available (for example, if they are on a network drive and the network goes down).</li><li>▶ The repository is invalid, and you need to run the Verify Repository utility on the files.</li><li>▶ A field in the repository is a Boolean field with a size other than 4. You must update your repository to change the size of the field to 4.</li></ul>
DDBADLOG	528	%s is not defined The specified logical name is not defined. This error is only generated by the <b>dbl8</b> compiler.
DDBADLOG2	749	%s is not defined The specified logical name is not defined. One possible reason is that the logical specified for the repository location in a .INCLUDE statement is lowercase. For example, on OpenVMS, since DCL uppercases the command line by default, a lowercase logical in your code will not match it. A similar problem can occur on UNIX. We recommend that you uppercase logicals that define a repository location in your .INCLUDE statements. If not, you can work around the problem by adding a second define in which the logical is lowercase and quoted (which preserves the case).

DDBADVERS	547	Incompatible repository version You are attempting to compile with an unsupported version of the repository files.
DDENUM	551	Enumeration %s not found in repository %s The enumeration specified in your .INCLUDE directive is not found in the specified repository.
DDINVLOG	535	Invalid specification for repository files: “%s” The specified repository filename is not valid. This error is only generated by the <b>dbl8</b> compiler.
DDINVLOG2	753	Invalid specification for repository files: “%s” The specified repository filename is not valid.
DDNOLOG	527	Logical for repository name expected Your repository filename must include a logical if the DBLDICTIONARY or RPSMFIL and RPSTFIL environment variables are not set. This error is only generated by the <b>dbl8</b> compiler.
DDNOLOG2	750	Logical for repository name expected Your repository filename must include a logical if the DBLDICTIONARY or RPSMFIL and RPSTFIL environment variables are not set.
DDOPN	525	Cannot open repository’s main file: %s You’ve .INCLUDEd from the repository, and the compiler cannot find the repository’s main file. This error is only generated by the <b>dbl8</b> compiler.
DDOPN2	751	Cannot open repository’s main file: %s You’ve .INCLUDEd from the repository, and the compiler cannot find the repository’s main file.
DDRD	523	Cannot read repository’s main file The compiler cannot read the repository’s main file (usually called <b>rpsmain.ism</b> ).
DDSTRUCT	549	Structure %s not found in repository %s A structure or class is not found in the repository.
DDTXTOPN	526	Cannot open repository’s text file: %s You’ve .INCLUDEd from the repository, and the compiler cannot find the repository’s text file.

DDTXTRD	524	Cannot read repository's text file The compiler cannot read the Repository's text file (usually called <b>rpstext.ism</b> ).
DECLORDER	906	Cannot use %s before declaring it You've attempted to use the specified enumeration value as an initial value before the enumeration was declared with the ENUM statement.
DECXP	407	Decimal or integer operand required (%s) An invalid data type has been used in the procedure division.
DEFSIZE	720	Default size may not match %s definition A field within a global or external literal or common was specified without an explicit size (for example, <b>i*</b> or <b>d</b> where the size is determined from the size of the initial value). Access may be incorrect if the size differs from the global definition. We recommend that you explicitly specify the size of the field. (Level 4)
DELAYSGN	948	Delay signing was requested, but no keyfile was given If the <b>-delaysign</b> compiler option is specified, you must also specify the <b>-keyfile</b> option with file containing the cryptographic key. (Level 3)
DELEVT	892	Delegate type expected for event %s The type for a delegate must be handle.
DESCTYPRQD	935	%s is not a descriptor type You tried to access a non-descriptor type using ^ARG, ^ARGA, ^ARGN, ^ARGDIM, ^ARGTYPE, or ^ARGPRECISION. In traditional Synergy, you will only get this error when using the <b>-qnet</b> (or <b>/NET</b> ) compiler option. (Level 1)
DEXPREQ	412	Decimal or integer expression required Either an invalid expression has been used in the data division or an alpha has been passed as a dimension value for an array.
DIMMISMCH	952	Parameter dimensions mismatch: %s A routine parameter that is specified with explicit dimension sizes is not the same dimension size as the argument passed in a call to the routine. (Level 4)
DIMNDIM	798	Dimensioned access of non-dimensioned item %s A bounds check occurs when an array is provided to a non-arrayed field. (Level 4)

## Error Messages

### Compiler Errors

DIVIDE	30	Attempt to divide by zero An arithmetic operation attempted to divide by zero. (Synergy .NET only)
DLLIMP	983	DllImport error: %s The DllImport attribute was added to a method for which one or more of the following is true: <ul style="list-style-type: none"><li>▶ It is not marked STATIC</li><li>▶ It is marked EXTERNAL</li><li>▶ It has a parameter type that's a descriptor, a non-CLS type, or a group</li></ul>
DUPACC	789	Property cannot have multiple get or set methods You've provided more than one GET or SET method for a property. A property can have a maximum of one GET and/or SET method.
DUPATTR	901	Duplicate %s attribute You've specified the same assembly attribute more than once.
DUPDRCTV	953	Duplicate directive More than one .ARRAY directive is present in the source file. Once .ARRAY is declared, it is valid until the end of the source file.
DUPMETH	704	Class %s already defines a method %s with the same parameter types The specified class contains two or more methods whose declarations differ only by return type.
DUPMODIF	764	Modifier %s can only be specified once The preprocessor detected duplicate modifiers.
DUPMRNK	957	Class %s already defines a method %s whose signature only differs by rank Synergy .NET cannot support overloads whose parameters differ only by array rank. For example, you cannot define <b>meth1(parm1, [1]a)</b> and <b>meth1(parm1, [1,1]a)</b> in the same class. You must remove one of the overloads. (Level 1)



DUPOPTS	914	Class %s already defines a method %s that differs only by optional parameters  Either the overloading method is the same as method being overloaded except for optional parameters, or more than one method overload per ID has trailing optional parameters (including parameters defined explicitly and methods marked with VARARGS).
DUPQUAL	687	Qualifier %s can only be specified once  A preprocessor directive contains duplicate qualifiers.
DUPRMV	875	Duplicate %s in class %s has been removed  More than one method with same signature existed in a class. The duplicate method was removed. (Level 4)
DUPRTN	967	Duplicate routine %s  Two subroutines or functions with the same name exist globally or in a namespace.
DUPSYP	664	Duplicate symbol %s for same parent not allowed  A local data field has the same name as a record, group, or field in an enclosing scope.
EFDEF	509	External function already declared  You've declared the same external function more than once.
EFNFND	907	Call to external function %s cannot be resolved  For any function declared in the EXTERNAL FUNCTION section, a unique routine must exist that matches the external function's name and return type. If none is found, this error is generated.
EMPTYA	724	Invalid usage of empty alpha string  A USING statement contained an empty alpha variable.
EMPTYUCSTMT	731	Empty USING or CASE statement  The USING or CASE block does not contain any statements to execute. (Level 4)
ENDGBLEXP	542	ENDGLOBAL expected  A GLOBAL DATA SECTION has no ENDGLOBAL statement.
ENDGRPEXP	504	ENDGROUP expected  A GROUP statement in the data division has no ENDMETHOD statement.

## Error Messages

### Compiler Errors

ENDXP	337	END statement expected Your code contains a BEGIN statement without a matching END statement. This error is only generated by the <b>db18</b> compiler.
EOLXP	408	End of line expected at { <i>%s</i> } Some extraneous text follows a statement at the end of the line.
ERRCNT	303	Too many errors Too many compilation errors have occurred. You may want to change the setting of the DBLMAXERR environment variable. (Level 1)
EVTCLS	895	Event <i>%s</i> must be raised from within <i>%s</i> The specified event must be raised from within the class in which it is declared.
EVTCUST	980	Custom event error: <i>%s</i> A custom event was written incorrectly. Both an add and a remove accessor must be declared, and a custom event cannot be raised.
EVTEXP	894	Event expected for <i>%s</i> The first parameter for the specified statement (ADDHANDLER, REMOVEHANDLER, or RAISEEVENT) was not an event. All three of these statements require an event as their first parameter.
EVTHDLR	893	Event handler <i>%s</i> does not match event signature The passed-in event handler is either a method whose signature does not match the signature of the event delegate or a nonmethod whose type is not the same as the event's delegate.
EVTIMP	988	Event interface implementation error: <i>%s</i> An event was implemented as both a WinRT event and a .NET event. Only one or the other is allowed.
EXACTPATH	425	Exact specification of ambiguous path used (...{ <i>%s</i> }) You've specified a path name that is not unique. Ambiguous path names are tolerated for VAX DIBOL compatibility; however, we recommend that your path names be unique. (Level 1)
EXPHVAL	790	^VAL function expected The Undefined functions compiler option ( <b>-qimplicit_functions</b> ) resolves to a nonlocal function that's not ^VAL.

EXTMERR	<p>977        Extension method %s %s</p> <p>An error has occurred due to one of the following reasons:</p> <ul style="list-style-type: none"> <li>▶ The specified extension method has no parameters. It must have at least one.</li> <li>▶ The extension method you declared is non-static. It must be static.</li> <li>▶ You declared an extension method in a class that is either generic or non-static (or both).</li> <li>▶ The extension method's first parameter type is a group or a non-CLS structure.</li> </ul>
FENEWINDXR	<p>911        A class with an indexer marked NEW is not supported in FOREACH</p> <p>You've tried to use a descendant class of ArrayList or SynergyDE.ArrayList with an indexer property marked as NEW in a FOREACH statement. This is not allowed. If you change the indexer to OVERRIDE, it should work with FOREACH as desired. (Note that you will need to match the signature of ArrayList's indexer, which means the property type must be @*.)</p>
GBLDR	<p>1007       External common record name containing %s does not match global common record name %s</p> <p>A field or group in a named external common does not match the name of the global common record that contains that field. To eliminate the warning, change the external common record name to match the global common record name. (Synergy .NET only) (Level 3)</p>
GBLND	<p>939        Global declaration does not contain %s</p> <p>The name of the specified external common field was not found in a global common. (Synergy .NET only)</p>
GBLTNM	<p>938        Type or size for %s does not match global declaration</p> <p>An external common field differs from the global declaration in either type or size. In traditional Synergy, you will only get this error when using the <b>-qnet</b> (or <b>/NET</b>) compiler option. (Level 1)</p>
GBLTYP	<p>940        Type %s not allowed in common declaration</p> <p>You've used a remapped type (double, float, or decimal) in a global common. In traditional Synergy, you will only get this error when using the <b>-qnet</b> (or <b>/NET</b>) compiler option. (Level 1)</p>

## Error Messages

### Compiler Errors

GDSDEF	510	Global data section already defined You've referenced the same global data section more than once. Within any given routine, only one GLOBAL statement can reference a particular global data section.
GDSMSMCH	950	Global data section '%s' does not match initialized Global%s A global data section reference (no ,INIT) is different from the global data section with ,INIT. This can happen when a field name is missing, the data type is different, or the size does not match. Note that not all named records in the global data section need to be referenced.
GENSTRCLS	995	Generic structure type must be CLS A generic structure (a structure with type parameters in its definition) is not marked with the CLS modifier. A generic structure must be a .NET value type structure.
GLCMN	77	COMMON invalid in GLOBAL sections You've specified the COMMON statement in a global data section.
GLLCL	538	LOCAL RECORD invalid in GLOBAL sections You've specified the LOCAL RECORD statement in a global data section.
GLLIT	505	LITERAL invalid in GLOBAL sections You've specified the LITERAL statement in a global data section.
GLOBDUPIGN	1017	Duplicate global %s ignored You've compiled with the <b>-qrelaxed:allowdup</b> option, and a global structure name or enumeration was reused. (The first declarations are used and the succeeding ones ignored.) (Level 3)
GLSTC	500	STATIC RECORD invalid in GLOBAL sections You've specified the STATIC RECORD statement in a global data section.
GLSTK	539	STACK RECORD invalid in GLOBAL sections You've specified the STACK RECORD statement in a global data section.
GLSTR	540	STRUCTURE invalid in GLOBAL sections You've specified the STRUCTURE statement in a global data section.
GNCEXT	930	Generic class cannot extend %s You cannot extend the named item with a generic class.

GRPBIG	507	GROUP %s size too large You've specified a <i>length</i> argument in your group declaration, and the size of the fields and/or groups within that group exceed the specified length.
GRPETY	824	Group is empty The group does not contain any members. (Level 4)
HANDRQD	709	Handle to %s required You did not specify a handle when using a class type.
HDIGXP	146	Hexadecimal digits expected in argument (%s) You must specify an argument that only contains hexadecimal digits (0 – 9 and A – F). (Synergy .NET only)
HIDABS	862	%s hides an inherited abstract member A method is attempting to hide an abstract method (in other words, declare a member using the NEW modifier) with the same name in an abstract class that it is extending.
HIDEERR	788	%s hides method that is not VIRTUAL, OVERRIDE, or ABSTRACT in class %s. NEW required The specified class member hides an inherited member that is not declared as VIRTUAL, OVERRIDE, or ABSTRACT, but the NEW modifier has not been declared for the overriding member. You must specify NEW to hide an inherited member of the same name. (Level 1)
HIDEHAT	916	Routine hides data reference operation %s You have defined a routine whose name conflicts with a Synergy data reference operation. To call the system-supplied data reference operation, you must use the “^” syntax. Otherwise, your user-defined function will be called using the “%” syntax. (Level 4)
HIDEW2	1002	%s hides %s The first specified local data field or CATCH variable hides the second specified local data field or CATCH variable in the same routine. (Level 3 [or Level 4 for class members])
IDIGN	534	.IDENT ignored On OpenVMS, an object module can only have one .IDENT record. You've compiled multiple source modules into one object module using the compiler command syntax: DIBOL SOURCE1+SOURCE2+SOURCEn . . .

This warning tells you that only the first .IDENT compiler directive will be acted upon. This warning only occurs on OpenVMS. (Level 3)

IDXCLSARY	843	Cannot index real array %s A real array has an index.
IDXP	414	“%s” expected at or near { %s} The compiler found an unexpected item in the source line.
IDXRPARM	783	Indexer property must have at least one parameter The indexer does not have any arguments. You must provide at least one argument to the indexer.
IFCIMP	888	Interface %s not implemented by class % You’ve declared an explicit interface member, but the interface name is not included in the IMPLEMENTS list of the class definition.
IFSTMT	354	Statement part of IF missing You’ve specified an IF statement condition without the statement to be executed.
ILASM	900	ILASM error: %s When compiling, Synergy .NET code generated by the compiler caused an IL error. Please report such errors to Synergex, along with code that reproduces the problem.
IMPBOXCAST	963	Implicit argument boxing cast (@a/@d/@i); explicit cast recommended A numeric parameter (type <b>n</b> ) was passed to a method parameter of type System.Object, and that value was implicitly boxed, but the actual boxed value type may be unknown. We recommend explicitly casting the parameter to the expected type using typed boxing casts (@ <b>d</b> , @ <b>i</b> , @ <b>a</b> , etc.). An explicit cast suppresses the warning.
IMPDIRIGN	1029	Import directory clause no longer used, ignored You’ve specified a directory location (DIRECTORY ‘ <i>location</i> ’) on an IMPORT statement. As of version 10.3.3, the IMPORT statement no longer supports a DIRECTORY specification. Use the <b>-qimpdir</b> compiler option or the SYNIMPDIR environment variable instead. (Level 3)
IMPFLD	796	Position of field %s causes an implicit field to be added A nonoverlay field that doesn’t start at the end of the previous field has been added. (Level 4)

IMPORT	701	Namespace %s not found No prototypes are found in the import directory for the specified namespace.
IMPORTNS	797	Invalid import namespace %s The namespace specified in the IMPORT command isn't a valid identifier.
IMPOUTNS	803	IMPORT statement must be declared outside of a namespace An IMPORT statement is not allowed within a NAMESPACE-ENDNAMESPACE block.
IMPSTOP	777	Implicit STOP added to end of subroutine The code generator added a STOP statement where one was needed. This warning may be generated erroneously if we cannot detect that the routine has a valid exit (for example, RETURN, XRETURN, FRETURN, or MRETURN). (Level 4)
INCMODIF	761	The modifier %s is incompatible with %s The .INCLUDE modifier that you provided is invalid and can't be specified together with another modifier.
INDIMGRP	853	%XTRNL not allowed within a dimensioned group The ^XTRNL function was specified as an initial value for a field that is a member of an arrayed group.
INFINALLY	792	%s must be completely inside the FINALLY block A statement in a FINALLY block transfers control to a label outside the FINALLY block.
INITCONST	691	CONST/LITERAL field missing initialization value A field denoted as CONST or LITERAL doesn't have an initial value like it should. (Level 3 or 4)
INITEXP	719	Initial value expected, defaulting to %s The initial value on a field within a global literal was not specified. The initial value will default to a 0 or a space, as indicated. (Level 4)
INITIGN	721	Initial value on external field ignored A specified initial value on a field is ignored because it's within an external literal or external common. The initial value is only recognized on the global definition. (Level 4)

## Error Messages

### Compiler Errors

INITIGN2	1001	Parameter not marked as DEFAULT A parameter with an initial value was not defined with the DEFAULT modifier. (Level 3)
INITVALMIS	1020	Missing initialization value The initial value expressions is missing in a collection initializer.
INITVERR	982	Initial value error: %s You used ^M as an implicit data type. For example, the following is not allowed: <pre>data var = ^m(struct1,hnd)</pre> (Synergy .NET only)
INMTHD	568	MRETURN %s within a method You used the MRETURN statement in a subroutine, function, or main routine. MRETURN returns control from a method, and a method that has a declared, non-VOID return type must have at least one MRETURN statement.
INTCMPERR	685	Internal compiler error: %s An internal compiler error occurred.
INTCMPERR2	746	Internal compiler error: %s A parsing failure occurred, or the compiler encountered a structure it didn't expect.
INTERR	517	Internal error number: %d An internal problem with the specified internal error number has occurred in the compiler.
INTLEXERR	605	Internal lexer error: invalid state %s Due to erroneous code, the lexer/parser has entered into an invalid state.
INTLEXERR2	606	Internal lexer error: invalid token %s Due to erroneous code, the lexer/parser has encountered an invalid token.
INTRNG	799	Ranged access of integer fields not portable between machines of different endian types You are accessing a range on an integer field, which is not portable between machines of different endian types. (Level 4)



INVALIGN	725	Invalid .ALIGN value (%s)  The boundary position that you specified for the .ALIGN compiler directive is invalid. The boundary must be one of the following values: BYTE, WORD, LONG, QUAD, PAGE, or an expression in the range of 0 through 9. In Synergy .NET, objects must be aligned on a native boundary. Unless you specify <b>-platform=x86</b> when you compile, objects must be on an 8-byte boundary for 64-bit Windows.
INVARG	327	Invalid argument: %s  One of the following has occurred: <ul style="list-style-type: none"><li>▶ An invalid record name has been specified in a .INCLUDE from the S/DE Repository.</li><li>▶ A field name or a literal has been passed to ^PASSED.</li></ul>
INVARGOBJ	912	^ARGDIM not allowed on objects; use method to get length  The ^ARGDIM operation cannot be used on an object parameter. Since collection object types typically implement a method or property that returns the number of items in the collection, you can call the appropriate method or access the appropriate property to get the number of items in the object.
INVARGTYP	409	Invalid argument type  You've specified an invalid data type in an argument declared under the SUBROUTINE statement.
INVBGNHDR	828	BEGIN should be followed by end of line  Additional text appears on the same line as a BEGIN statement. The BEGIN keyword should be the only thing on the line.
INVCALL	754	Invalid calling convention  You have attempted to call a subroutine as a function or to call a method as a subroutine or function. (Level 1)
INVCAST	699	Invalid cast  Your attempt to cast an object variable to a specific class type using the parenthesis syntax was not successful. This type of cast must meet at least one of the following conditions: <ul style="list-style-type: none"><li>▶ The cast type falls within the hierarchy (either up or down) of the object being cast.</li><li>▶ An explicit conversion exists from the source to the destination type.</li><li>▶ A value type is being cast to or from System.Object.</li></ul>

## Error Messages

### Compiler Errors

INVCLASS	560	Invalid class: %s You did not pass a valid class as the second parameter of the .IS. operator.
INVCLIDEF	600	Invalid/obsolete command verb “%s” used to invoke compiler  On OpenVMS, the error “%CLI-E-ENTNF, specified entity not found in command tables” was generated. The compiler queries the system for each and every valid compiler switch to see if it is present on the command line and if so, how it is set. This error is generated if the system has no knowledge about the specified switch.
INVCLSENT	806	Invalid class member at or near {%s} The syntax for a class member is not valid.
INVCRECENT	807	Invalid class record entry at or near {%s} The syntax for a class record member is not valid.
INVDATADECL	845	Invalid %s declaration at or near {%s}  The syntax of the specified structure, record, common, literal, or external function declaration is not valid. For example, the declaration <code>public structure mystruct##</code> would cause the error “Invalid structure declaration at or near {##}.”
INVDATAENT	813	Invalid data division entry at or near {%s} The syntax for a data member is not valid.
INVDATALOC	842	Invalid DATA statement location The DATA statement must be the first statement inside a scope.
INVDATASTMT	834	Invalid data statement syntax The syntax for a DATA statement is not valid.
INVDECSIZ	418	Invalid data size specification {%s} You have declared a size on a parameter group, or an incorrect size was declared for a decimal or packed parameter. (Level 3)
INVDEFNS	727	Invalid default namespace %s The SYNDEFNS environment variable specifies a namespace that does not have the correct syntax.

INVDSTRCTR	740	Destructor cannot have %s You have declared a destructor with arguments, modifiers, and/or a return type. A destructor does not accept any arguments or modifiers and it cannot return a value.
INVENUMFLD	810	Invalid enum field at or near { %s } You have declared an enumeration that has invalid syntax in one or more fields.
INVEVT	896	Incorrect use of event %s Something other than an event was passed as the <i>event</i> argument to the ADDHANDLER, REMOVEHANDLER, or RAISEEVENT statement.
INVEXC	743	Invalid exception type The CATCH or the THROW exception variable was not the correct type. Both THROW and CATCH block exception variables must either be type System.Exception or inherit from System.Exception.
INVEXFTYP	417	Invalid external function data type The access type that you specify in an EXTERNAL FUNCTION declaration must be <b>a</b> , <b>d</b> , <b>d.</b> , <b>i</b> , <b>p</b> , <b>p.</b> , or <b>^VAL</b> .
INVEXPR	817	Invalid expression at or near { %s } The syntax for an expression is not valid.
INVEXTENT	815	Invalid external function entry at or near { %s } The syntax for an external function declaration is not valid.
INVFCALL	972	Function %s should be called using function call convention A function has been called without function call syntax (a leading “%” or parentheses). To avoid this warning, add a “%” or parentheses (or both) to the function call. Using function call syntax prevents other potential compile-time errors by unambiguously telling the compiler to look for a function, so it won’t accidentally resolve to things that aren’t functions. (Level 4)
INVFF	90	Invalid fixed field size An implied-decimal variable has an invalid size. The maximum size of the whole number part is 28 significant digits, and the maximum size of the fractional precision is also 28 digits (for example, <b>d28.28</b> ). The field size must be equal to or greater than the precision.

## Error Messages

### Compiler Errors

INVFNCD	426	Invalid function identifier { %s} You've specified an illegal function name. The function name must be an alphanumeric identifier.
INVFUNCNAME	695	.IF directive does not support function %s%s The specified function is no longer supported by the preprocessor.
INVGDSNT	814	Invalid global data section entry at or near { %s} The syntax for a global data section member is not valid.
INVGNSYNTX	922	Invalid generics syntax at or near %s The syntax you specified for a generic type is invalid. See " <a href="#">Generic Types</a> " in the "Understanding Objects and Classes" chapter of the <i>Synergy DBL Language Reference Manual</i> for valid generic syntax.
INVHFNC	737	Invalid ^ function ( %s) The function that you have attempted to call as a data reference operation is not a data reference operation and therefore cannot be called with a caret (^).
INVHMP	903	Partial structure cannot be used in ^m function You cannot use the PARTIAL modifier on a structure declaration in an ^M statement.
INVIMP	802	Invalid IMPORT declaration The import directory specified in the <b>-qimpdir</b> compiler option is not an existing directory.
INVINIT	763	Invalid initializer The initializer method signature must resolve to a constructor method in the parent class (if <b>parent</b> is declared) or in the current class (if <b>this</b> is declared). In addition, an initializer argument value must be a literal, a STATIC field value, a CONST field value, or a parameter name from the constructor.
INVINITMBR	999	Invalid initializer member Either the initializer member list contains a syntax error or you have attempted to mix collection and object initializers. (Nesting is fine, but mixing causes this error.)

INVINTFCENT	867	Invalid interface entry at or near { %s}  The interface name you specified contains invalid characters. See <a href="#">“Identifiers”</a> in the “Welcome to Synergy DBL” chapter of the <i>Synergy DBL Language Reference Manual</i> for more information about identifiers.
INVIOERRCODE	819	Invalid I/O error code at or near { %s}  The syntax for an I/O error code is not valid.
INVKSW	350	Invalid key switch: %s  The key switch that you specified on the SORT or MERGE statement to indicate the direction of the search is invalid. Valid switches are <b>f</b> , <b>r</b> , <b>a</b> , <b>d</b> , and <b>c</b> .
INVKWD	818	Invalid keyword at or near { %s}  Either the EXTENDS keyword is misspelled, or an invalid code follows a valid subroutine or function identifier without a comma.
INVMAINDECL	854	Invalid MAIN declaration at or near { %s}  The specified MAIN declaration syntax is not valid. See <a href="#">MAIN</a> in the “Synergy DBL Statements” chapter of the <i>Synergy DBL Language Reference Manual</i> for valid syntax.
INVMETHINIT	809	Invalid method initialization list at or near { %s}  The syntax for a method initializer is not valid.
INVMOD	739	Invalid modifier %s on %s  A constructor can only have the following modifiers: PUBLIC, PROTECTED, PRIVATE, or STATIC. All other modifiers are invalid. (Level 1)
INVMRET	569	Invalid MRETURN: %s  Either you did not specify a value for an MRETURN statement on a method that has a declared, non-VOID return type, or the MRETURN statement specifies a value for a method that is a constructor or a destructor or that has a VOID return type.
INVMTHDHDR	850	Invalid method header at or near { %s}  The method declaration is not valid, because it includes the specified invalid token. All subsequent tokens from the specified token to end of line are ignored or discarded.

## Error Messages

### Compiler Errors

INVNEW	771	NEW modifier not required and will be ignored You specified the NEW modifier on a member that does not hide an inherited member. (Level 1)
INVNIND	986	Invalid number of indices You tried to access an array whose rank was 2 or greater but didn't provide enough indices. In traditional Synergy, you will only get this warning when using the <b>-qnet</b> (or <b>/NET</b> ) compiler option. (Level 3)
INVNSENT	805	Invalid namespace entry at or near { <i>%s</i> } The syntax for a namespace member is not valid.
INVNUMDIM	421	Incorrect number of dimensions for { <i>%s</i> } You've referenced an array with an incorrect number of subscripts.
INVOBJREF	678	Object references not allowed in <i>%s</i> An object field was declared in a global data section.
INVOCB	99	Invalid OCB index: <i>%s</i> An internal problem has occurred in the compiler.
INVOHND	563	Invalid object handle: <i>%s</i> You did not pass an object handle as the first parameter of the .IS. operator.
INVOPT	103	Invalid option: <i>%s</i> One of the following has occurred: <ul style="list-style-type: none"><li>▶ You've specified an invalid option on an OPEN, READ, ISAMC, or other statement with optional qualifiers.</li><li>▶ You've specified an invalid option or an invalid number of options (for example, more than one <b>-out</b> option) on the command line. For information on available compiler options and the arguments they accept, see <a href="#">"Linking Object Modules" on page 1-37</a> (for traditional Synergy) or the <a href="#">"Synergy .NET Compiler Options"</a> topic in Synergy/DE WebDocs (for Synergy .NET).</li><li>▶ You're using a version of the Common Language Runtime that's lower than 4.</li></ul>
INVPARAMEN	811	Invalid parameter entry at or near { <i>%s</i> } The syntax for a parameter is not valid.

INVPARMGRP	849	Invalid parameter group declaration at or near {%s} The group declaration is not valid, because it includes the specified invalid token. For example, dimensions are not allowed in a group declaration, so brackets ( [ ] ) would cause this error to occur. All subsequent tokens from the specified token to end of line are ignored or discarded.
INVPASSED	762	^PASSED on required parameter is always true You have used ^PASSED on an argument marked as REQUIRED. Either mark the parameter as optional or remove the ^PASSED call altogether. (Level 3)
INVPASSED2	1021	^PASSED is always true for non-Synergy type You have used ^PASSED on an argument that is not a Synergy type. (Level 3)
INVPOS	102	Invalid OCB load position An internal problem has occurred in the compiler.
INVPRC	224	Invalid fractional precision During evaluation of an arithmetic expression, the number of digits in the fractional portion of an implied-decimal variable exceeded 28. (Synergy .NET only)
INVPROGENT	804	Invalid program entry at or near {%s} The syntax is not valid at the program level (namespace, subroutine, or function).
INVPROTOCXT	829	Prototypes must be imported using the IMPORT statement You attempted to include a prototype in a file using the .INCLUDE directive. A prototype file cannot be .INCLUDED; it must be imported using the IMPORT statement.
INVQUAL	688	%s qualifier is incompatible with %s An I/O statement or preprocessor directive contains an invalid qualifier.
INVRECFLD	812	Invalid record field at or near {%s} The syntax for a field in a record is not valid.
INVRERR	362	Invalid error literal You've specified an incorrect error literal in an I/O error list or ONERROR statement.

## Error Messages

### Compiler Errors

INVRQUAL	415	Invalid routine qualifier { %s } You've specified an invalid qualifier on the SUBROUTINE statement. Valid qualifiers are REENTRANT, RESIDENT (on Windows and UNIX), TRUNCATE, and ROUND.
INVRSW	353	Invalid record switch: %s The record switch that you specified on the SORT or MERGE statement to identify the record as fixed-length or variable-length is invalid. Valid switches are <b>f</b> and <b>v</b> .
INVRTHRW	744	Cannot rethrow outside of CATCH An attempt to rethrow the most recently caught exception occurred outside of a CATCH block.
INVSCRFC	360	Invalid screen function or parameter { %s } You've specified an unknown screen function command (\$SCR_) on the DISPLAY statement.
INVCTOR	883	Cannot declare %s on static constructor You've attempted to declare parameters on a static constructor.
INVSTMT	816	Invalid statement at or near { %s } The syntax for a statement is not valid.
INVSTMTBDY	821	Invalid %s body at or near { %s } The syntax for a statement body is not valid.
INVSTMTHDR	820	Invalid header in %s at or near { %s } The syntax for a statement header is not valid.
INVSTRUCENT	808	Invalid structure entry at or near { %s } The syntax for a structure member is not valid.
INVSTX	674	Invalid syntax A general parsing syntax error occurred.
INVTERM	822	Invalid termination of %s at or near { %s } The termination of an item was not successfully completed.



INVTYP	860	Invalid use of type %s One of the following occurred: <ul style="list-style-type: none"><li>▶ A structure or class name was used incorrectly. For example, a class name can't be used as a variable, a structure name can't be passed as an argument to a routine, etc.</li><li>▶ A non-CLS structure containing object type fields was returned from a method or as the type of a property. (Level 1)</li></ul>
INVTYPSIZ	413	Invalid data type/size specification {%s} A field or group's type or size specification is incorrect. For example, a group may not have a <b>p</b> (packed) data type; the size value cannot be 0 or less; and the precision cannot be less than 0, greater than the specified field size, or greater than 28. In addition, you cannot specify a fixed-size array of any class type (derived from System.Object). (Level 1)
INVUSNGENT	915	Invalid entry within using statement There is a syntax error within the body of a USING statement.
ISSTAT	717	%s required to access static member %s You must precede the method name with the class name when calling an accessible STATIC routine declared within another class from within a STATIC or non-STATIC routine. The same rule applies to CONST records.
ITOPBAD	877	Unable to do interop for %s While using the <b>-qrelaxed:interop</b> option, an unexpected value was passed to either of the parameters of the ADDHANDLER statement.
IVBAD	112	Initial value not allowed here Initial values are not allowed in overlay or STACK records.
IVLNG	115	Initial value too long The initial value specified was larger than a single element of the field.
IVXP	116	Initial value expected You must specify an initial value for an automatically sized variable (indicated with a *) to determine the variable's size.

LAMBDA	<p>978        Lambda error: %s</p> <p>Either a hoisting error or a code restriction occurred within the body of the lambda function. A lambda function cannot contain any of the following:</p> <ul style="list-style-type: none"><li>▶ A field that is a member of group or a named record</li><li>▶ A field that overlays or is overlaid by another field</li><li>▶ A parameter that is BYREF, OUT, INOUT, or a parameter group</li><li>▶ A CALL, RETURN, or ONERROR statement</li><li>▶ An EXIT or GOTO to a label outside of the lambda function</li><li>▶ A YIELD statement</li><li>▶ An I/O error list</li></ul> <p>This error also occurs if a data division can't be created in the generated method or if an inline lambda is used in the initial value for a non-local, non-class field.</p>
LBLDEF	<p>339        Label previously declared</p> <p>The current label name has already been used. A statement label must be unique within the routine.</p>
LBLSCOPE	<p>550        Label %s out of scope</p> <p>Code jumps to labels (for example, CALL, GOTO, I/O error list, EXIT lbl, READS(,eof), ACCEPT(,eof), GETS(,eof), PUTS(,eof), RECV(,nomsg), and WRITES(,full)) from any of the following to a higher scope are not allowed in the FINALLY block of a TRY-CATCH-FINALLY statement. A call to a label in a higher scope is not allowed in the following:</p> <ul style="list-style-type: none"><li>▶ A BEGIN-END block that contains a DATA statement using an object variable</li><li>▶ The CATCH block of a TRY-CATCH-FINALLY statement</li><li>▶ A FOREACH statement whose loop variable is an object</li></ul>
LHSBOX	<p>865        Explicit box invalid on left side of equate statement</p> <p>A variable on the left side of an assignment cannot be boxed. (This is because the boxed object would be temporary and would disappear after the statement executed.)</p>

LINLNG	301	Logical line too long You've specified too many continuation lines for a statement or XCALL, making your logical line too long. A logical line can have a maximum of 1023 characters of compilable text (excluding comments, tabs, and preceding and trailing blanks).
LISWRI	213	Cannot write to listing file The compiler cannot write to the specified listing file.
LNGTITL	314	Title too long The title that you've specified as the listing page header in the .TITLE compiler directive is longer than 128 characters. This error is only generated by the <b>db18</b> compiler.
LOGTOOBIG	357	Logical expression too large You've specified too many Boolean operations (with the Boolean operators .AND., .OR., and so forth) on the same logical line. (Level 2)
LSLENX	215	Listing length expected You've specified the LENGTH compiler option without specifying a value for the length of each page of the listing. (Level 3)
LSTXP	126	List file name expected You've specified the list compiler option on your command line without specifying a list filename to which to generate the program listing.
LSWIDX	214	Listing width expected You've specified the WIDTH compiler option without specifying a value for the width of the program listing. (Level 3)
LVARBOX	961	Explicit box invalid on loop variable in %s You've attempted to explicitly box a variable within a loop variable. For example, the following is not allowed: <code>foreach ((@a)rec) in al</code>
LVARTYP	913	Incompatible loop variable type %s in FOREACH The specified loop variable is a boxed loop variable, which is invalid in a FOREACH statement. (traditional Synergy only)
LYTPARTS	905	Cannot apply StructLayout attribute on partial structure A structure defined with the PARTIAL modifier cannot have the StructLayoutAttribute on it.

## Error Messages

### Compiler Errors

MAININDLL	975	MAIN not allowed in DLL  You've attempted to build a source file that contains the MAIN statement into a DLL. A main routine is not allowed in a DLL.
MAXLINE	879	Exceeded maximum number of lines in file %s  You've attempted to compile a file that has more than 262,143 lines, which is the maximum the compiler can handle in a single file. Split the file into smaller files and then compile them.
MBOTH	660	%s cannot be both %s  You've declared two mutually exclusive modifiers for the same item. The following modifiers are mutually exclusive: <ul style="list-style-type: none"><li>▶ SEALED and ABSTRACT on a class</li><li>▶ VIRTUAL and ABSTRACT on a method</li><li>▶ PRIVATE and VIRTUAL or ABSTRACT on a method</li><li>▶ PUBLIC, PROTECTED, and PRIVATE</li><li>▶ NEW and OVERRIDE on a class member</li><li>▶ LOCAL, STACK, or STATIC</li><li>▶ STATIC and VIRTUAL, ABSTRACT, or OVERRIDE</li><li>▶ STATIC and CONST</li><li>▶ CONST and READONLY</li><li>▶ VOLATILE and READONLY</li><li>▶ VOLATILE and CONST</li><li>▶ BYREF and BYVAL</li><li>▶ BYVAL and OUT or INOUT</li><li>▶ BYVAL and OPTIONAL</li><li>▶ IN, OUT, and INOUT</li><li>▶ REQUIRED and OPTIONAL</li></ul>
METHDEL	890	Method %s does not match delegate  The method signature of the method passed into the constructor for the delegate does not match the method signature of the delegate.

MISDECL	962	Missing declaration of %s You've declared the implemented part of a partial method without declaring an unimplemented part. You can declare one partial method by itself without implementation. If you declare two partial methods with the same signature, one must be with implementation and one must be without implementation. However, if you declare one partial method with implementation by itself, this error is generated.
MISIMP	768	Does not implement %s One of the following occurred: <ul style="list-style-type: none"><li>▶ You did not provide an override for an inherited method declared as ABSTRACT.</li><li>▶ A member of an interface is unimplemented. If you declare a class that implements an interface, either that class or one of its ancestors must implement all members in the interface and any interfaces from which the interface inherits.</li></ul>
MISIMP2	897	Does not implement %s since it is either static, not public, or has the wrong return type The specified interface member was not implemented by the class. The class member definition must match that of the interface member and it must be a public instance member.
MISIMP3	969	Does not implement %s since case does not match A class that implements an interface must implement every aspect of that interface exactly as it is defined. Because a class member did not match the case of the specified interface member (even though it matched in all other ways), the interface member was not implemented. Change the case of the class member to match the interface member.
MISMAIN	984	Missing MAIN in EXE An attempt to compile to an executable can't be accomplished because there's no main routine.
MISNGFIL	840	Missing source file There are no files to parse.
MISTYP	659	Missing expected type for %s You did not specify a return type for the specified method. A method declaration requires the return type of the method being defined.
MODLOAD	899	Unable to load module %s : %s The specified error prevented a .NET module from being loaded.

## Error Messages

### Compiler Errors

MOREARGS	672	More argument values than declared parameters in routine %s  When resolving a routine call to a routine declaration, the compiler selects the best matching routine declaration, irrespective of access, based on the method name, number of arguments, and types of arguments. This error is generated if the best match has fewer parameters than the call.
MRGFILS1	351	MERGE requires at least 2 input files  You've specified only one input file (or no input files) to merge on the MERGE statement.
MRGFILS2	352	Too many input files for MERGE  You've specified more than seven input files to merge on the MERGE statement.
MTHOUTCLS	955	Cannot declare method outside of a class/interface  You attempted to declare a method that's not part of a class or interface.
MULTMAIN	675	More than one main routine encountered  Your program contains more than one main routine. You can specify only one MAIN statement within a program.
NAMEREQ	767	Name must be specified with group modifier %s  You did not specify the name of a data structure with the GROUP modifier in a .INCLUDE directive. When creating a data structure of the format  <i>[opt_req] [direction] GROUP = name</i>  you must specify the name of the data structure to create.
NAMESUB	132	.NAME not allowed in subroutines  The .NAME compiler directive should only be specified in the main routine, and you've specified it in a subroutine. This error is only generated by the <b>dbl8</b> compiler.
NARROWING	696	Narrowing conversion could cause loss of data  You specified an assignment or a passing of parameters that would cause a narrowing conversion (that is, an assignment from a larger source type to a smaller destination type) on an integer value. (Level 4)
NETALLOW	827	%s not allowed %s on .NET  The specified feature is not allowed in Synergy .NET. (Level 1)

NETAPI	918	Routine %s not supported in .NET The specified API routine is not included in Synergy .NET. In traditional Synergy, you will only get this error when using the <b>-qnet</b> (or <b>/NET</b> ) compiler option. (Level 1)
NETRFA	795	RFA variable must be type a in .NET In Synergy .NET, an RFA variable must be type <b>a</b> . (Level 1)
NETSUPRT	793	%s not supported in .NET The specified item is not included in Synergy .NET. In traditional Synergy, you will only get this error when using the <b>-qnet</b> (or <b>/NET</b> ) compiler option. (Level 1)
NEWARRINIT	966	Invalid syntax for dynamic array initialization on new When attempting to initialize a dynamic array in a new expression, you used parentheses instead of curly braces and a value instead of a "#". The correct syntax is <pre>record     arr, [#] int proc     arr = new int [#] {1, 2, 3} end</pre> (Note that you will get a different error message if you make only one of these syntax errors without the other.)
NEWREQ	770	NEW modifier is required on %s since it hides a member of an inherited class The specified method hides an inherited class method with the same signature, but the hidden member has not been marked with the <b>OVERRIDE</b> or <b>NEW</b> modifier. (Level 1)
NFND	662	%s not found The specified identifier could not be found. If you are attempting to use a member of a namespace that is not the current namespace, make sure you have imported the namespace whose member you want to use.
NLREC	134	Preceding RECORD empty The previous record contains no data. (Level 4)
NMETHODARG	676	%s not allowed as %s Type <b>n</b> is specified on a method that is not marked <b>UNIQUE</b> .

NOALIGN	365	Alignment not performed You've specified the .ALIGN compiler directive before a declaration that does not define any data (such as an overlaid data declaration statement, an EXTERNAL COMMON statement, or any field in an EXTERNAL COMMON declaration). No alignment will be performed. This error is only generated by the <b>dbl8</b> compiler.
NOARYBASE	954	0 or 1 expected The value specified for the .ARRAY directive is something other than 0 or 1. The only valid values are 0 and 1.
NOASM	n/a	No assembly file created The build process failed due to other errors, so an assembly could not be created. (Synergy .NET only)
NOAUTO	503	Autosizing not allowed here You cannot specify a field length of * (to designate automatic sizing) on an external literal.
NOBLOCK	358	Not within a BEGIN-END block The EXIT statement is not enclosed within a BEGIN-END block.
NOBYVAL	960	BYVAL not supported on %s. Use ^VAL instead The BYVAL modifier is only available in Synergy .NET. To pass a parameter as a value in traditional Synergy, use the ^VAL identifier.
NOCND	320	Previous .IFDEF/.IFNDEF statement expected You've specified an .IFT, .IFF, or .IFTF compilation control directives outside of an .IF, .IFDEF, or .IFNDEF conditional block. (Level 1)
NOCONS	861	Cannot find matching constructor in class %s A constructor that matches the arguments to the NEW statement cannot be found in the class that you're creating. (Level 1)
NOENDC	310	.ENDC expected Your code contains an .IF, .IFDEF, or .IFNDEF block without a closing .ENDC compiler directive.
NOENDC2	755	.ENDC expected A required .ENDC directive is missing at the end of the file. .ENDC must always close an .IF, .IFDEF, or .IFNDEF conditional block. (Level 1)



NOENDCASE	422	No ENDCASE for current CASE The current CASE statement has a BEGINCASE without a matching ENDCASE.
NOENDGRP	502	No ENDGROUP allowed You cannot specify an ENDGROUP statement here.
NOENDUSING	420	Missing ENDUSING statement Your code contains a USING statement without a closing ENDUSING statement.
NOEXTEND	682	Cannot extend record by %s bytes with overlay field %s An overlay field is extending a record. You can add a field to cover the overlay.
NOFCSUB	729	Cannot call subroutine %s as a function because it has no parameters The specified subroutine has no arguments and therefore cannot be called as function.
NOFIELDS	689	NOFIELDS cannot be used with another qualifier When using .INCLUDE to include from a repository, you specified the NOFIELDS modifier along with some other modifier. NOFIELDS cannot be specified in conjunction with any other modifier.
NOFXD	406	Only integer and decimal operands allowed (%s) You've used implied-decimal data in an expression in which this data type is inappropriate.
NOGLB	136	Not processing a GLOBAL section Your code contains an ENDGLOBAL statement with no matching GLOBAL statement.
NOGLOBAL	726	%s cannot be declared globally The compiler encountered a class that was declared outside of a namespace. A class must be declared within a namespace.
NOGROUP	501	No GROUP to end Your code contains an ENDGROUP statement with no matching GROUP statement.
NOIDENT	307	Identifier expected The current compiler directive (for example, .DEFINE) requires an identifier.

NOINSTAT	880	Cannot declare instance member in static class Only static members are permitted in a static class.
NOLEN	537	Length not allowed You've specified a length where a length is not allowed.
NOLOOP	356	Not currently within a loop statement Your code contains an EXITLOOP statement that's not within a looping statement (such as DO FOREVER, FOR, REPEAT, or WHILE).
NONCLS	898	Non-CLS type used in CLS declaration You've declared a descriptor type or another type that contains a descriptor type in a structure that was declared with the CLS option (indicating that it is a .NET value type structure). This is not allowed.
NONCLSUNSP	679	Non-CLS structures not supported %s You attempted to use ^TYPEOF on a non-CLS structure.
NOOPER	693	No operator %s One of the following occurred: <ul style="list-style-type: none"><li>▶ You tried to invoke a binary method using the specified binary operator syntax, and a binary operator method with the specified name, parameter types, and return type does not exist.</li><li>▶ A built-in operator other than ==, !=, .EQ., .EQS., .NE., or .NES. was used with a base type (@*) or an explicit boxed type (@d, @i, @a, or @struct).</li><li>▶ You tried to use ^INCR or ^DECR with a variable type that doesn't support increment or decrement operations.</li></ul>
NOOVR	786	No suitable %s found to override You have attempted to override a method or property that does not exactly match the signature of an inherited method or property.
NOPROSEC	138	No procedural section Your routine is missing a .PROC or PROC statement. You cannot compile a class that doesn't contain a method, as there is nothing to compile.
NOPRPMTH	852	Operation not allowed with missing or unmatched %s on property %s The specified property was called using a GET or SET syntax, but it is either missing a GET or SET method or the method does not match the call.

NOPRPSET	965	Cannot set read-only property %s You attempted to set the specified property, which only has a GET method and therefore cannot be set.
NOPSEUDO	800	Pseudo arrays not allowed in this context A pseudo array is not allowed as a local data statement type or within a class definition.
NOQUOTE	319	Quoted string expected The current compiler directive requires a string in quotation marks.
NORETURN	774	%s cannot have a return type You have declared a return type for the specified method. Constructors, destructors, and property accessor methods do not allow a return type.
NOSPECL	311	%s expected The specified item is required but missing from the specified context. (This can be anything from a punctuation mark that is missing from the current line to a statement that is missing from a declaration.) (Level 3)
NOTALIGNED	604	Physical memory alignment not guaranteed The boundary position specified for the .ALIGN compiler directive is greater than the natural register size of the machine on which it is being used. Memory alignment can only be guaranteed up to the natural register size, and it cannot be guaranteed when using MASK functionality on I/O statements. (Level 1)
NOTALLOWED	683	%s not allowed in %s One of the following occurred: <ul style="list-style-type: none"><li>▶ While importing a prototype, the compiler encountered implementation of a method, function, subroutine, property, indexer, or operator.</li><li>▶ A class type was declared within a local record within a re-entrant routine.</li><li>▶ The INIT statement was applied to a parameter type other than <i>structfield</i>.</li><li>▶ .NOPROTO or .PROTO was specified within a conditional preprocessing block.</li><li>▶ An ONERROR statement was used in the same routine as a TRY-CATCH statement.</li></ul>

- ▶ INRANGE or OUTRANGE was specified on an unranged USING statement.
- ▶ A duplicate CATCH variable type is present in a TRY-CATCH block.
- ▶ A return type was specified on a GET or SET method in a property.
- ▶ A keyword was used as an identifier in a DATA statement.
- ▶ An INIT was performed on an array reference.
- ▶ Multiple statements were used in a TRY-CATCH-FINALLY block without an encasing BEGIN-END block.
- ▶ Index access on a structure in a ^M data reference operation or on local data was specified in parentheses, not square brackets ([ ]). Square brackets are required for structures in ^M and local data.
- ▶ Data containing object handles was passed to routine arguments that are type alpha.
- ▶ Local data was referenced in an INIT statement.
- ▶ Ranging, indexing, or dimension access occurred in an INIT statement.
- ▶ RETURN, MRETURN, FRETURN, XRETURN, XCALL, CALL, EXITTRY, ONERROR, or OFFERROR occurred in a FINALLY block.
- ▶ An object handle was used in a parameter group.
- ▶ An indexer contains a routine, field, or record named “item”.
- ▶ A group was defined in a class structure.
- ▶ A nonstatic method was defined in a static class.
- ▶ The NEW keyword was used on a field in a class record.
- ▶ The MISMATCH modifier was used on a parameter in a non-unique method.
- ▶ A type other than **n** or **n.** (or **a** for a subroutine or function) was used on a MISMATCH parameter.
- ▶ A structure that contains an object handle was used as a parameter.
- ▶ A structure marked with the CLS modifier was used with ^M or declared in a real array (or anywhere an alpha is expected).

- ▶ A nonstring object was returned on a function.
- ▶ An object was preceded by an “@” character when being passed as a type argument.

NOTATTR	871        %s is not an attribute class  You tried to use a non-attribute class as an attribute.
NOTBEXP	882        Assignment in Boolean expression! Did you intend “==”?  An IF statement contains an assignment (=) of simple variables in a Boolean expression that might have been intended as a relational operator (== or .EQ.). For example,  <code>if (a = b) nop</code>  generates this warning. (Level 4)
NOTCALLABLE	741        Cannot directly call this method  You have attempted to call a constructor or destructor directly. A constructor is called whenever an instance of the class is created, and a destructor is called before an instance of the class is destroyed.
NOTCEXP	411        Not a compile-time expression  Synergy DBL must be able to evaluate all .DEFINE symbols at compile time. Either the current .DEFINE symbol includes a variable and therefore cannot be evaluated at compile time, or the current function is not supported at compile time.
NOTDEF	322        Identifier not defined: %s  You’ve attempted to .UNDEFINE an identifier that has not been defined in a prior .DEFINE directive line. This error is only generated by the <b>dbl8</b> compiler.
NOTEXE	367        Statement can never be executed  The compiler has detected a FOR loop that never executes. For example:  <code>for i from 10 thru 1   nop</code>  NOTEXE is usually a warning except on the USING statement, where it’s an error. To avoid this error on USING, make sure the null term (“( )”) is the last condition in the <i>match_term</i> list. (Level 1)
NOTIMPLE	313        Feature not yet implemented: %s  The feature you’ve attempted to use has not yet been implemented on this operating system.

NOTINFUNC	742	Not within a function	<p>You used the FRETURN statement in a subroutine or a method. FRETURN specifies the return value of a user-defined function and returns control to the calling routine. Use XRETURN or MRETURN for a subroutine or method, respectively.</p>
NOTINI	937	Global data section ‘%s’ not initialized	<p>In Synergy .NET, the global data section with the INIT option is not defined in the same assembly as the routine that references it, nor is it defined in a referenced assembly.</p>
NOTINPR	722	Previous .NOPROTO expected	<p>The .PROTO compiler directive was specified without a preceding .NOPROTO directive. .PROTO cannot be specified without a matching .NOPROTO.</p>
NOTINSUB	359	Not within function or subroutine	<p>You’ve specified XRETURN, FRETURN, ^PASSED, ^ARG, or ^ARGN in a main routine. None of these statements or data reference operations are permitted in a main routine.</p>
NOTINTRY	745	Not in TRY statement	<p>An EXITTRY statement occurs outside of a TRY block. (EXITTRY is not allowed in the FINALLY block.)</p>
NOTSTAT	716	%s required to access non-static member %s	<p>You have attempted to access an instance of a class within a static method using one of the following techniques:</p> <ul style="list-style-type: none"><li>▶ The <b>this</b> initializer</li><li>▶ The <b>parent</b> initializer</li><li>▶ Accessing any instance member of the class without using an instance variable, including nonstatic methods, nonstatic instance fields, and nonstatic properties</li></ul>
NOTSUPT	997	%s is not supported %s	<p>Disposable data is not supported in the FOREACH statement. Put disposable data in a nested BEGIN-END block. You might also get this error if an array in a portable class library or Universal Windows Platform (UWP) app has more than three dimensions. These project types don’t support arrays with more than three dimensions.</p>

NOWINRT	990	%s not supported in WinRT
		When compiling with the <b>-rt</b> option, syntax that is not supported in WinRT occurred. The following are examples of unsupported features:
		<ul style="list-style-type: none"> <li>▶ An INOUT dynamic array declared as a parameter</li> <li>▶ The LPQUE statement</li> <li>▶ STOP “aexp”</li> <li>▶ The SLEEP statement</li> <li>▶ OPEN “tt:”</li> <li>▶ A specific boxed type, such as @int, in a cast or type declaration</li> </ul>
		If you need to box an object in WinRT, you should cast to object.
NOWRITE	736	Cannot write to this entity
		A property was encountered as a destination in a SET, CLEAR, or INIT statement. A property cannot be a destination in any of these statements.
NOWRITE2	949	Cannot write to %s
		You cannot write to the specified item.
NSRES	661	%s namespace reserved
		You have declared a namespace that is reserved by Synergy DBL. Reserved namespaces include “System” or any nested namespace within the System namespace, “Synergex” or any nested namespace within the Synergex namespace, and “SynGlobal.”
NULLCOAL	1026	Left side of a null coalescing operator must be a reference type or nullable
		A null coalescing operator was used on a variable that is neither a nullable type nor a reference type (boxed or object handle).
NULPR	148	No primary files specified
		The compiler could not find the <b>.dbl</b> source input file on the command line.
NUMALPHARQD	690	Numeric or alpha type required: %s
		An I/O statement requires a numeric or alpha data type if the type is not a built-in type or a string.

NUMFILES	889	Too many files  The OpenVMS FILLM parameter was not high enough to open all of the .INCLUDE files. This error is reported in place of the Synergy “Cannot open: %s” error (OPENIN2) and the OpenVMS “EXQUOTA, exceeded quota” error when an EXQUOTA error is detected on the open of an .INCLUDE file.
NUMINCL	82	Too many nested .INCLUDE levels  Your code contains more than 20 nested .INCLUDE source levels.
NUMINCL2	748	Too many nested .INCLUDE levels  You have exceeded the maximum of 20 nested .INCLUDE levels.
NUMLINES	920	Too many file lines. Subsequent debug line information ignored  The file position of a routine being compiled with the debug option exceeds 16777215 (0xFFFFF). (The file position is the byte offset of the beginning of the line from the beginning of the file.) The line number information for subsequent file segments is ignored for the current routine.
NUMREQ	410	Numeric expression required  The current operation requires a numeric expression.
NUMSEGS	919	Too many file includes (more than 255). Subsequent debug line information ignored  More than 255 file segments have been .INCLUDEd in a routine being compiled with the debug option. The line number information for subsequent file segments is ignored for the current routine.
NUMTYPARG	921	Incorrect number of type arguments passed to %s  You attempted to define a generic type but did not provide enough type arguments. You must specify the same number of type arguments as the generic type has type parameters.
NUMXP1	401	Numeric operand required (unary %s)  The current arithmetic expression requires a numeric operand.
NUMXP2	405	Numeric operands required (%s)  The current arithmetic expression requires a numeric operand.
NVRTF	652	Cannot resolve return type for function %s  You have declared an invalid return type for a function. See <a href="#">“Where data types can be used”</a> in the “Defining Data” chapter of the <i>Synergy DBL Language Reference Manual</i> for a table that specifies valid return types.



If you get this error and you are using **gennet40**, we recommend that you either use the include file generated by **gennet40** or separate your **gennet40** prototypes into a different directory from the ones in SYNIMPDIR.

NVRTM	651	Cannot resolve return type for method %s	<p>You have declared an invalid return type for a method. See “<a href="#">Where data types can be used</a>” in the “Defining Data” chapter of the <i>Synergy DBL Language Reference Manual</i> for a table that specifies valid return types. If you get this error and you are using <b>gennet40</b>, we recommend that you either use the include file generated by <b>gennet40</b> or separate your <b>gennet40</b> prototypes into a different directory from the ones in SYNIMPDIR.</p>
NVTEVT	891	Cannot resolve type for event %s	<p>Within the event declaration, you specified a delegate identifier that is not declared.</p>
NVTF	654	Cannot resolve type for field %s	<p>An invalid data type was declared for the specified field. See “<a href="#">Field definition components</a>” in the “Defining Data” chapter of the <i>Synergy DBL Language Reference Manual</i> for a list of valid data types for fields. If you get this error and you are using <b>gennet40</b>, we recommend that you either use the include file generated by <b>gennet40</b> or separate your <b>gennet40</b> prototypes into a different directory from the ones in SYNIMPDIR.</p>
NVTG	655	Cannot resolve type for group %s	<p>The group type was specified, but because it was not a type that was declared or imported, the compiler could not resolve it. If you get this error and you are using <b>gennet40</b>, we recommend that you either use the include file generated by <b>gennet40</b> or separate your <b>gennet40</b> prototypes into a different directory from the ones in SYNIMPDIR.</p>
NVTP	653	Cannot resolve type for parameter %s	<p>The argument type that you specified is invalid for the type of routine being declared. See “<a href="#">Where data types can be used</a>” in the “Defining Data” chapter of the <i>Synergy DBL Language Reference Manual</i> for a table that specifies which data types are valid in which circumstances. If you get this error and you are using <b>gennet40</b>, we recommend that you either use the include file generated by <b>gennet40</b> or separate your <b>gennet40</b> prototypes into a different directory from the ones in SYNIMPDIR.</p>

NVTPR	656	Cannot resolve type for property %s The specified property type was not one of the types that was declared or imported and therefore could not be resolved by the compiler. If you get this error and you are using <b>gennet40</b> , we recommend that you either use the include file generated by <b>gennet40</b> or separate your <b>gennet40</b> prototypes into a different directory from the ones in SYNIMPDIR.
NVTREF	884	Cannot find %s used in %s The specified assembly contains an unresolved type reference. Add an assembly reference that has a definition of the specified type.
OBJDIS	885	Object returned from %s discarded A method that returns an object is called in such a way as to discard the object returned. (Level 4)
OBJHNDXP	565	Object handle/function expected You have attempted to access an instance of a class within a static method using the <b>this</b> initializer or the <b>parent</b> initializer, or you are accessing an instance member of the class (including nonstatic methods, nonstatic instance fields, or nonstatic properties) without using an instance variable.
OBJWRI	144	Cannot write to object file The compiler has encountered a disk I/O problem. One possibility is that your disk is full.
ODIGXP	147	Octal digits expected in argument (%s) You must specify an argument that contains only octal digits (0 – 7). (Synergy .NET only)
OBJXP	149	Object file name expected You’ve specified the Object compiler option without specifying an object filename. (Level 3)
OLYBD	150	Overlay not allowed A record overlay has been specified illegally. See “ <a href="#">Overlaying data</a> ” in the “Defining Data” chapter of the <i>Synergy DBL Language Reference Manual</i> for a list of overlay rules.
OLYBG	511	Overlay record too big The size of the overlay record exceeds the size of the record it is overlaying.

ONEALPHA	368	Only one character allowed: %s  You've specified a RANGE on a USING statement, and the alpha literal that you've specified as your RANGE or match value is more than one character. (Level 1)
ONERRDTA	874	Local data cannot be defined in a routine containing ONERROR/OFFERROR  A DATA statement was used in the same routine as an ONERROR statement and/or an OFFERROR statement.
OPARGTYP	714	Operator must have at least one argument of the enclosing type  A unary operator method must have an argument that is type @class, where class is the enclosing class.
OPENIN	158	Cannot open: %s  The compiler cannot open the specified file for input. This error is only generated by the <b>db18</b> compiler.
OPENIN2	747	Cannot open: %s  The compiler cannot open the specified file for input. (Level 1)
OPENMD	348	Invalid OPEN mode: %s  The specified OPEN mode is invalid. See <a href="#">OPEN</a> in the "Synergy DBL Statements" chapter of the <i>Synergy DBL Language Reference Manual</i> for a list of valid OPEN modes.
OPENOUT	159	Cannot open: %s  The compiler cannot open the specified file for output.
OPENSUB	349	Invalid OPEN submode: %s  The specified OPEN submode is invalid. See <a href="#">OPEN</a> in the "Synergy DBL Statements" chapter of the <i>Synergy DBL Language Reference Manual</i> for a list of valid submodes.
OPSTPUB	711	Operator must be declared STATIC and PUBLIC  A unary operator method must be declared as both STATIC and PUBLIC.
OPTEXP	346	%s option requires an assigned value  The specified qualifier requires a value. (For example, this error is generated if you specify the DIRECTION qualifier on the READ statement without a direction specification.)

OPTNOEXP	347	<code>%s</code> option does not allow an assigned value  You've assigned a value to a qualifier that does not support one. (For example, the REVERSE qualifier on the READ statement does not require any value specification.)
OPTOBJ	776	REQUIRED modifier must be used on object parameters  You declared an argument whose type is an object (for example, string, @class, @structure, @i4, boxed type) as OPTIONAL. All object parameters must be REQUIRED (which is the default if neither modifier is specified).
OPTREQ	1013	Passing optional parameter to required parameter <code>%s</code> in routine <code>%s</code>  You've passed an optional parameter into a required parameter in the specified routine. (Level 4)
OPTRUE	996	Calling <code>op_True</code> on <code>%s%s</code> : may not be what is intended  An object handle is implicitly cast as Boolean. To remove the warning, explicitly cast the handle as Boolean.
OPTTYP	345	<code>%s</code> expression expected for <code>%s</code> value  The specified value in the OPTIONS string of the OPEN or MERGE statement has the wrong data type.
OPVOID	712	Operator cannot return void  A unary operator method cannot return a VOID.
OUTOFRANGE	366	Select value is outside defined range  You've specified a RANGE on a USING statement, and one of the following conditions is true: (Level 1) <ul style="list-style-type: none"><li>▶ The match value is not within the range.</li><li>▶ The match value is a range of values and no value within this range is within the range you specified in your USING statement.</li><li>▶ You've specified a relative operation, and no values that match this operation are within the range.</li></ul>
OUTPARM	698	Must be able to write to argument <code>%s</code> because parameter was declared as OUT or INOUT  An argument that was declared as IN was passed to a parameter marked as OUT or INOUT.

OVRERR	738	Cannot override %s because it is not declared VIRTUAL, OVERRIDE, or ABSTRACT in class %s  The specified method or property cannot be overridden because it is not declared as VIRTUAL, ABSTRACT, or OVERRIDE in the parent class.
OVRSLD	830	Cannot override a SEALED %s  You have declared a method or property in an inheriting class to override a method or property marked as SEALED in the parent class.
OVRVIRT	787	%s hides inherited member in class %s. OVERRIDE or NEW required  The specified method hides an inherited class method with the same signature, but neither the OVERRIDE nor the NEW modifier has been declared for the hiding member. You must specify OVERRIDE or NEW. (Level 1)
PARMOVR	876	New or overridden method changes parameter type from %s to %s  A method has a different integer parameter than its parent. Change the non-matching parameter type in the inherited method so that all types match the parent.
PARSE	404	%s at or near {%s}  The specified syntax error has occurred.
PARSERR	599	Fatal parsing error: %d  A syntax error has occurred in your Synergy code. Call Synergy/DE Developer Support for assistance. This error is only generated by the <b>dbl8</b> compiler.
PARTACC	835	Accessibility declarations of partial %s %s do not match  Two separate partial declarations of the same item have different access modifiers (PUBLIC, PROTECTED, or PRIVATE).
PARTALPH	904	Cannot pass partial structure to an alpha parameter  A partial structure cannot be passed to an alpha parameter.
PASSIMPL	868	Implied decimal argument %s passed to non-implied parameter %s  You have passed a decimal value to a non-implied ( <b>d</b> or <b>n</b> ) parameter. Either make both the argument and the parameter implied or make both non-implied to eliminate the warning. (Level 4)

PASSUR	971	Passing %s for %s could give unexpected results
		You have passed a decimal or implied-decimal value into a MISMATCH alpha parameter that is not marked as IN, which could cause unexpected results when examining the decimal or implied-decimal after the call. You can do one of the following to avoid the warning:
		<ul style="list-style-type: none"><li>▶ Change the parameter to IN. (This is the recommended solution.)</li><li>▶ Explicitly set ^A around the decimal being passed on the call. For example,</li></ul>
		<pre>xcall mysub (^A(dvar))</pre>
		In traditional Synergy, you will only get this error when using the <b>-qnet</b> (or <b>/NET</b> ) compiler option. (Level 3 in Synergy .NET, level 4 in traditional Synergy)
PLATSUPT	987	%s not supported in WinRT
		You have attempted to use the specified feature, which is not supported on WinRT.
PMETHODARG	833	%s not allowed as %s
		Type <b>p</b> or <b>p.</b> was specified as an parameter type on a method.
PRNPROP	858	Parentheses not needed for property %s
		You have attempted to access a property like a method (for example, <code>c1.myprop()</code> ).
PRNRQD	857	Parentheses required on call to method %s
		The parentheses are missing on a call to the specified method.
PROPCONF	887	Property %s conflicts with %s in same class
		The name of a class member conflicts with the specified property name. (For example, a method named <b>get_myproperty</b> conflicts with a property named <b>myproperty</b> .) Rename either the property or the offending class member.
PROPPARM	784	Properties that are not indexers cannot have parameters
		You have specified one or more arguments for a property that is not an indexer property. Only indexer properties can have arguments.
PROTOMISMCH	705	%s %s does not match prototype
		When importing a prototype into the source file where an item is defined, the compiler found that the declaration for the specified item does not match the prototype. If this error is encountered and level 4

warnings are on (in other words, you're compiling with **-W4**), the compiler will provide the signature of the declared item and the prototype item to help you determine the differences between them.

PRTLOAD	947	Unable to load prototype %s : %s	A binary prototype could not be loaded due to the specified reason. Often this error occurs because a namespace wasn't imported for a type you are using. (Level 3)
PRTYPPRM	923	Type parameter %s for partial declaration does not match original declaration	A partial generic type declaration must have same number and type of type arguments as the original declaration.
PSCTRUNC	881	Psect name truncated	A global data section name is too long after the prefix ("\$\$" on OpenVMS and "\$GDS_" on Windows or UNIX) is added to the name. The maximum name length before the prefix is 29 on OpenVMS and 25 on Windows or UNIX. (Level 4)
READONLY	734	Cannot write to read-only data	One of the following has occurred: <ul style="list-style-type: none"> <li>▶ You've attempted to initialize a field in a class that was declared with the READONLY modifier. The value of a READONLY field can only be set during declaration or in the constructor of the class.</li> <li>▶ You've attempted to assign a value to a parameter defined with an IN modifier.</li> </ul>
RECBIG	1	%s size too large	The RECORD data area is too large. A named record must have fewer than 65,535 bytes.
RECETY	839	Record is empty	A record does not contain any declared members. (Level 3)
RECURSE	851	Recursion not allowed on %s	A type declaration was used inside another type declaration that it was a part of (for example, a field that was part of struct1 was declared to be of type struct1). This recursion must be removed.
RECWHANDLE	697	Cannot %s a record that contains a handle	You attempted to perform an illegal operation (for example, write) to a record, group, or structure, that contains a handle.

REFARYSCP	832	Cannot apply scope to reference array  Scope cannot be applied to an array of class types or a collection. For example, if <b>fld1</b> is declared as type <b>@myclass</b> , you can't use it as follows:  <code>x = ^size(fld1[])</code>
REFCOMPARE	959	%s compares instance not contents  In Synergy versions prior to 9.5, an equality or inequality comparison of base types (@*) or explicit boxed types (@d, @i, @a, or @struct) automatically unboxed a boxed object. In version 9.5 and higher, it instead compares the reference (instance) rather than the contents. The REFCOMPARE warning is generated if either a base or boxed type equality or inequality comparison previously resulted in unboxing. You can eliminate the warning by explicitly casting the object handle. (Level 4)
REFLIT	838	Integer literals cannot be passed by reference due to optimization  An integer literal is being passed by reference. If compiler optimization is being performed (for example, by default or if the <b>-qoptimize=1</b> compiler option is set), you can't pass integer literals by reference. (Level 1)
REFREQD	836	^REF required for argument %s  You have called a method that passes an argument without ^REF, but the argument is declared as ^REF in the method declaration.
REFTYPMSMCH	934	Parameter %s must match type exactly because it is passed by reference  Value types passed as BYREF parameters (whether explicitly or by default) must match exactly. A mismatch exists between the specified parameter and the parameter it is being passed to.
REQDIM	512	Dimension specifications required for { %s}  You've attempted to reference the entire scope of a pseudo array using a null subscript (pseudo[]). Synergy DBL only allows you to refer to one element of a pseudo array at a time.
REQPARAM	673	Missing required parameter %s in routine %s  The specified required argument was not passed on a call to the specified routine. A call to a routine that declares a required parameter must provide a value to that parameter.



RESWORD	760	Reserved word %s
		The following keywords cannot be used as identifiers for anything but local variables, data statements, routine arguments, common records, or global data outside of a class:
		<ul style="list-style-type: none"> <li>▶ this</li> <li>▶ parent</li> <li>▶ Synergy relational operators EQ, NE, GT, LT, GE, LE, EQS, NES, GTS, LTS, GES, LES, EQU, NEU, GTU, LTU, GEU, LEU</li> <li>▶ Synergy Boolean operators OR, AND, NOT, XOR</li> <li>▶ Synergy bitwise operators BOR, BXOR, BAND, BNAND, BNOT</li> </ul>
		Also, a local variable within a class cannot be named <b>this</b> or <b>parent</b> . (Level 3)
RETEXP	684	%s expected in %s
		The specified method does not contain an MRETURN statement.
RETRQD	826	%s missing in %s
		The XRETURN or FRETURN statement is missing from the specified subroutine or function. (Level 4)
RETTYP	859	Return type for %s assumed to be %s
		The return type was not explicitly declared or cannot be determined by looking at the FRETURNS in the specified function. You can turn off this warning by explicitly declaring the function type.
RFAERR	1023	RFA error: size must be 6 or 10
		The field passed to an RFA: or GETRFA: IO qualifier has a size other than 6 or 10. The RFA must be an <b>a6</b> field and the GRFA must be an <b>a10</b> field.
RNGARY	1022	Range on array acts on first element
		A reference was made to an entire array with no explicit element reference (for example, “ary(1:2)” instead of “ary[2](1:2)”). (Level 4)
SAMXP	403	Operands must be both alpha or both numeric (%s)
		You’ve specified an expression that contains different data types. An expression requires like data types, such as <b>n + n</b> or <b>a + a</b> (not <b>n + a</b> ).

## Error Messages

### Compiler Errors

SCPRNG	1006	Scope used instead of range on %s You've ranged over an entire array, causing the range to change to a scope operator. To avoid getting this warning, use the scope operator instead ([]). (Level 4)
SEGBIG	514	%s segment too big A data element has caused a program section to exceed hex FFFF bytes at compile time. The program has too much data.
SFLDINV	370	Invalid usage of STRUCTURE field A structure ID was used where a field or parameter was expected.
SFLDREQ	369	STRUCTURE field required You've specified a ^M statement that does not reference a STRUCTURE field.
SGN2USGN	951	Assigning negative literal value to unsigned field A signed literal integer value is assigned to a unsigned integer field. (Level 4)
SRCLINBIG	544	More than 65534 source lines (%s). Debug not possible Under rare circumstances, it is possible for the number of source line numbers to exceed 65,534 in the compiler. This error is only generated by the <b>dbl8</b> compiler.
SRCLINBIGW	545	More than 65534 source lines (%s). Reported error line incorrect Under rare circumstances, it is possible for the number of source line numbers to exceed 65,534 in the compiler. This error is only generated by the <b>dbl8</b> compiler.
SRCLINLNG	976	Source line too long A logical line in your source code is longer than 8192 bytes.
STARFIELD	677	'*' not allowed in parameter field specification You declared a parameter that specifies * as the size (for example, parm1, a*). <i>Size</i> cannot be an asterisk in a parameter definition.
STKMAIN	831	STACK not allowed on MAIN and has therefore been ignored The STACK modifier cannot be specified on the MAIN statement and is ignored by the compiler. (Level 1)

STKOVRSZ	956	Data division stack larger than reserved stack  The amount of stack storage exceeds 1 MB (unless the size was set by the <b>-stack</b> compiler option). STKOVRSZ is reported as an error when building an <b>.exe</b> file and as a warning when building a <b>.dll</b> file. (Synergy .NET only) (Level 3)
STKSIZ	1008	Stack data size greater than %s  The specified total stack data used by a routine exceeds 65536 bytes or the limit set by <b>-qwarnstacksize</b> . (Synergy .NET only) (Level 1)
STMTXP	340	Statement expected  You've specified an empty compound statement. For example, the following causes this error to be generated:  for 1 from 1 thru 90 begin end
STRTOPT	185	Unrecognized .START option: %s  You've specified an invalid option on the .START compiler directive. Valid options are /NO/COND, /NO/LIST, /NO/OFFSETS, /NO/PAGE, AND /NO/SUMMARY. This error is only generated by the <b>dbl8</b> compiler.
STRUCTHIDE	844	Local structure %s hides non-local structure %s  A structure inside a method hides a structure outside a method. (Level 4)
STXPFOR	516	Statement expected following FOR  You've specified a FOR loop without the statement to be executed. This error is only generated by the <b>dbl8</b> compiler.
SYMDEFD	312	Symbol already defined: %s  A .DEFINE compiler directive is attempting to define a symbol that's already been defined. (Level 3)
SYMND	183	Record or field not declared  You've referenced an undefined symbol.
SYSRTNOVR	706	System routine overridden. Invalid binding may occur  You have specified a local routine that hides a system routine. As a result, binding may work correctly. (Level 4)

## Error Messages

### Compiler Errors

TERNMSMCH	1015	Type mismatch between %s and %s on each side of the “:” in a ternary operator  The expressions on either side of the “:” in a conditional operator aren’t the same type. <i>Expression1</i> and <i>expression2</i> can be any type, object, or value type, but they must be compatible types with one other.
TOKUDF	187	Symbol %s already uniquely defined  You’ve declared two or more common variables with the same name and non-unique paths, or two or more identifiers are not unique within a single scope.
TOOMANYSYM	543	More than 65534 symbols in debug source module. Excess ignored  There is a limit of 65,535 symbol entries in a debugger table per routine. The compiler has marked all referenced symbols as referenced. All symbols that have not been referenced are then marked as referenced until the limit of 65,535 symbols has been reached, at which point this warning is issued and no additional symbols are emitted. If this limit has been reached, the symbols that have not been emitted into the debug table may not be used to debug the routine. (Level 3)
TOOMNYSRC	546	More than 254 source modules  The maximum number of source files per compiled routine is 254. This limit was exceeded. This error is only generated by the <b>dbl8</b> compiler.
TOOSHORT	694	Argument too short for %s  The size of the MASK variable to an I/O statement is too small.
TPRMITYP	931	Type parameter %s must be an identifier and not a type  A type parameter cannot take the name of a type.
TPSTAT	936	Keyword %s is invalid with static class member  The <b>this</b> or <b>parent</b> initializer was immediately followed by a static class member. To fix the problem, include the class name when accessing the static class member.
TRUEINT	713	Operator op_True must return integer  The unary operator for the op_True method must return an integer type.
TRUNC	195	Source line too long  The physical line is too long. A line cannot be longer than 255 characters (excluding line terminators, such as carriage returns and line feeds).

TYP32	<p>941        Type %s cannot be %s on 32-bit platforms</p> <p>You've used the specified data type in an invalid location on a 32-bit platform. For example, an <b>i8</b> or <b>long</b> is not permitted as the return type for a method or function or as the data type for a property on a 32-bit platform. (Level 3)</p>
TYPCONS	<p>925        Type %s does not meet constraint</p> <p>One of the following occurred:</p> <ul style="list-style-type: none"><li>▶ You've declared a class constraint on a type parameter, and the type used to create the constructed type does not match or is not a child of the class mentioned in the constraint.</li><li>▶ You've declared an interface constraint on a type parameter, and the type used to create the constructed type does not implement the specified interface.</li><li>▶ You've declared a constructor constraint on a type parameter, and the type used to create the constructed type does not have a default constructor.</li><li>▶ A generic class inheriting a constraint provides a constraint on the same type parameter, which conflicts with the parent's constraints.</li></ul>
TYPHID	<p>968        Built-in type %s can be hidden by %s</p> <p>The specified method or class identifier matches the name of a built-in type, thus hiding the type. Change the name of the method or class to avoid problems. (Level 3)</p>
TYPMISMCH	<p>686        Type mismatch between %s and %s</p> <p>The specified type names don't match, which is most likely due to one of the following situations:</p> <ul style="list-style-type: none"><li>▶ The type of one or more variables being set in the SET statement is not compatible with the type of the value being assigned.</li><li>▶ When importing a prototype of a subroutine or function into the source where it is called, the compiler compares the signature of the call with that of the prototype of the routine. The specified argument type did not resolve to the parameter type within the prototype.</li></ul>
TYPARM	<p>670        Type mismatch for parameter %s in routine %s</p> <p>You have attempted to pass an argument type that doesn't match or cannot convert to the parameter type, and the parameter was not defined with the MISMATCH modifier. (For more information about the MISMATCH modifier, see "<a href="#">Parameter definitions</a>" in the "Defining Data" chapter of the <i>Synergy DBL Language Reference Manual</i>.)</p>

## Error Messages

### Compiler Errors

TYPRET	671	Type mismatch for return type  The expression passed to MRETURN was not the same type as the return type of the method, or a function that doesn't have a declared type has an FRETURN type that is different from the first FRETURN.
TYPUNEXP	908	Type %s unexpected here  The item specified in the error message is a type name, and it was passed where it wasn't expected. Check whether you meant to pass a field of the specified type instead of the type name itself.
UNBXCNCL	866	Auto unbox cancels out boxing of %s. Boxing removed  Because automatic unboxing would cancel out boxing of the specified item, the box has instead been removed for optimization purposes. (Level 4)
UNDEFLBL	338	Undefined label: %s  The specified GOTO or CALL target is not defined.
UNKNOWN	n/a	Unexpected and undefined compiler exit  An error situation that isn't handled by any of the other compiler errors occurred, such as an out-of-memory situation. This error is hardcoded and doesn't have an error number.
UNQOVR	773	A UNIQUE method cannot be overloaded, overridden, or redeclared  You have attempted to overload, override, or redeclare one of the following: <ul style="list-style-type: none"><li>▶ A method that was declared with the UNIQUE modifier</li><li>▶ A subroutine or function inside a class declaration</li></ul>
UNSIGNED	1028	Loading unsigned assembly %s  When an assembly is built with a specified keyfile, all referenced assemblies are checked to see if they are signed. If you're building with the <b>-portable</b> option and an assembly is not signed, a W_UNSIGNED warning is reported. (Level 3)
UNSPFMT	993	Unsupported file format  The file contains a UTF-16 file marker, which is not supported.
UNSUPPORT	692	Unsupported syntax  The syntax you have used is not supported in this version of Synergy DBL.

UNTYPCAST	964	Boxing of untyped literal; explicit boxing cast (@a/@d/@i/@int) recommended  An untyped literal was explicitly cast to the base type (Object) or passed to a method parameter of type System.Object and implicitly boxed, but the actual boxed value type may be unknown. We recommend explicitly casting the literal to the expected type using typed boxing casts (@d, @i, @a, @int, @decimal, etc). An explicit cast suppresses the warning.
UNXPTOK	590	%s (%s)  A preprocessor tokenizing error occurred.
USDBDECL	973	Data statement %s used before declared  A DATA statement with the specified variable was declared after it was used. Move the DATA statement declaration before its first use in the code.
UTLXP	202	UNTIL statement expected  You've specified a DO statement without the corresponding UNTIL statement.
VALIDXP	823	Valid identifier expected at or near { %s }  The syntax of an identifier is not valid.
VALREQD	735	^VAL required for argument %s  You have called a method that passes an argument without ^VAL syntax, but it is declared as a BYVAL or ^VAL in the method declaration.
VARARGARY	946	If ^VARARGARRAY used, no other extra arguments allowed  When ^VARARGARRAY is used in a call, it must be the last item in the argument list.
VARGOBJ	902	Cannot pass object type for %s as a vararg value  You've attempted to pass an object as a value for a variable argument. Even if a routine uses the VARARGS modifier, an object can only be passed as a declared argument; it cannot be passed as an argument beyond the declared number of arguments.
VRNTXP	216	Variant value expected  You've specified the variant compiler option with a negative or nonnumeric ^VARIANT value. (This error may also be displayed as VARXP.)

WALIGN	1018	Align warning: %s This warning may occur with the following text: ► Structure padded because of alignment A field in the structure contained an alignable type, causing the structure to be padded to the same alignment size. Alignable types include double, float, or reference types (e.g., a handle to a class, an array, etc.).
WRIFIL	160	Error writing %s The compiler could not write to one of its work files.
WRILIT	331	Writing to a literal or missing argument The current XCALL statement is supposed to return data, but it cannot because the argument it's supposed to write to is either a literal or is missing altogether.
WRNGRC	201	Field not in this %s You've used the position indicator (@) to position a variable outside the current record or group.
WRODEFARG	371	Wrong argument count in define reference You've specified an incorrect number of arguments in a macro call.
XCALLHFUNC	733	Cannot XCALL a ^ function You have attempted to call a data reference operation as an external subroutine. The only data reference operation that can be XCALLED is ^SIZE.
XFMTHSTR	870	Structure %s used in XFMethod %s must come from repository A structure used with the xfMethod attribute must be .INCLUDED from the repository; it cannot be defined in the source file.
XMLDOC	801	%s A cref property is missing a quote or there is a parameter tag for a parameter name that doesn't exist. (Level 1)
XRETVAL	825	XRETURN cannot return a value on .NET Although you can use XRETURN to return a value on a subroutine in traditional Synergy, this functionality is not included in Synergy .NET. (Level 1)



XTRAEND	336	Too many END statements Your code contains at least one END statement that doesn't have a matching BEGIN statement.
XTRASRC	533	Only the first source file on the command line was compiled On OpenVMS, you can only specify one source file on the compiler command line unless you separate each filename with a plus sign (+). Because your source files were not connected with a plus sign, only the first file was compiled. This error only occurs on OpenVMS. (Level 3)
XTREND	44	Too many .ENDC statements Your code contains at least one .ENDC compiler directive without a matching .IF, .IFDEF, or .IFDEF directive.
YLDERR	992	Yield error: %s A YIELD statement occurred in a LAMBDA function, a CATCH block, a FINALLY block, a TRY block that contains a CATCH statement, or a method whose return type is not IEnumerable or IEnumerable<>.
ZEROSIZ	985	%s has zero size A non-parameter group with a size of 0 is not supported in .NET.

## Informational error messages

The following errors provide additional information about other errors.

BADDSCR	1217	Corrupted descriptor: type = %d, class = %d An internal failure has occurred.
ERTEXT	1052	%s This message provides more information to support other error messages.
ERTXT2	1053	%s%s This message provides more information to support other error messages.
EXPAN	529	Within expansion: %s This message provides additional information about an expanded replacement identifier (a .DEFINED constant) that caused a syntax error.
INCFIL	531	Occurring in the source file: %s This message displays the expansion of a replacement identifier.

## Error Messages

### Compiler Errors

OPWCRE	1149	Operation was CREATE This message provides additional information about file I/O errors.
OPWDEL	1146	Operation was DELETE This message provides additional information about file I/O errors.
OPWFND	1144	Operation was FIND This message provides additional information about file I/O errors.
OPWRDS	1145	Operation was READS This message provides additional information about file I/O errors.
OPWRED	1143	Operation was READ This message provides additional information about file I/O errors.
OPWSTO	1147	Operation was STORE This message provides additional information about file I/O errors.
OPWWRI	1148	Operation was WRITE This message provides additional information about file I/O errors.
YIELD	541	Resulting in: %s This message specifies the source file in which an error occurred when the error occurred in an .INCLUDEd file.

# Linker Errors

## Fatal error messages

The following error messages cause the linker to abort.

BADFIL	37	Bad record format in file: %s
		You might get this error for any of the following reasons:
		<ul style="list-style-type: none"> <li>▶ Your object code is extinct.</li> <li>▶ Your input file is not an object file, an OLB, or an ELB.</li> <li>▶ The order of your object code is invalid.</li> <li>▶ Your object module has been truncated.</li> </ul>
		To fix this problem, recreate your file from scratch.
BADNDN	49	Unrecognized module id 0x%x in %s
		Your module was compiled on a machine of the wrong endian type. Recreate your file from scratch.
BIGSEG	1	Segment exceeds maximum size in module %s
		A data segment is too big to be linked.
BITOPP	89	File %s built with opposite bit size
		An attempt was made to load an ELB built with the wrong architecture size (32- or 64-bit) for this linker.
CIRELB	54	Circular ELB reference in file %s
		The ELB being created is also directly or indirectly referenced as input. For example, the following set of <b>dblink</b> commands will cause this error:
		<code>dblink -l util util.dbo mylib.elb</code>
		<code>dblink -l mylib mylib.dbo util.elb</code>
		The first command creates <b>util.elb</b> and links it against <b>mylib.elb</b> . The second command tries to create <b>mylib.elb</b> and link <b>util.elb</b> against it. However, when <b>util.elb</b> is opened and the reference to <b>mylib.elb</b> is encountered, <b>mylib.elb</b> will be opened and a circular reference occurs.

CLSCRC	59	Class CRC mismatch: class %s in module %s
The specified module contains a class with a different cyclic redundancy check value than the same class from a previous module. This means that the same class name is used with two different layouts in files that are linked together. The following are some example scenarios in which this error would occur:		
<ul style="list-style-type: none"><li>▶ One file imports a class and is compiled, and another file imports a modified version of the same class and is compiled, and then the two files are linked together.</li><li>▶ Object-oriented code is prototyped with <b>dblproto -single</b>.</li></ul>		
CLSUND	60	Undefined class: %s
The specified class was referenced but never defined.		
CMDBIG	2	Command line exceeds maximum length
The specified command line was too long. Use continuation lines.		
COMDUP	51	Duplicate common symbol definition: %s in module %s
A global literal or global common symbol has been defined more than once. Change the name of the common or literal or change one from GLOBAL to EXTERNAL.		
COMERR	3	Compiler errors in module: %s
Your module contained errors when it was compiled. Remove the errors and recompile the module.		
COMNF	4	COMMON record undefined: %s in module %s
The common or literal does not have a global definition. Change the external common or literal to global or add an EXTERNAL or GLOBAL definition.		
DUPCLS	68	Duplicate class: %s in ELB %s
The specified class in the specified ELB has been defined more than once, and the sizes of the duplicates are different.		
DUPCOMN	63	Duplicate Global Common: %s in ELB %s
The specified global common in the specified ELB has been defined more than once, and the sizes of the duplicates are different.		
DUPGBL	67	Duplicate global: %s in ELB %s
The specified global variable in the specified ELB has been defined more than once, and the sizes of the duplicates are different.		

DUPGDS	65	Duplicate Global Data Section: %s in ELB %s The specified global data section in the specified ELB has been defined more than once, and the sizes of the duplicates are different.
DUPLICATES	70	Duplicate symbols A symbol has been defined more than once.
DUPLIT	64	Duplicate Global Literal: %s in ELB %s The specified global literal in the specified ELB has been defined more than once, and the sizes of the duplicates are different.
DUPMOD	6	Duplicate module name: %s The module has been specified more than once. Remove or rename one of the modules.
DUPSREC	66	Duplicate Static Record: %s in ELB %s The specified static record in the specified ELB has been defined more than once, and the sizes of the duplicates are different.
DUPSYM	7	Duplicate symbol definition: %s The psect or module has been defined twice. Remove one of the modules containing the duplicate.
E65K	39	Modules or shared data symbols > 65K Either the number of modules being put into a .dbr or ELB file exceeds 65535, or the sum of the lengths of the common data symbol names exceeds 65535.
ELBNAM	43	ELB name specified is too long An ELB has been specified with a name longer than 31 characters.
ERRCNT	36	Too many errors; compilation aborted The number of errors exceeded 20, which is the maximum. (Level 1)
FILINUSE	52	File %s is open by another user The linker is attempting to open a file that is open by another user.
GBLPSNF	38	Global psect {%d} not found An error has occurred in the linker's processing.
GLOBNF	8	Global psect {%d} not in module %s The psect was referenced but not defined. Check for coding errors in the module and recompile.

## Error Messages

### Linker Errors

INSMEM	9	Insufficient memory for attempted operation The program could not allocate enough memory for the linker to perform its function. Make more memory available for the linker to use.
INTERR	517	Internal failure: %d An error has occurred in the linker's processing.
INVFN	10	Invalid command file name You've specified an indirect command file with an invalid filename.
INVNUM	33	Option requires numeric value specification You've specified the stack size linker option without specifying the size of the stack.
INVOPT	11	Invalid option You've specified an invalid option on the command line. See <a href="#">“Linking Object Modules” on page 1-37</a> for a list of available Synergy linker options.
INVSOFT	12	Invalid switch or file name You've specified an invalid command line switch or filename.
LIBMAX	15	Too many library input files You've specified more than 32 OLBs or ELBs.
MAXIF	16	Too many input files open You've input command lines to the linker by entering them indirectly through a command file, but the depth of indirect command files specified was too deep.
MAXTF	18	Too many files The number of input files exceeded 265. Reduce the number of input files by using object libraries.
NDNBAD	88	Unrecognized file ID in %s The specified file contains an unidentified magic number. This error occurs, for example, if a version 5 or version 6 ELB is specified on a link line for a version 9 linker, or if a non-ELB file is specified where a valid ELB should be.
NDNOPP	87	File %s built with opposite 'endian' An attempt was made to load an ELB built with the wrong byte order for this machine.

NOCLS	57	Class not found Either a CLSDEF object record specifies a class that was not specified in a CLSDECL object record, or a CLSREF object record specifies a class that was never declared. This error should never occur unless the object file was not created correctly by the compiler.
NOENDMOD	42	Internal error: No ENDMOD record in %s The physical end of file was reached without finding an ENDMOD object record. The file was truncated. Recreate the file that contains the module, and recompile if necessary.
NOINF	81	No input module specified. No input modules were specified through DBO or OLB files.
NOLIB	20	No library input allowed for library output You've attempted to extract a module from an object file or ELB. A module can only be extracted from an object library.
NOMAIN	21	No main-line or primary module specified You've attempted to create a <b>.dbr</b> file without specifying a main routine. Specify a file that contains a main routine or create an ELB.
NOMOD	40	Module %s not found in library The module was not found in the object library.
NONAME	22	No file name specified with OUTPUT You've used a linker option on the command line that requires a filename (such as the extract, map file, library file, or output file option), but no filename was specified.
NOTELB	35	Invalid ELB format in file: %s The specified ELB has an invalid internal format. Re-create the ELB.
ONEPRI	24	Second main-line or primary module illegal: %s You've specified more than one main or primary routine.
OPFNF	26	Cannot open input file: %s The specified file could not be opened for input. Check whether the file exists, and if it does, check the protections on that file and its directories.
OPOUT	27	Cannot open output file: %s The specified file could not be opened for output. Check whether the file exists and can be replaced. Check whether the device and directory exist.

## Error Messages

### Linker Errors

OPPNDN	48	Module %s built opposite 'endian' (%s) You've attempted to link objects that have different endian types. Recompile the modules on the same endian machines.
REFER	45	Fatal referencing errors Not all parts of the executable being built are present. Find the missing parts and relink with them.
SUBRLB	29	No main-lines allowed in executable libraries You've attempted to place a main routine into an ELB.
WRterr	30	Out of disk space for output file There is not enough disk space for Synergy DBL to write to the disk file. Remove enough files from the device to provide enough space for the file being created.
XCLREF	31	Too many subroutines referenced from ELB module You've exceeded the maximum number of subroutines that can be referenced from an ELB module.
XUNDEF	32	XCALL routine undefined An XCALL in the object code was unresolved by the linker.

## Informational error messages

The following errors provide additional information about other errors.

FRSTDEF	71	Symbol %s first defined in %s Specifies where the first occurrence of the duplicate symbol was found.
FRSTRTN	72	Routine %s first found in %s Specifies where the first occurrence of the duplicate routine was found.

## Warning error messages

CLSDUP	78	Duplicate class: %s in ELB %s The specified class in the specified ELB has been defined more than once, and the sizes of the duplicates are identical.
CMNGBL	50	COMMON '%s' conflicts with GLOBAL DATA SECTION in module %s A common and a global data section have the same name.



COMNDUP	73	Duplicate Global Common: %s in ELB %s The specified global common in the specified ELB has been defined more than once, and the sizes of the duplicates are identical.
COMWAR	5	Compile warnings in module: %s The module had warnings when it was compiled. Remove the warnings and recompile the module.
DUPRTN	69	Duplicate routine: %s in %s The specified routine has been defined more than once in the specified file.
ELBSUB	46	In subroutine '%s', ELB '%s' This informational message displays where the undefined global data reference was specified.
GBLDUP	77	Duplicate global: %s in ELB %s The specified global variable in the specified ELB has been defined more than once, and the sizes of the duplicates are identical.
GDS DUP	75	Duplicate Global Data Section: %s in ELB %s The specified global data section in the specified ELB has been defined more than once, and the sizes of the duplicates are identical.
LITDUP	74	Duplicate Global Literal: %s in ELB %s The specified global literal in the specified ELB has been defined more than once, and the sizes of the duplicates are identical.
SRECDUP	76	Duplicate Static Record: %s in ELB %s The specified static record in the specified ELB has been defined more than once, and the sizes of the duplicates are identical.
NOTINI	41	Global data section '%s' not initialized In traditional Synergy, the set of references to the specified global data section does not contain the INIT option. Add “,INIT” to exactly one instance of that global data section.
ONEINI	23	Global data section %s has duplicate , INIT More than one of the specified global data sections contains the INIT option. Remove all but one “,INIT” from the specified global data sections.

## Error Messages

### Linker Errors

REFBIG	84	Global data reference larger than definition ‘%s’ A global data section reference is larger than the same named global data section with “,INIT”. (This could happen, for example, if an .INCLUDEd file was changed after one module was compiled.) When this occurs, the linker increases the size of the global data section to match the larger of the two, but the data initialization stays the same as the one with the ,INIT. This means the increased size has random data in it, and you cannot assume that it is initialized to anything or that the initialization data is aligned correctly for the larger definition.
SUNDEF	47	COMMON symbol ‘%s’ undefined in module %s The specified symbol was not present to build the executable routine properly. Find the missing parts and relink with them.

# Librarian Errors

## Fatal error messages

The following errors cause the librarian to abort.

BADFIL	1	Bad record format in file: %s The object module is invalid and the file has been truncated. Recompile the module.
BITOPP	39	File %s built with opposite bit size An attempt was made to load an ELB built with the wrong architecture size (32- or 64-bit) for this librarian.
CLOSIN	2	Cannot close %s An internal error occurred when closing a file.
CMDBIG	3	Command line too long: %s The command line was too long. Use continuation lines.
CONOPT	4	Conflicting options on command line The Delete or Extract librarian option was specified on the same command line as the Add or Replace option.
DNXSTMD	5	Cannot delete non-existing module: %s The specified module did not exist and therefore could not be deleted.
EXSTMOD	6	Cannot add existing module: %s You've attempted to add a module that already exists. Replace the module instead of adding it.
INSMEM	9	Insufficient memory for attempted operation The program could not allocate enough memory for the library to perform its function. Make more memory available for the librarian to use.
INTERR	517	Internal Error: %d This error is generated for any of the following reasons: <ul style="list-style-type: none"><li>▶ 40201 Invalid object record position</li><li>▶ 40202 Empty OLB</li><li>▶ 40203 Invalid object record</li></ul>

- ▶ 40204 Invalid object record position extracting
- ▶ 40205 No ENDMOD for BEGMOD object record
- ▶ 40301 Cache initialization error
- ▶ 40302 Cannot unlink main OLB
- ▶ 40303 Error reading file header
- ▶ 40304 Cannot rename temp to main OLB
- ▶ 40305 Error reading OLB
- ▶ 40306 Error creating temporary file
- ▶ 40403 Error accessing error file (W)
- ▶ 40404 Error accessing error file (E)

INVOPT	11	Invalid command line option	You've specified an invalid librarian command line option. See <a href="#">chapter 1, "Building and Running Synergy Applications,"</a> for a list of available librarian options.
MAXIF	14	Exceeded maximum input file limit	You've input command lines to the librarian by entering them indirectly through a command file, but the depth of indirect command files specified was too deep.
MAXMOD	16	Exceeded maximum number of module names	The number of modules exceeded 256. Reduce the number of modules.
MAXTF	17	Exceeded maximum number of files	The number of input files exceeded 256. Reduce the number of input files.
MODXP	18	Module name expected	You've used the Delete command line option without specifying the name of the module to delete.
NDNBAD	38	Unrecognized file ID in %s	The specified file contains an unidentified magic number. This error occurs, for example, if a non-ELB file is specified where a valid ELB should be.
NDNOPP	37	File %s built with opposite 'endian'	An attempt was made to load an ELB built with the wrong byte order for this machine.

NONAME	19	No name specified for command line option You've used a command line option that requires a filename (such as Add, Create, or Replace), but no filename was specified.
NOTSUBR	20	Module %s is not a subroutine The module is not a subroutine or function. Specify a subroutine or function.
OBJXP	21	Object file expected: %s An object file was expected but not found.
OLBXP	22	Object library expected: %s An object library was expected but not found.
OPENIN	23	Cannot open input file: %s The input file does not exist or cannot be opened. Check whether the file exists, as well as its protections.
OPOUT	24	Cannot open output file: %s The specified file could not be opened for output. Check whether the file exists and can be replaced. Also check whether the device and directory exist.
OPTXP	28	Option expected One of the following librarian command line options was not specified: Add, Replace, Delete, or Extract.
WRterr	30	Out of disk space for output file There is not enough disk space for Synergy DBL to write to the disk file. Remove enough files from the device to provide enough space for the file being created.
XNXSTMD	26	Extracting non-existing module: %s You've attempted to extract a module that doesn't exist.

## Warning error messages

RNXSTMD	25	Warning - Replacing non-existing module: %s A nonexistent module was added when the replace librarian option was used.
---------	----	---

## Synergy DBMS Errors

The following list maps the Synergy DBMS error numbers to their message text. If you run **isutl** with messaging enabled, the text specified here will be displayed. If you run it without messaging enabled (**-m0**), only the number will be displayed.

1	Bad ISAM file control
2	Specified key out of range
3	Lock failure
4	Filename length too long
5	EOF encountered
6	Index incongruity error
7	Illegal decimal key of reference
8	Illegal alpha key of reference
9	Invalid OPEN mode, Requires update
10	Invalid RFA
11	I/O error
12	Illegal record size
13	Key not same
14	No current record established
15	No duplicates allowed
16	I/O error: No disk space
17	Not an ISAM file
18	Record not locked for WRITE/DELETE
19	Cannot open data file
20	Cannot open index file
21	Qualifier incongruity error
22	I/O error: Read failure
23	Record is locked
24	Input size exceeds destination size
25	I/O error: Write failure
26	Data incongruity, key to deleted rec
27	Data compression/uncompression error
28	Data freelist error
29	Deleted record error
30	Cannot create file
31	Insufficient memory for attempted op
32	Invalid option
33	Invalid compression option

34	Invalid key length
35	Invalid record length
36	Invalid start position
37	Missing required parameter
38	Mismatched segments
39	Key spans end of record
40	Existing file, cannot overwrite
41	Undefined keys, cannot create
42	Flush error
43	Encountered incompatible ISAM file
44	Record not found (No record Read/Found)
45	Invalid null value
47	File in use by another user
48	C-ISAM file corrupted
49	Too many open files
50	Cannot open C-ISAM file
51	C-ISAM read error
52	Cannot open BTRIEVE file
53	BTRIEVE file is corrupted
54	BTRIEVE requester not loaded
55	No privilege to this file or directory
56	Generic Btrieve Error
57	File not found
58	Bad file specification
59	Invalid I/O mode on open
60	Bad file org on open
61	Bad I/O options in I/O statement
62	Operation timed out
63	Illegal function for this control
64	Remote LPQUE error
65	No caching allowed
66	Bad decimal key value
67	Partial numeric key not allowed
68	Invalid overlay of numeric key
70	Invalid index page size
71	Invalid index density
72	Invalid key type
73	Invalid key order

## Error Messages

### Synergy DBMS Errors

74	Incorrect number of types specified
75	Incorrect number of orders specified
76	Invalid non-key integer size
77	Non-key integer data cannot overlap
78	Invalid overlay of non-key integer data
79	Specified segment out of range
80	Null value doesn't exist for key
81	Error in XDL file
82	Error in XDL string
83	Interrupt detected
100	Illegal record number
101	No room to write to file
102	Invalid relative record
103	Cannot delete file
104	Device not available
105	No FDL file allowed on open yet
106	Server license limit reached
107	No winsock
108	TCP/IP init error
109	TCP/IP bad remote user name
110	Cannot connect to port
111	Cannot create client connection
112	bad host name
113	Network problem
114	Server not running on remote host
115	Deadlock condition detected
116	Licensing error, see log for details
117	Licensing timed out
118	Network error after open
119	Version of xfServer not compatible
120	Sort failure
121	Merge failure
122	Sort work file in use
123	Error writing to exception file
124	Bad data segment - correctable
125	Bad data segment - non-correctable
126	Index error
127	Data error



128	ISAM Utility failure
140	Bad allocation pointer - Only read-only access allowed
141	Operation ignored
142	Incompatible ISAM file - unsupported by this Synergy version
143	Invalid timestamp key exceeds current time
144	Network share is not allowed with this file
145	Specified file revision not supported (ISAMC_REV)
146	File access is restricted to read-only

## List of Runtime Error Numbers

The table below lists all runtime errors, sorted by error number, as they are listed in the message file **syntxt.ism**. The Type column indicates one of the following: trappable (E), fatal (F), informational (I), success (S), warning (W).

Number	Mnemonic	Type	Message
00001	EOF	E	End of file
00002	NOCALL	F	Return with no CALL or XCALL
00005	RECEXTCAL	E	Recursive XCALL or .NET CALL exceeds 64
00006	WROARG	E	Incorrect number of subroutine arguments
00007	SUBSCR	E	Invalid subscript specified
00008	WRTLIT	E	Writing to a literal or missing argument
00009	NOMEM	E	Not enough memory for desired operation
00010	ILLCHN	E	Illegal channel number specified
00011	NOOPEN	E	Channel has not been opened
00012	ONLYWR	E	Attempt to open output device in input mode
00013	BACKPEND	E	Backup mode is On
00014	BIGALPHA	E	Alpha temporary result exceeds 65535
00015	BIGNUM	E	Arithmetic operand exceeds maximum size
00016	CHNUSE	E	Channel is in use
00017	FILSPC	E	Bad filename
00018	FNF	E	File not found
00019	NOTAVL	E	Device not available
00020	DIGIT	E	Bad digit encountered
00021	FILOPT	E	Invalid operation for file type
00022	IOFAIL	E	Failure during I/O operation
00023	TOOBIG	E	Input data size exceeds destination size
00024	NOSPAC	E	No space exists for file on device

Number	Mnemonic	Type	Message
00025	FILFUL	E	Output file is full
00027	UPDNFD	E	Update of non-file device
00028	RECNUM	E	Illegal record number specified
00029	BADCMP	F	Compile not compatible with execution system
00030	DIVIDE	E	Attempt to divide by zero
00031	ARGSIZ	E	Argument specified with wrong size
00032	REPLAC	E	Cannot supersede existing file
00033	CHNEXC	E	Too many files open
00037	DEVUSE	E	Device in use
00038	FINUSE	E	File in use by another user
00039	OUTRDO	E	Output to read-only device
00040	LOCKED	E	Record is locked
00041	BACKUPMODE	E	Backup mode error
00044	ELBREF	W	Undefined global data reference '%s'
00052	BADKEY	E	Illegal key specified
00053	KEYNOT	E	Key not same
00054	NODUPS	E	Duplicate key specified
00056	NOTISM	E	Not an ISAM file
00061	NOCURR	E	No current record
00062	PROTEC	E	Protection violation
00064	RNF	E	Record not found
00067	VMSError	F	Unexpected VMS system error
00077	ARGORD	E	Arguments out of order for PAK or UNPAK
00078	ARGREC	E	PAK/UNPAK fields not in record
00080	NOSQL	E	SQL Connection installation error or DBLOPT 48 not set

## Error Messages

### List of Runtime Error Numbers

Number	Mnemonic	Type	Message
00082	AORDXP	E	Alpha or decimal variable expected
00086	RECLNG	E	Invalid record length
00087	ARGMIS	E	Argument missing
00095	OPNERR	E	OPEN error
00098	INTRPT	E	Interrupt character detected
00100	RMSERROR	E	Unexpected RMS error
00101	NOCHAIN	E	Stop chain not allowed in callback on Windows 7 or Windows X64
00102	RUNERR	F	Internal runtime failure: %s
00103	FILORG	E	Invalid file organization
00104	OUTRNG	E	Value out of range
00106	EXQUOTA	E	Exceeded quota
00107	DEVNOTRDY	E	Device not ready
00108	IOMODE	E	Bad mode specified
00111	TIMOUT	E	Terminal input operation timeout
00115	BLKSIZ	E	Invalid value specified for BLKSIZ
00120	EXCACT	E	Too many activation characters
00122	QUEUENOTAV	E	Invalid queue specified on LPQUE
00128	VMSRMS	F	Unexpected VMS or RMS error
00131	SMERR	E	SORT or MERGE error
00132	SAMOP	E	Operands must be both alpha or both numeric
00141	MAXIF	E	Too many input files open
00144	ARGDIGPT	E	Numeric digit(s) and at most one decimal point expected
00145	BDIGXP	E	Binary digits expected in argument (%s)
00146	HDIGXP	E	Hexadecimal digits expected in argument (%s)

Number	Mnemonic	Type	Message
00147	ODIGXP	E	Octal digits expected in argument (%s)
00151	ARGDIG	E	Numeric digit(s) expected in argument
00153	DECXPT	E	Decimal expected
00154	IORDXP	E	Only integer and decimal operands allowed
00155	RNDVAL	E	Invalid round value: %d
00156	IRNDVAL	E	Invalid round value for integer operand: %d
00157	INVFORENT	E	Invalid entry to FOR loop
00158	IDXP	E	Implied data type required
00159	INVHDL	E	Invalid memory handle
00160	ADDRSIZ	E	Invalid address size
00161	HSIZERR	E	Map outside bounds of field or handle
00162	NOMETHOD	E	Method's routine not found
00166	NUMXP	E	Numeric argument expected
00216	FLSPCW	I	File specification was %s
00223	INVDIM	E	Invalid number of dimensions
00224	INVPRC	E	Invalid fractional precision
00226	MRGERR	E	Merge error
00243	SRTFAI	E	SORT failure
00254	NOXCAL	E	Undefined XCALL referenced
00256	LPQERR	E	LPQUE failed
00301	RECBLK	E	Record must be a multiple of block size
00302	INTISM	E	Internal Isam error - File may be corrupt
00303	INTLCK	E	Unexpected system locking error
00308	MAXPRC	E	Too many processes
00309	INVACT	E	Invalid action for XCALL FATAL

## Error Messages

### List of Runtime Error Numbers

Number	Mnemonic	Type	Message
00311	NOFORK	E	Cannot fork
00313	RELREC	E	Invalid relative record
00316	IRCSIZ	E	Invalid record size
00317	INVALRFA	E	Invalid record's file address
00318	DELREC	E	Deleted record
00319	CLNTERR	E	Client server error, host: %s
00320	NETPROB	E	Network problem reaching server %s
00321	NOSERVER	E	Synergy server on %s is not running or has been shut down
00322	NULARG	E	Improper use of null argument
00323	SETTYP	E	SET data types must be the same
00324	MSGFAIL	E	SEND/RECV message failure
00325	BADHOST	E	Unknown host "%s" in server spec
00326	BADUSER	E	Bad username, login rejected on %s
00327	UNDEFERR	E	Undefined error
00329	WNDERR	E	Window Manager error
00330	LIBMAX	E	Maximum open libraries exceeded
00331	NETCONFIG	E	Local network configuration error
00332	SQLERR	E	SQL Connection error
00333	NOMORECURS	E	SQL: No more available open cursors
00334	CURSERR	E	SQL: Error on cursor
00335	BADDATATYP	E	SQL: Invalid data type for this operation
00336	SQLSTACK	E	SQL: Stack variable still bound/defined on routine exit
00337	SQLDYN	E	SQL: ^M variable still bound/defined on dynamic memory deletion

Number	Mnemonic	Type	Message
00338	RCBDYN	E	RCB: ^M variable still bound on dynamic memory deletion%s
00339	RCBSTACK	E	RCB: Stack variable still bound/defined on routine exit%s
00340	INVKVAL	E	Invalid key value
00341	INVPKEY	E	Invalid partial key
00342	BADXDLF	E	Bad XDL file
00343	BADXDLS	E	Bad XDL string
00400	MSGNOTFND	F	Error message number %d not found or internal failure
00417	INVEXFTYP	E	Invalid external function data type
00420	INVARG	E	Invalid argument
00421	AXERR	E	Error while processing an ActiveX control
00422	AXNOLOAD	E	Could not load ActiveX control
00423	AXNOSUB	E	Could not find subroutine or function
00424	AXUNSUP	E	Unsupported feature
00425	AXNOFIND	E	ActiveX parameter not found
00426	MISSMETH	E	Method/Delegate not found
00427	MISSFLD	E	Field/Type/Property/Event not found
00428	NOLOAD	E	Could not load assembly
00429	NULLREF	E	Invalid use of NULL object
00430	NODOTNET	E	Could not load the .NET CLR
00431	RECNOT	E	Record not same
00432	NETCRYPT	E	File requires network encryption
00433	DATACRYPT	E	Error encrypting/decrypting data field: %s
00434	SALTIV	E	Error getting salt/initialization vector: %s
00500	INVFATERR	F	Invalid fatal error number for XCALL FATAL

## Error Messages

### List of Runtime Error Numbers

Number	Mnemonic	Type	Message
00503	NOTDBR	F	%s is not a DBR file
00506	STKOVF	F	Runtime stack overflow
00507	UNSUP	F	Unsupported command
00508	SIGNAL	F	Signal trap
00509	OPENF	F	Cannot open %s
00510	STPMMSG	S	STOP
00511	RTNNF	E	Cannot access external routine %s
00512	GBLNF	F	Cannot access named global %s
00513	BADSYS	F	License management problem
00514	LMFAIL	F	Licensing failure
00515	CMDBIG	F	Command line too long
00516	INVOPT	F	Invalid option
00517	INTERR	F	Internal failure: %d
00519	ALPHARG	E	Alpha argument required
00520	WRTERR	F	write failure
00521	INTARG	E	Integer argument required
00522	AORIARG	E	Integer or alpha argument required
00523	FNOTFOUND	E	Function not found
00525	BADFORMAT	E	Bad format string
00526	BADHANDLE	E	Bad DLL handle
00527	INVDATE	E	Invalid date
00528	DLLOPNERR	E	DLL could not be opened: %s
00529	DLLCLSERR	E	DLL could not be closed
00530	PURGE	E	DCL purge error
00531	BADADDR	E	Bad address detected: %s



Number	Mnemonic	Type	Message
00532	BADELB	E	Bad ELB detected: %s
00533	NOFDL	E	Invalid open mode for FDL usage
00534	SRVRLICNS	E	Server license limit reached on %s
00535	DEADLOCK	E	Operation would cause deadlock
00536	SRVLICERR	E	Licensing error on server %s
00537	SRVLICTIMO	E	Licensing timed out on server %s
00538	WINRSRC	E	Windows resource exhausted
00539	BADFONTNAM	E	Invalid font name specified: %s
00540	DUPFONTNAM	E	Duplicate font name specified: %s
00541	FONTINUSE	E	Font %d in use, cannot delete
00542	BADFONTID	E	Invalid font ID specified: %d
00543	WNFNCERR	E	Windows API function failure: %s
00544	INVRPTHND	E	Invalid report handle
00545	INVPNHAND	E	Invalid pen handle
00546	INVCLLSEQ	E	Invalid calling sequence
00547	OPTINV	E	Invalid option
00548	DLLOPNMOD	E	Associated DLL not in path or not found
00549	BADWNDID	E	Window %d bad or no longer open
00550	XFBADPKTID	E	Incorrect packet identifier
00551	XFBADMTHID	E	Method ID too long
00552	XFNUMPARMS	E	Invalid parameter count
00553	XFBADPKT	E	Packet format error
00554	XFBADTYPE	E	Invalid parameter type
00555	XFREQPARM	E	Required parameter not sent
00556	XFBADARRAY	E	Error mapping array element

## Error Messages

### List of Runtime Error Numbers

Number	Mnemonic	Type	Message
00557	XFIOERR	E	File I/O error occurred on server
00558	XFMETHKNF	E	Method key not found
00559	XFRTNNF	E	Cannot access remote routine
00560	XFNOCONN	E	No connection to remote host
00561	XFHALT	E	Fatal error occurred on server
00562	XFNOINIT	E	RX_DEBUG_START called without RX_DEBUG_INIT
00563	SRVNOTSUP	E	Unsupported server version/Feature not available
00564	XFUNKERR	E	Unknown error reported by xfServerPlus
00565	XFNOCDT	E	Unable to open method catalog file
00566	XFNOCMPDT	E	Unable to open method parameter file
00567	XFNOELB	E	Unable to open ELB file.
00568	INVDSCR	E	Invalid descriptor
00569	SYNSOCK	E	Synsock error %d
00570	INVRCBHND	E	Invalid RCB handle
00571	INVNETHND	E	Invalid network handle
00572	INVWNDHND	E	Invalid window handle
00573	INVNAMHND	E	Invalid namespace handle
00574	INVCLSHND	E	Invalid class handle
00576	PRTOBJHND	E	Protected object handle cannot be deleted
00580	NOTRCBHND	E	Handle is not an RCB handle
00581	NOTNETHND	E	Handle is not a network handle
00582	NOTWNDHND	E	Handle is not a window handle
00583	NOTNAMHND	E	Handle is not a namespace handle
00584	NOTCLSHND	E	Handle is not a class handle
00588	NOTRPTHND	E	Handle is not a report handle

Number	Mnemonic	Type	Message
00589	NOTPNHAND	E	Handle is not a pen handle
00590	EXECF1	E	Cannot execute: %s
00591	SDMS	E	ISINFO error
00592	XFINCALL	E	Remote call already in progress
00593	XFNOCALL	E	No current call in progress
00594	OLDDBR	F	Old DBR file format%s detected: relink %s
00595	OLDELB	E	Old ELB file format%s detected: relink %s
00596	XFMETHCRYPT	E	Method requires encryption
00597	XFSERVNOSEC	E	Encryption not enabled on server
00598	NDNOPP	E	File built with opposite 'endian': %s
00599	BITOPP	E	File built with opposite bit size: %s
00600	INCPTCLS	E	Incompatible classes
00601	NOOBJ	E	No object for handle
00602	NOTOHNHND	E	Both source and destination must be object handles
00603	CLSMTCH	E	Class mismatch between routines
00604	OHNDREQ	E	Object handle required
00605	IDPARMREQ	E	Implied-decimal parameter required
00606	OHNDCPY	E	Invalid copy of an object handle
00607	NODBGPORT	E	Debug port number not specified: %s
00608	BADDBGPORT	E	Invalid debug port number: %s. Must be an integer within the range 1024 to 65535, inclusive
00609	BADDBGTMOT	E	Invalid remote debug timeout value: %s
00610	DBGNOSOCK	E	Unable to attach to remote debug port
00611	DBGNOCONN	E	No debug client connection was established
00612	DBGSOCKER	E	Remote debug socket error; continuing without debug

## Error Messages

### List of Runtime Error Numbers

Number	Mnemonic	Type	Message
00613	DBGCLOSED	E	Remote debug client closed the connection; continuing without debug
00614	TOOLKIT	E	Toolkit error
00615	SEQRDS	E	Sequential read caching error
00616	EXCEPT	E	Exception of type '%s'
00617	INVCAST	E	Invalid cast operation
00618	SINGLEDIM	E	Array is not a one-dimensional array
00619	ARRAYBND	E	Index is outside the bounds of the array
00620	DIFDIMS	E	Arrays must have the same number of dimensions
00621	UNHANDLED	E	Unhandled exception: %s
00622	NORETURN	E	Leaving local scope where a CALLED subroutine is still active
00623	OBJPASSED	E	Unexpected object handle passed as argument
00624	STRMTCH	E	Structure mismatch between routines
00625	HNDCORUPT	E	Handle has been modified; possible subscripting violation
00626	ALCOMPAT	E	ArrayList compatibility issue. See the 9.1.5 release notes
00627	INVOPER	E	Invalid operation: %s
00628	HKNOOPS	E	No I/O hook operations specified; at least one is required
00629	HKNOTACT	E	I/O hooks no longer active on channel
00630	HKNOTIMP	E	Referenced I/O hook operation(s) not implemented: %s
00631	HKACTIVE	E	Attempted I/O on hooked channel from within hook routine
00632	COMPATISAM	E	Incompatible ISAM file - unsupported by this Synergy version

Number	Mnemonic	Type	Message
00633	INVATIME	E	Invalid timestamp key exceeds current time
00634	NONETSHR	E	Network share is not allowed with this file
00635	REVUNSUP	E	Specified file revision not supported (ISAM_REV)
00636	SRVREXP	E	Server session has expired or has been terminated.
00637	SRVCONRTY	E	Server connection retry failure
00638	SIZLIMIT	E	File size limit exceeded
00639	RECLIMIT	E	File record limit exceeded
00641	GBLSIZ	F	Reference exceeds allocated size of global data: %s
00642	JOINKEYREQ	E	Inner table requires key reference: %s
00643	JOINOUTREF	E	Outer table reference required for inner table: %s
00644	JOINOPER	E	Invalid Join predicate operator
00645	JOINONREQ	E	Valid On expression required
00646	JOINISMREQ	E	ISAM file required for inner table: %s
00647	JOINENUM	E	Invalid enumerator type for Select object with Join
00701	WRONGTHREAD	E	.NET xfServer client I/O on wrong thread
00900	CATCH	E	(internal use only)
00901	SQLOBJ	F	SQL: Class variable still bound/defined on destruction
00902	SQLREL	F	SQL: Variable bound/defined to local record on memory reclaim Routine: %s
00903	RCBREL	F	RCB: Variable bound to local record on memory reclaim Routine: %s
00904	RCBCALL	E	RCB: Call in progress
00905	RCBOBJ	E	RCB: Class variable still bound/defined on destruction%s
01001	ACCVIO	I	Access violation
01002	ALITXP	I	Alpha literal expected

## Error Messages

### List of Runtime Error Numbers

Number	Mnemonic	Type	Message
01003	ARGWAS	I	Argument number was %d
01011	BADIND	I	Bad index: %d
01012	BADRNG	I	Bad range value: %d,%d
01013	BADRNGR	I	Bad range value: %d:%d
01021	CHNWAS	I	Channel specified: %u
01022	CHRSPC	I	Character specified: %s
01029	COLEQL	I	Colon or equal sign expected
01030	CREFIL	I	Error creating file
01040	DBLDIR	I	DBLDIR not set
01041	DCMPER	I	Data compression/uncompression error
01042	DECXP	I	Decimal expected
01043	DELFIL	I	Error deleting file
01046	DINCON	I	Data incongruity
01047	DRCSIZ	I	Destination record size: %d
01052	ERTEXT	I	%s
01053	ERTXT2	I	%s %s
01054	ERTXTN	I	%s %d
01055	EQLEXP	I	Equal sign expected
01056	EXECF	I	Cannot execute: %s
01061	FILWAS	I	File specification was %s
01063	FRCSIZ	I	File record size: %d
01070	IINCON	I	Index incongruity
01076	INVCMD	I	Invalid I/O command: %s
01077	OPTWAS	I	Invalid option: %s
01078	INVSU	I	Invalid switch

Number	Mnemonic	Type	Message
01079	INVSMD	I	Invalid OPEN submode
01080	INVVAL	I	Invalid value for %s
01084	IOOPN	I	Cannot open %s
01088	IOERR2	I	Channel %d, open mode %s
01101	KEYSPC	I	Could not locate key with identifier %s
01120	MAXSIZ	I	Maximum record size is %u
01132	NOEOFC	I	No EOF character found. Physical EOF was used
01134	NUMSPC	I	Number specified: %ld
01140	OPNFIL	I	Cannot open file
01142	OPTSPC	I	Option specified %s
01143	OPWRED	I	Operation was READ
01144	OPWFND	I	Operation was FIND
01145	OPWRDS	I	Operation was READS
01146	OPWDEL	I	Operation was DELETE
01147	OPWSTO	I	Operation was STORE
01148	OPWWRI	I	Operation was WRITE
01149	OPWCRE	I	Operation was ISAMC
01160	RBKXP	I	Right bracket expected
01162	READER	I	Cannot read input file
01163	NUMWAS	I	Record number: %ld
01164	RECWAS	I	Record size specified: %u
01166	RENFIL	I	Error renaming file
01168	RORKXP	I	Record or key expected
01169	RPEXP	I	Right parenthesis expected
01185	TTSBMD	I	Submode ignored for terminal open

## Error Messages

### List of Runtime Error Numbers

Number	Mnemonic	Type	Message
01191	VALSPC	I	Value specified is %ld
01192	VALRNG	I	Value range is %d to %d
01193	ATLIN	I	At line %d in routine %s
01194	CALFRO	I	Called from line %d
01195	ATLINE	I	At line %s in routine %s
01196	CALFRM	I	Called from line %s
01205	WRTFIL	I	Cannot write to file
01207	MSGBIG	I	Message exceeds maximum size
01208	MSGEXP	I	Message communication timeout
01209	SYSFLT	I	System fault (%d)
01210	NOLMD	I	Cannot access Synergy License Manager
01211	DEVFUL	I	Device full
01212	INTCON	I	Internal consistency failure
01213	EXPDEMO	I	This system has timed out
01214	EXUSER	I	Exceeded concurrent user capacity
01215	NOTCONF	I	Synergy Runtime license is not configured
01216	CONSUP	I	Please contact your Synergy/DE supplier
01217	BADDSCR	I	Corrupted descriptor: type = %d, class = %d
01218	AMBKWD	I	Ambiguous XDL keyword: %s
01219	MLTKWD	I	Keyword specified multiple times: %s
01220	REQKWD	I	Missing required keyword: %s
01221	NOVAL	I	No value supplied with keyword: %s
01222	INVAVAL	I	Invalid %s value: %s
01223	INVDVAL	I	Invalid %s value: %d
01224	AMBVAL	I	Ambiguous %s value: %s



Number	Mnemonic	Type	Message
01225	KEYSPEC	I	Key specified: %d
01226	INVIVAL	I	Numeric return argument expected
01227	INVBUFF	I	Alpha return argument expected
01228	ERTEXTT	I	%s
01229	DIMSPC	I	Dimension specified: %d
01230	DIMEXP	I	Dimensions of passed argument: %d
01231	STKTRC	I	in %s:line %s



# A

## Traditional Synergy Compiler Listings

The information in this appendix applies to traditional Synergy only. A compiler listing is generated when you compile your program with the list compiler option (-l).

### **Sample Compiler Listing   A-2**

Provides a sample compiler listing and a description of each item in the listing.

### **Compiler Listing Tables   A-5**

Describes the compiler listing tables that can be generated.

# Sample Compiler Listing

```

LIST                               Mon Feb  9 13:00:42 2010   DBL V9 Compiler p001
                                   /usr2/list.dbl

1      ;
2      ; list.dbl
3      ;
4      ; Example of listing
5      ;
6
7      subroutine secnds
8      begint, d
9      endtim, d
10
11     .include "mydata.dbl"
2.1    ; First line of "mydata.dbl"
2.2    ;
2.3    .include "mydata2.dbl"
3.1    ; First line of "mydata2.dbl"
3.2    ;
3.3    .define MAXSEC  ,8000000
3.4    .define MINSEC  ,0
3.5    ; Last line of "mydata2.dbl"
2.4    record
2.5        fld1,  d4
2.6        fld2,  d4
2.7    ; Last line of "mydata.dbl"
12    record
13        curtim  ,d6
14        hr      ,d2 @curtim
15        mi      ,d2 @curtim+2
16        se      ,d2 @curtim+4
17        cursec  ,d5
18
19    proc
20        xcall time (curtim)
21        cursec=(hr*3600)+(mi*60)+se
22        if (cursec .lt. begint)
23            begin
24                cursec = cursec + 86400
25    .ifdef TEST
26        C            if (cursec .gt. MAXSEC)
27        C            begin
28        C                cursec = MAXSEC
29        C            end
30    .endc
31        end

```

```
32             endtim=cursec-begtim
33             return
34         endsubroutine

Errors:          0, in file /usr2/list.dbl
dbl -w 80 -l list list
SYNCMPOPT: -qcheck
DBLOPT: 11
```

## An explanation of the compiler listing

### Header

The listing begins with a page break. The header's first line contains the following information:

- ▶ Routine name (the name of the current routine being compiled, whether it be a subroutine name, function name, or the main routine name; for example, in the sample compiler listing, the routine name is **LIST**).
- ▶ Current date. For example, in the sample compiler listing, the current date is  
Mon Feb 9 13:00:42 2010
- ▶ Compiler header. For example, in the sample compiler listing, the header is  
DBL V9 Compiler
- ▶ Page count. For example, in the sample compiler listing, the page count is  
p001

The second line of the header contains the following information:

- ▶ Title (initialized to blanks; the title is set with the **.TITLE** compiler directive). For example, in the sample compiler listing, the title is initialized to blanks underneath the routine name.
- ▶ Source file path. For example, in the sample compiler listing the source file path is  
**/usr2/list.dbl**

After the two header lines, the compiler generates a blank line.

### Line numbering

A line number is generated for each line in the source file that contains the **PROC** statement. For example, in the sample compiler listing, the first line of the source begins on line number 1, the second begins on line number 2, and so forth.

#### Include files

Source files that are `.INCLUDEd` are generated to the listing file. Each include file has its own set of line numbers. In our sample compiler listing, notice how the line numbers begin with 2.1 after the **mydata.dbl** file is included at line 11 and 3.1 after the **mydata2.dbl** file is included at line 2.3.

An include level counter is incremented each time the compiler accesses an include file and decremented each time the compiler returns from an include file. If this counter is greater than 0, it is displayed to the left of the line numbers in the listing file, as illustrated in the sample compiler listing.

#### Lexical level

A lexical level counter is incremented each time the compiler encounters a `PROC` (or `.PROC`) or `BEGIN` statement. The counter is decremented each time the compiler encounters an `END` (or `.END`) statement. This counter is displayed to the right of the line numbers in the listing file for each `PROC`, `BEGIN`, or `END` statement, as illustrated in lines 19, 23, 31, and 34 of the sample compiler listing above. The lexical level counter is not displayed next to line 27 and 29 because they are enclosed within a false conditional block. See [“False conditionals”](#) below for more details.

#### False conditionals

The letter “C” is displayed to the right of the line numbers of lines within false conditional blocks, as illustrated in lines 26 – 29 of the sample compiler listing. (These lines are not compiled.) If you turned off the printing of false conditionals (using the `NOCOND` option on the `.START` compiler directive, the conditionals compiler option, or the `+NOCOND` compiler list option), lines 25 – 30 would not be generated to the listing file.

#### Footer

A count of the warnings and errors that the compiler encountered is generated at the end of each listing, along with the command line that was specified to generate the listing.

If the compiler encounters any warnings or errors during compilation, those error messages are also generated to the listing file, following the line that caused the warning or error.

#### SYNCMPOPT

If the `SYNCMPOPT` environment variable is set in the environment, its contents are generated to the listing file. This information helps you determine which options were active when compilation occurred.

#### DBLOPT

If the `DBLOPT` environment variable is set in the environment, its contents are generated to the listing file. This information helps you determine which options were active when compilation occurred.

# Compiler Listing Tables

Depending on which .START and/or compiler options you've set, the compiler might generate one or more tables in the compiler listing at the end of each routine. The two listing tables that are currently available are as follows:

- ▶ Symbol table offsets table
- ▶ Memory usage summary table

## Sample listing tables

TABLES    Mon Feb 9 13:01:15 2009    DBL Version 9.1.5a    Compiler    Page: 1  
   /usr2/tables.dbl

```

1      ;
2      ; tables.db1
3      ;
4      ; Shows examples of compiler listing tables
5      ;
6
7      record
8          avar          ,a50
9          group grp     ,[20]a
10         fld1          ,d3
11         fld2          ,a3
12     endgroup
13     dvar              ,d8
14     idvar             ,d8.4
15
16     proc
17         avar = idvar + dvar
18         idvar = grp[4].fld1
19         xcall sub1(4, dvar)
20     end

```

## Symbol Table Offsets

-----

AVAR	0
DVAR	1
FLD1	3 (GRP.)
GRP	4
IDVAR	2

```

21
22     subroutine subl
23     arg_1    ,d

```

**Traditional Synergy Compiler Listings**  
Compiler Listing Tables

```
24      arg_2    ,d
25      record rec
26      avar      ,a8
27      proc
28      avar = arg_1
29      xcall sub2
30      return
31      end
```

Symbol Table Offsets

```
-----
ARG_1                      -1
ARG_2                      -2
AVAR                       0 (REC.)
```

```
32
33      subroutine sub2
34      record
35      dvar      ,d8
36      avar      ,a4
37      proc
38      dvar = avar
39      end
```

Symbol Table Offsets

```
-----
AVAR                      1
DVAR                      0
```

TABLES Mon Feb 9 13:01:15 2009 DBL Version 9.1.5a Compiler Page: 2  
/usr2/tables.dbl

Errors: 0, in file /usr2/tables.dbl  
dbl -il tables\_i tables

TABLES Mon Feb 9 13:01:20 2009 DBL Version 9.1.5a Compiler Page: 1  
/usr2/tables.dbl

```
1      ;
2      ; tables.dbl
3      ;
4      ; Shows examples of compiler listing tables
5      ;
```



```

6
7      record
8          avar          ,a50
9          group grp     , [20]a
10             fld1      ,d3
11             fld2      ,a3
12         endgroup
13         dvar          ,d8
14         idvar         ,d8.4
15
16     proc
17         avar = idvar + dvar
18         idvar = grp[4].fld1
19         xcall sub1(4, dvar)
20     end

```

#### Memory Usage Summary

-----

FXDCTL	=	A0
DATA	=	BC
CODE	=	20
LITERAL	=	4
DESCR	=	40
LINCTL	=	30
ADDR	=	4
FXD4CTL	=	4
STKREC	=	0
DYNCTL	=	0

Total size: 1F8

```

21
22     subroutine sub1
23         arg_1    ,d
24         arg_2    ,d
25     record rec
26         avar          ,a8
27     proc
28         avar = arg_1
29         xcall sub2
30         return
31     end

```

## Traditional Synergy Compiler Listings

### Compiler Listing Tables

#### Memory Usage Summary

-----

FXDCTL	=	A4
DATA	=	8
CODE	=	C
LITERAL	=	0
DESCR	=	8
LINCTL	=	30
ADDR	=	4
FXD4CTL	=	4
STKREC	=	0
DYNCTL	=	0

Total size: F8

TABLES Mon Feb 9 13:01:20 2009 DBL Version 9.1.5a Compiler Page: 2  
/usr2/tables.dbl

```
32
33      subroutine sub2
34      record
35          dvar      ,d8
36          avar      ,a4
37      proc
38          dvar = avar
39      end
```

#### Memory Usage Summary

-----

FXDCTL	=	A0
DATA	=	C
CODE	=	8
LITERAL	=	0
DESCR	=	10
LINCTL	=	2C
ADDR	=	4
FXD4CTL	=	4
STKREC	=	0
DYNCTL	=	0

Total size: F8

Errors: 0, in file /usr2/tables.dbl  
dbl -ml tables\_m tables

## An explanation of the compiler listing table

### Symbol table offsets table

If you set the symbol table offsets option (for example, **-i** on Windows and UNIX) when you invoke the compiler, the compiler generates a list of every symbol referenced in the preceding routine with its offsets into the symbol table. You can use these offsets to reference symbols while debugging a program that was not compiled and linked with the debug option. (If you compile and link with the debug option, you can reference the symbols by their names as opposed to their offsets.)

You can turn the listing of this table on and off with the [NO]OFFSETS option of the .START compiler directive. See [.START](#) in the “Preprocessor and Compiler Directives” chapter of the *Synergy DBL Language Reference Manual* for more information about the .START options.

If a symbol is a member of a group or a named data structure (as “AVAR” is in the second table in [“Sample listing tables” on page A-5](#)), the path to that symbol is also listed in the table. Also notice the “(GRP.)” next to the entry for “FLD1” in the first table.

### Memory usage summary table

The compiler generates a memory usage summary table at the end of each routine when you set the memory compiler option (for example, **-m** on Windows and UNIX). The memory usage summary table lists the size (in bytes) of each program component. The size is represented as a hexadecimal number.

You can turn the listing of this table on and off with the [NO]summary option of the .START compiler directive. See [.START](#) in the “Preprocessor and Compiler Directives” chapter of the *Synergy DBL Language Reference Manual* for more information about the .START options.



# Index

## Symbols

\$ suffix 1-9  
! debugger command 2-54  
@ debugger command 2-53

## Numerics

.257 filename extension 3-80  
32-bit object file creation 1-13  
64-bit object file creation 1-13

## A

-A compiler option 1-9  
-a compiler option 1-8  
ActiveX control  
    registering 4-31  
    testing 4-31, 4-32  
ActiveX Diagnostic utility 4-31  
Advanced Format drive 3-6  
/ALIGN compiler option 1-8  
align data compiler option 1-8  
allocation map 1-43  
allowdup option 1-14  
allowing duplicate structures and routines 1-14  
alpha ISAM key 3-13, 3-94  
alternate  
    IF compiler option 1-8  
    key 3-13, 3-47  
    store compiler option 1-8  
/ALTIF compiler option 1-8  
/ALTSTORE compiler option 1-8  
argument, omitted  
    compiler option 1-18  
    linker option 1-40  
array  
    enforcing bounds checking 1-15  
    size 1-8  
arraylist variable, examining 2-24  
ascending key  
    characteristics 3-18  
    defining 3-46, 3-47

autokeys 3-15, 3-64, 3-78  
%AX\_LOAD routine 4-32  
axutil.exe utility 4-31

## B

-B compiler option 1-9  
-b compiler option 1-8  
backup mode 4-34 to 4-38  
binary data in text file 3-66  
/BIND compiler option 1-8  
binding  
    compiler option 1-8, 1-19  
    converting nonbound programs 1-19  
    primary routine 1-8  
bldism utility 3-34, 3-35, 3-38 to 3-47  
block  
    separator 3-4  
    submode 3-31 to 3-32  
bounds checking 1-9, 1-15, 1-20  
BREAK debugger command 2-13 to 2-17  
breakpoint  
    canceling 2-18, 2-29  
    deleting 2-20  
    displaying 2-42  
    saving 2-37  
    setting 2-13 to 2-17  
building  
    shared image 1-36  
    Synergy program 1-6, 1-37

## C

-C compiler option 1-9  
-c compiler option 1-10  
caching ISAM files 3-11  
CANCEL debugger command 2-18  
canceling watchpoints and breakpoints 2-18  
case sensitivity  
    compiler options 1-17  
    ISAM keys 3-13, 3-94  
CASE statement 1-12

- C-f-h-m compiler option 1-15
- change tracking 3-10 to 3-11, 3-50 to 3-55
- channel, displaying open 2-42
- /CHECK compiler option 1-9
- chklock utility 3-34, 3-48 to 3-49
- class
  - displaying information 2-44
  - displaying instantiated 2-42
  - generating for .NET 4-46 to 4-50
- clearing ISAM file 3-72
- command, executing from debugger 2-54
- command line, displaying
  - compiler 1-23
  - librarian 1-32
  - linker 1-44
- Command Prompt window 1-2
- common
  - compiler option 1-18
  - variable, \$ suffix 1-9
- /COMMON compiler option 1-9, 1-10, 1-11
- COMMON statement
  - treat as external 1-10
  - treat as global 1-11
- comparing database file definitions 3-56 to 3-61
- compiler
  - built-in definitions 1-11, 1-21
  - error message 5-52 to 5-116
  - invoking 1-6
    - methods for 1-2 to 1-3
    - with redirection 1-3
  - recursion 1-13
  - redirecting commands 1-23
- compiler listing
  - conditional blocks in A-4
  - contents of A-3 to A-4
  - controlling 1-7
  - footer 1-11, A-4
  - form feed in 1-11
  - generating 1-11
  - header 1-11, A-3
  - include files in A-4
  - lexical level A-4
  - line numbering A-3
  - page break 1-13, A-3
  - page length 1-13
  - sample A-2
  - table A-5 to A-9
- compiler option 1-6 to 1-18
  - align data 1-8
  - alternate IF 1-8
  - alternate store 1-8
  - array size 1-8
  - bind 1-8
  - bind primary 1-8
  - bounds checking 1-9
  - common suffix 1-9
  - concatenating files 1-9
  - conditionals 1-9
  - debug 1-10, A-9
  - external common 1-10
  - FIND lock 1-10
  - form feed 1-11
  - global common 1-11
  - global definition 1-11
  - header 1-11
  - import directories for prototyping 1-11
  - include file 1-11
  - list 1-11
  - local record 1-11
  - namespaces to import 1-12
  - .NET compiler warnings 1-12
  - no object file 1-12
  - numeric argument 1-12
  - object 1-12
  - omitted argument 1-18
  - optimize 1-13
  - page break 1-13
  - page length 1-13
  - platform 1-13
  - profiling 1-13
  - recursion 1-13
  - refresh 1-13
  - relax strong prototyping and error checking 1-14
  - setting defines 1-10
  - show information 1-15
  - stack record 1-15
  - static record 1-15
  - stream file 1-15
  - strict 1-15
  - trim 1-15
  - truncate 1-15
  - undefined functions 1-15
  - variable usage 1-16
  - variable usage level 1-16
  - variant 1-16

- warnings 1-17
- width 1-17
- compiling 1-6 to 1-24
  - bound program 1-19
  - conditionally based on system 1-21
- COMPRESS option 3-41
- compressing record data 3-9, 3-41, 3-81
- /CONCAT compiler option 1-9
- concatenating source files 1-9
- conditional
  - block, compiler listing A-4
  - compiler option 1-9
  - compiling 1-21
  - excluding false 1-9
- connection manager 4-15
- continuation character 1-23
- Control Panel. *See* Synergy UI Toolkit Control Panel
- converting ISAM file 3-71
- corruption, file vs. data 3-19
- counted file 3-7, 3-32, 3-65
  - exception file 3-19
  - isload and 3-74 to 3-75
  - restriction 3-33, 3-65, 3-74
- COUNTED option on isload 3-74
- C-P-h-m compiler option 1-15
- CPU, tracking 4-11
- ctimestamp ISAM key 3-15, 3-94
- ctutil utility 3-34, 3-50 to 3-55
- customizing text messages 4-3 to 4-10

## D

- d compiler option 1-10
- d runtime option 1-50
- data
  - compressing 3-9, 3-41, 3-81
  - corruption 3-19
  - refreshing 1-13
- database file 3-1 to 3-101
  - accessing records 3-2
  - comparing to system catalog or repository 3-56 to 3-61
  - converting from one type to another 3-34, 3-62 to 3-67
  - moving to other systems 3-101
- DBGSRV environment variable 2-41
  - displaying 2-42
  - setting 2-40
- dbl command
  - examples 1-22
  - invoking on Windows 1-2 to 1-3
  - syntax 1-6
- .dbl extension 1-6
- dbl2xml utility 4-51 to 4-52
- DBLCASE environment variable 1-4
- dbl.def file 1-21
- dblibr command
  - examples 1-31
  - invoking on Windows 1-2 to 1-3
  - syntax 1-29 to 1-31
- dblink command
  - examples 1-43
  - invoking on Windows 1-2 to 1-3
  - syntax 1-37 to 1-39
- dblproto 4-39 to 4-43
- DBMS errors 5-128 to 5-131
- .dbo file 1-10
  - created with librarian 1-30
  - object files 1-12
- .dbp extension 4-39
- dbr command
  - invoking on Windows 1-2 to 1-3
  - scheduled task 1-55
  - syntax 1-50
- dbrpriv runtime 1-51
- dbs runtime 1-52 to 1-55
  - limitations 1-55
  - scheduled task 1-55
- dbspriv runtime 1-51, 1-52 to 1-55
- dbssvc runtime 1-52 to 1-55
  - License Manager and 1-54
  - limitations 1-55
- DCL RUN command 1-52
- debug
  - compiler option 1-10, A-9
  - linker option 1-38
  - runtime option 1-50
- debug after error runtime option 1-50
- /DEBUG compiler option 1-10
- \_DEBUG define 1-21, 2-4
- debugger 2-1 to 2-63
  - assigning value to variable 2-22
  - breakpoint
    - canceled 2-18
    - deleting 2-20

- setting 2-13 to 2-17
    - shared image 2-16
  - command 2-12 to 2-54
    - line 2-4
    - recalling and editing 2-12
  - continuing program execution 2-29 to 2-30
  - displaying
    - debug entry 2-48
    - source code 2-32
    - traceback 2-47
  - ELB routines available to 2-35
  - examining variable 2-23 to 2-27
  - example session 2-55 to 2-63
  - executing system command 2-54
  - exiting program
    - with traceback 2-28
    - without traceback 2-36
  - font used in 2-4
  - help 2-4, 2-31
  - indirect command file 2-5, 2-53
  - invoking 1-50, 2-3
  - log file 2-34
  - non-interactive runtime and 1-55
  - options
    - examining 2-42 to 2-45
    - setting 2-40 to 2-41
  - program state information 2-42 to 2-45
  - remote 2-8 to 2-11
    - log file messages 5-48
  - searching for a string 2-39
  - settings, saving 2-37
  - source file
    - moving 2-5
    - path 2-41
  - stepping to next routine 2-46
  - symbolic access table and 1-38
  - UI Toolkit, invoking 2-52
  - updating screen 2-38
  - variable, specifying 2-5
  - Visual Studio .NET 2-1
  - watchpoint
    - canceling 2-18
    - setting 2-49
  - window, size and placement 2-4
  - Windows settings 2-4
  - debugging log messages 5-48
  - /DECARGS compiler option 1-12
  - decimal
    - argument, mapping to numeric 1-12
    - ISAM key 3-13, 3-94
  - decoding profile.dat or lines.dat 4-12
  - /DECSCOPE compiler option 1-8
  - define
    - setting at compile time 1-10
    - system 1-21
  - .DEFINE, clearing 1-14
  - /DEFNS compiler option 1-12
  - DELETE debugger command 2-20
  - deleting watchpoints and breakpoints in debugger 2-20
  - density
    - file 3-8, 3-41, 3-92
    - ISAM key 3-19, 3-45
  - /DENSITY option 3-45
  - DENSITY option 3-41
  - DEPOSIT debugger command 2-22
  - deprecated syntax, allowing 1-14
  - descending key
    - characteristics 3-18
    - defining 3-46, 3-47
  - detached process 1-53
  - diagnostics, ActiveX controls 4-31
  - DIBOL compatibility 1-8
  - /DISWARN compiler option 1-17
  - dlib.lib file 1-41
  - drive, large sector 3-6
  - dtktxt.ddf file 4-7
  - duplicate key
    - characteristics 3-15
    - defining 3-47
  - dynamic memory 2-42
  - dz runtime option 1-50
- E**
- E compiler option 1-13
  - ELB
    - creating
      - from OLB 1-43
      - linker option 1-28, 1-38
    - debugger and 2-35
    - extension, default 1-38, 1-39
    - general information 1-27
    - global data section and 1-28
    - linking 1-40, 1-41
      - with other ELBs 1-27, 1-39, 1-42
    - listing information about 1-47



- making changes to 1-28
- maximum open 1-40
- OLB vs. 1-28, 1-41
- opening at runtime 1-27, 1-40
- unresolved references 1-38
- ELB.OBJ file 1-49
- encryption, network 3-42, 3-79, 3-83, 3-91
- ERASE\_ON\_DELETE option 3-42
- \$ERR\_ prefix 5-3
- ERRCNT error 5-68
- error 5-1 to 5-147
  - fatal 5-3
  - literal 5-3
  - numbers, list of 5-132 to 5-147
  - trappable 2-43, 5-2
- error message
  - adding 4-8
  - compiler 5-52 to 5-116
    - informational 5-115 to 5-116
  - customizing 4-3 to 4-10
  - DBMS 5-128 to 5-131
  - librarian 5-125 to 5-127
    - fatal 5-125 to 5-127
    - warning 5-127
  - linker 5-117 to 5-124
    - fatal 5-117 to 5-122
    - informational 5-122
    - warning 5-122
  - runtime 5-4 to 5-51
    - fatal 5-45 to 5-47
    - informational 5-36 to 5-44
    - success 5-48
    - trappable 5-4 to 5-36
    - warning 5-47
  - windowing system 5-49 to 5-51
- \$ERR\_QUENOTAV, decoding 5-27
- /ERRWARN compiler option 1-17
- EXAMINE debugger command 2-23 to 2-27
- executable
  - file
    - creating 1-37, 1-39, 1-48
    - listing information about 1-45
    - name of 1-39, 1-48
  - program, creating from many routines 1-19
- executing a program 1-50
- EXIT debugger command 2-28

- exiting in debugger
  - with traceback 2-28
  - without traceback 2-36
- expand macro compiler option 1-10
- export compiler option 1-10
- expression, not optimizing 1-12
- external common compiler option 1-10

## F

- F compiler option 1-10
- f compiler option 1-11
- facility code 4-5
- false conditional, excluding from listing 1-9
- fast-loading ISAM files 3-65, 3-82
- fatal error 5-3
  - librarian 5-125 to 5-127
  - linker 5-117 to 5-122
  - runtime 5-45 to 5-47
- fcompare utility 3-34, 3-56 to 3-61
- fconvert utility 3-19, 3-34, 3-62
  - backups and 4-34
  - ISAM files 3-65
- FDL keyword, correspondence to XDL 3-97 to 3-100
- file
  - binding to executable program 1-8
  - corruption 3-19
  - creating 3-38
  - ISAM 3-2
    - clearing 3-72
    - converting 3-71
    - creating 3-38
    - density 3-8, 3-41, 3-92
    - keyed access 3-2
    - loading 3-72
    - physical representation of 3-7
    - rebuilding 3-79
    - recovering 3-71
    - sequential access 3-2
    - status information 3-87
    - unloading 3-72
    - verifying structure 3-62, 3-76, 3-79
  - linker options 1-48
  - moving between Windows and UNIX 2-5
  - name 1-4, 1-5
  - object, default extension 1-48
  - open status 4-21

- parameter 3-35 to 3-37, 3-68 to 3-70
- type 3-7
  - defining 3-41
  - fixed-length 3-7
  - multiple fixed-length 3-7
  - variable-length 3-7
- FIND statement 1-10
- /FIND\_LOCK compiler option 1-10
- firewall and debugging via Telnet 2-9
- FIXED option on isload 3-74
- fixed-length
  - file 3-7
  - isload option 3-74
- font, debugger 2-4
- footer in compiler listing A-4
- foreign language, translating text into 4-6
- form feed 1-11
- /FULL option 2-42
- function
  - truncating name 1-15
  - undefined 1-15

## G

- G compiler option 1-11
- g compiler option 1-11
- /GBLDEFS compiler option 1-11
- gennet40 utility 1-14, 4-46 to 4-50
- global common compiler option 1-11
- global data 1-45
- GO debugger command 2-29 to 2-30

## H

- h compiler option 1-11
- header
  - compiler listing A-3
  - compiler option 1-11
  - page, excluding from listing 1-11
- HELP debugger command 2-31
- help in debugger 2-4, 2-31
- HTTP, debugging and 2-8

## I

- i compiler option 1-12
- IF statement 1-8
- /IMPDIR compiler option 1-11
- /IMPLICIT compiler option 1-15
- IMPORT statement 1-11
- importing namespaces 1-12
- include file A-4

- include file compiler option 1-11
- index 3-2
  - block 3-4
    - density 3-8, 3-41
    - size 3-9, 3-90
  - diagram 3-3
  - structure 3-2
- information advisor, isutl 3-83
- information line, text of 4-3
- informational error 5-2
  - compiler 5-115 to 5-116
  - linker 5-122
  - runtime 5-36 to 5-44
- input file, linker 1-40, 1-48
- integer
  - data
    - moving between machines 3-101
    - moving to or from RMS 3-101
    - portability of 3-12
  - ISAM key 3-14, 3-94
  - optimization, relaxing 1-12
- interop option 1-14
- I/O
  - ISAM statement 3-21
  - redirection 1-4
  - terminal 1-55
- ipar utility 3-34, 3-68 to 3-70, 3-86
- irecovr utility 3-34, 3-71
- ISAM 3-2
  - definition language 3-89 to 3-100
    - See also* XDL file
  - file 3-1 to 3-101
    - clearing 3-72
    - converting 3-62 to 3-67, 3-71
    - creating 3-38 to 3-47
    - density 3-8, 3-41, 3-92
    - keyed access 3-2
    - loading 3-65, 3-72, 3-82
    - moving to other systems 3-101
    - physical representation of 3-7
    - rebuilding 3-79
    - recovering 3-71
    - revision 3-5
    - sequential access 3-2
    - status information 3-87
    - structure 3-12
    - type 3-7, 3-41

- unloading 3-72
- verifying structure of 3-62, 3-76, 3-79
- File Maintenance Utility 3-77 to 3-86
- index 3-2 to 3-4
- I/O statement 3-21
- key 3-2, 3-13
  - ascending or descending 3-18, 3-46, 3-47
  - defining 3-44
  - density 3-19, 3-45
  - duplicates 3-15, 3-47
  - modifiable 3-16, 3-44
  - null 3-17, 3-44
  - order 3-45
  - segmented 3-17, 3-44
  - size restriction 3-10
  - type 3-13, 3-44
- patching to a different revision 3-78, 3-82
- routines 3-21
- utilities 3-34
- ISAMC routine 3-34
- ISAMxf. *See* Synergy DBMS
- ISCLR routine 3-34
- ISINFO routine 3-34
- ISKEY routine 3-34
- isload utility 3-34, 3-72 to 3-75
- ismvfy utility 3-34, 3-76
- ISSTS routine 3-34
- isutl utility 3-19, 3-34, 3-77 to 3-86
  - backups and 4-34
  - displaying information and advice 3-83
  - irecovr 3-71
  - ismvfy 3-76
  - temporary files 3-80
  - xfODBC and 3-80

## K

- key
  - alternate 3-13
  - ascending or descending
    - characteristics 3-18
    - defining 3-46, 3-47
  - attributes 3-13
  - defining 3-44
  - density 3-19, 3-45
  - duplicates
    - characteristics 3-15
    - defining 3-47

- modifiable
  - characteristics 3-16
  - defining 3-44
- null
  - characteristics 3-17
  - defining 3-44
- order 3-45
- primary 3-13
- segmented
  - characteristics 3-17
  - defining 3-44
- size restriction 3-10
- type 3-13, 3-44
- value 3-2, 3-4
- keyed access 3-2

## L

- L compiler option 1-13
- l compiler option 1-11
- language, translating 4-6
- large sector drive 3-6
- leaf block 3-4
- length compiler option 1-13
- lexical level counter in compiler listing A-4
- librarian 1-29
  - error message 5-125 to 5-127
  - invoking 1-29 to 1-31
    - examples 1-31
    - methods for 1-2 to 1-3
    - with full command 1-2, 1-3
    - with redirection 1-3, 1-32
  - maximum number of object files 1-30
  - options 1-29 to 1-31
- library 1-26 to 1-32
  - contents 1-31
  - executable
    - creating 1-38
    - listing information about 1-47
  - file, linker option 1-38
  - object, creating 1-29
  - recommendations for using 1-28
- line numbering
  - compiler listing A-3
  - removing from executable image 1-13
- line terminator, preserving 2-5
- lines.dat file 4-12
- LINK command 1-48 to 1-49

## linker

- error message 5-117 to 5-124

## invoking

- examples 1-43, 1-49
- methods for 1-2 to 1-3
- with full command 1-2, 1-3
- with redirection 1-3, 1-44

- omitted argument 1-40

- options 1-37 to 1-39, 1-48

## linking

- creating bound programs 1-19
- executable libraries 1-42
- executable programs 1-41
- object modules 1-37
- on OpenVMS 1-48 to 1-49
- procedure 1-49
- recommendations for 1-41

- .lis file 1-11

## list

- compiler option 1-11
- options 1-7, 1-19

- .LIST compiler directive 1-7, 1-19

- LIST debugger command 2-32

- listdbo utility 1-24

- listdbr utility 1-45

- listelb utility 1-47

- /LISTING compiler option 1-11

- listing file 1-15

- loading ISAM file 3-72

- /LOCAL compiler option 1-11

- local record, default 1-11

## locking a record

- FIND statement 1-10

- information about 3-48 to 3-49

- LOGGING debugger command 2-34

**M**

- macro, expanding 1-10

## map file

- creating 1-38, 1-43

- linker option 1-38

- MASK qualifier 1-55

## memory

- dynamic 2-42

- uninitialized 2-40, 2-43

## message

- creating new 4-7

- customizing 4-3 to 4-10

- command line interface 4-8

- from ASCII file 4-6

- interactively 4-4

- facility code 4-5

- file 4-3

- library, creating 4-7

- modifying 4-3

- printing file 4-3

- translating 4-3

- modifiable key 3-16, 3-44

- /MODIFY option 3-44

- module, descriptor block 1-46

## Monitor utility

- UNIX 4-23 to 4-30

- Windows 4-21 to 4-22

- moving database files 3-101

- multiple fixed-length file 3-7

**N**

- N compiler option 1-12

- n compiler option 1-12

- namespaces to import compiler option 1-12

## .NET

- compiling classes for 1-14

- wrapping classes 4-46 to 4-50

- .NET assembly API 4-49

- /NET compiler option 1-12

- .NET compiler warning 1-12

- NetChange method 3-52

- network encryption 3-42, 3-79, 3-83, 3-91

- NETWORK\_ENCRYPT option 3-42

- no expression optimization compiler option 1-12

- no object file compiler option 1-12

- no output linker option 1-38

- /NOALTIF compiler option 1-8

- /NOARGNOPT compiler option 1-12

- nocase ISAM key 3-13, 3-94

- /NODEBUG compiler option 1-10

- NODEBUG debugger option 2-29

- /NODECARGS compiler option 1-12

- /NOGBLDEFS compiler option 1-11

- .NOLIST compiler directive 1-7, 1-19

- non-interactive runtime 1-52 to 1-55

- debugging and 2-8

- scheduled task 1-55

- /NOOBJECT compiler option 1-12
- /NOOPTIMIZE compiler option 1-13
- /NOREFRESH compiler option 1-13
- NOROLLBACK option 3-42, 3-50, 3-52
- NOTBYTE option 3-43
- /NOWARNINGS compiler option 1-17
- null
  - key 3-17, 3-44
  - trimming trailing 1-15
- /NULL option 3-44
- numeric argument compiler option 1-12

## O

- o compiler option 1-12
- object
  - compiler option 1-12
  - displaying information 2-44
  - file
    - creating 1-6, 1-18
    - default extension 1-48
    - listing contents 1-24
    - name of 1-12, 1-18
    - not generating 1-12
    - order of source files within 1-18
  - module
    - adding to library file 1-29, 1-31
    - deleting from library file 1-29, 1-31
    - extracting from library file 1-30, 1-31
    - file vs. 1-29, 1-30, 1-37
    - getting information 1-30
    - linking 1-37, 1-41
    - replacing 1-30
- /OBJECT compiler option 1-12
- offset table A-9
- /OFFSETS compiler option 1-12
- offsets compiler option 1-12
- OLB 1-26
  - adding object module 1-29, 1-31
  - creating 1-26, 1-29, 1-31
  - deleting object module 1-29, 1-31
  - detailed description during processing 1-30
  - ELB vs. 1-28, 1-43
  - extracting object module 1-30, 1-31
  - generating list of contents 1-30
  - linking 1-26, 1-41
  - making changes to 1-28
  - no warnings 1-30
  - replacing object modules 1-30

- .olb file 1-30
- OPEN statement 3-34
- OPENELB debugger command 2-35
- operating system
  - conditionally compiling for 1-21
  - defining symbols for conditional compiling 1-21
- /OPTIMIZE compiler option 1-13
- /ORDER option 3-45
- output file, linker 1-38

## P

- P compiler option 1-13
- p compiler option 1-8
- page
  - break 1-13
  - header, excluding from listing 1-11
  - ISAM 3-9, 3-90
  - length, in compiler listing 1-13
- /PAGE\_SIZE compiler option 1-13
- param option 1-14
- paramad option 1-14
- parameter file 3-35 to 3-37, 3-68 to 3-70
- paramst option 1-14
- patching an ISAM file 3-78, 3-82
- path, source files in debugger 2-41
- platform compiler option 1-13
- portable
  - integer 3-12, 3-93
  - sequential file 3-75
- primary compiler option 1-8, 1-19
- primary key 3-13, 3-44 to 3-47
- printing message file 4-3
- privileges, elevated 1-51, 1-54
- /PROFILE compiler option 1-13
- profile program 4-12
- profile.dat file 4-12
- profiling routines 1-13, 4-11
  - excluding routines 4-12
  - non-interactive runtime and 1-55
- profline program 4-12
- program
  - binding 1-19
  - building 1-6, 1-37
  - detached 1-53
  - executable, listing information about 1-45
  - running 1-50

- prototyping
  - import directories 1-11
  - Synergy Prototype utility 4-39 to 4-43
- protxt.ddf file 4-7
- psect 1-34

## Q

- qalign compiler option 1-8
- qaltif compiler option 1-8
- qaltstore compiler option 1-8
- qcheck compiler option 1-9
- qconcat compiler option 1-9
- qdebug compiler option 1-10
- qdecargs compiler option 1-12
- qdescscope compiler option 1-8
- qdefine compiler option 1-10
- qdefs compiler option 1-12
- qerrwarn compiler option 1-17
- qexpand compiler option 1-10
- qexport compiler option 1-10
- qexternal compiler option 1-10
- qglobal compiler option 1-11
- qimkdir compiler option 1-11
- qimplicit\_functions compiler option 1-15
- qlocal compiler option 1-11
- qnet compiler option 1-12
- qnoaltif compiler option 1-8
- qnoaltstore compiler option 1-8
- qnoargnopt compiler option 1-12
- qnocheck compiler option 1-9
- qnodebug compiler option 1-10
- qnodecargs compiler option 1-12
- qnodecscope compiler option 1-8
- qnoobject compiler option 1-12
- qnooptimize compiler option 1-13
- qnosuffix compiler option 1-9
- qobject compiler option 1-12
- qoptimize compiler option 1-13
- qprofile compiler option 1-13
- qreentrant compiler option 1-13
- qrefresh compiler option 1-13
- qrelaxed compiler option 1-14
- qreview\_level compiler option 1-16
- qrntcompat compiler option 1-15, 1-20
- qrntcompat linker option 1-20, 1-38
- qstack compiler option 1-15
- qstatic compiler option 1-15
- qstrict compiler option 1-15

- qsuffix compiler option 1-9
- QUERY command 4-25
- QUIT debugger command 2-36
- quitting in debugger
  - with traceback 2-28
  - without traceback 2-36
- quserdef compiler option 1-11
- qvariant compiler option 1-16
- qvar\_review compiler option 1-16

## R

- r compiler option 1-13
- r runtime option 1-50
- rd runtime option 1-50
- rd.log file 2-10, 5-48
- rdltx.ddf file 4-7
- record
  - fixed-length 3-74
  - local 1-11
  - lock
    - default on FIND 1-10
    - getting information about 3-48 to 3-49
    - status 4-21
  - stack compiler option 1-15
  - static 1-15
  - structure 3-12
  - variable-length 3-74
- RECORD\_LIMIT option 3-43, 3-91
- recovering an ISAM file 3-71
- recursive compiler option 1-13
- RECV statement, non-interactive runtime and 1-55
- redirecting
  - compiler commands 1-23
  - librarian and 1-32
  - linker and 1-44
- /REENTRANT compiler option 1-13
- /REFRESH compiler option 1-13
- refresh compiler option 1-13
- registering ActiveX control 4-31
- regsvr32 utility 4-31
- relative file 3-22 to 3-25, 3-65
- relax integer optimization compiler option 1-12
- relax strong prototyping compiler option 1-14
- /RELAXED compiler option 1-14
- remote debug runtime option 1-50
- replacement identifier, clearing 1-14
- Repository, comparing database files to 3-56
- /REVIEW\_LEVEL compiler option 1-16

- revision, ISAM 3-5
- RFA, static 3-43
- rmoncore file 4-25
- RMS
  - DELETE statement and 3-24
  - FDL files and 3-100
  - key name 3-44
  - maximum record size 3-12
  - moving file to other systems 3-101
  - relative file 3-22
  - restriction 3-17
  - sequential file 3-25, 3-26
  - status utility and 3-87
  - usage 3-2
- /RNTCOMPAT compiler option 1-15
- rollback operation 3-10, 3-42, 3-91
- ROLLBACK option 3-42, 3-91
- root block 3-4
- routine
  - binding 1-19
  - compiling 1-6 to 1-24
  - enabling profiling 1-13
  - header in compiler listing A-3
  - linking 1-37
  - listing information about 1-46
  - unresolved names 1-41
- rpstxt.ddf file 4-7
- rptxt.ddf file 4-7
- rsynd daemon 4-23
- Run dialog box 1-3
- running Synergy programs 1-50
- runtime
  - error message 5-4 to 5-51
    - numeric list 5-132 to 5-147
  - invoking
    - methods for 1-2 to 1-3
    - syntax 1-50
    - with full command 1-2, 1-3
  - non-interactive 1-52 to 1-55
  - scheduled task 1-55
  - targeting specific 1-15, 1-20
- runtime compatibility compiler option 1-15, 1-20
- runtime compatibility linker option 1-20, 1-38

## S

- s compiler option 1-8
- SAVE debugger command 2-37
- scheduled task, dbr or dbs as 1-55

- SCREEN debugger command 2-38
- SEARCH debugger command 2-39
- /SEGMENT option 3-44
- segmented key
  - characteristics 3-17
  - defining 3-44
- SEND statement, non-interactive runtime and 1-55
- separator block 3-4
- sequence ISAM key 3-14, 3-94
- sequential
  - access 3-2
  - file 3-25 to 3-27
    - fixed-length records 3-74
    - portability between endian types 3-75
    - variable-length records 3-74
- service runtime 2-9
- servstat 4-15 to 4-20
- SET debugger command 2-40 to 2-41
- SFWINIPATH environment variable 1-55
- SFWUSRINIPATH environment variable 1-55
- shared image
  - building 1-33 to 1-36
  - setting breakpoint in 2-16
- /SHOW compiler option 1-15
  - CONDITIONALS 1-9
  - HEADERS 1-11
  - NEWPAGE 1-11, 1-13
  - NOCONDITIONALS 1-9
  - NOHEADERS 1-11
  - NONEWPAGE 1-11, 1-13
- SHOW debugger command 2-42 to 2-45
- SIZE\_LIMIT option 3-42, 3-91
- snapshots 3-50 to 3-55
- /STACK compiler option 1-15
- stack size
  - default 1-45
  - displaying 2-43
  - linker option 1-39
  - setting 1-39, 1-45
- .START compiler directive 1-19
  - offsets option A-9
  - overriding 1-7
  - summary option A-9
- statement I/O, ISAM 3-21
- static
  - record 1-15
  - RFA 3-43, 3-81
- /STATIC compiler option 1-15

- STATIC\_RFA option 3-43
- status utility 3-34, 3-87 to 3-88
- STEP debugger command 2-43, 2-46
- STOP statement, breaking after 2-40, 2-43
- storing ISAM data 3-12
- /STREAM compiler option 1-15
- stream file 1-15, 3-27 to 3-30
- /STRICT compiler option 1-15
- structure, record 3-12
- subroutine, external
  - ISAM 3-21
  - library 1-41
  - name, truncating 1-15
  - profiling 1-13, 4-11
  - unresolved call to 1-26, 1-27, 1-39
- success message 5-48
- suffix, common variable 1-9
- .sym file 1-38
- symbol controls, outputting 1-10
- symbol table offset 1-12, A-9
- symbolic access
  - file, creating 1-38
  - table 1-10
- synbackup utility 4-34 to 4-38
- synbackup.cfg file 4-36 to 4-37
- synckini utility 4-14
- SYNDEFNS environment variable 1-12
- Synergy DBL
  - database files 3-62
  - programs, creating and running 1-2
- Synergy DBMS 3-1 to 3-101
  - errors 5-128 to 5-131
- Synergy File Compare utility 3-56 to 3-61
- Synergy Prototype utility. *See* prototyping
- Synergy UI Toolkit Control Panel 4-3 to 4-10
  - command line 4-8 to 4-10
  - running 4-4
- Synergy windowing API 1-55
- synergy.ini file 1-55, 4-14
- SYNRTL.OPT file 1-48
- SYNTXT environment variable 4-3
- syntxt.ism file
  - contents 4-3
  - corruption 4-7
  - error mnemonics and numbers 5-132
  - modifying 4-4
- SYNUSERDEF environment variable 1-11
- synuser.ini file 1-55, 4-14

- synxfmon.exe utility 4-21
- system catalog, comparing database files to 3-56
- system defines 1-21
- system option
  - #34
    - compiler and 1-7
    - librarian and 1-29, 1-32
    - linker and 1-37
  - #49 2-4

## T

- T compiler option 1-15
- t compiler option 1-15
- targeting an earlier runtime 1-15, 1-20
- TBYTE option 3-43
- TBYTE qualifier 3-12
- Telnet, debugging via 2-9 to 2-10
- terabyte file 3-43, 3-81
  - ISAM 3-12
  - relative 3-22
  - sequential 3-25
- terminal I/O, non-interactive runtime and 1-55
- terminal settings runtime option 1-50
- text
  - customizing 4-3 to 4-10
  - file, converting 3-66
  - message file, creating 4-7
- TEXT option 3-43
- timestamp ISAM key 3-14, 3-94
- %TNMBR routine 1-55
- TRACE debugger command 2-47
- trace flag
  - compiler 1-23
  - librarian 1-32
  - linker 1-44
- traceback 2-43, 2-47, 5-3
- TRACK\_CHANGES option 3-43
- translating text to other languages 4-6
- trappable error
  - displaying mode 2-43
  - runtime 5-2, 5-4 to 5-36
- trapping errors 5-2
- /TRIM compiler option 1-15
- trim compiler option 1-15
- truncate compiler option 1-15
- truncating names 1-15
- %TTSTS routine 1-55
- /TYPE option 3-44



**U**

- u compiler option 1-13
- UI Toolkit, non-interactive runtime and 1-55
- uninitialized memory
  - displaying debugger mode 2-43
  - setting debugger to check for 2-40
- unloading an ISAM file 3-72
- unsigned ISAM key 3-14, 3-94
- .unu filename extension 1-16, 4-44
- unused variable, reporting 1-16, 4-44
- /USERDEF compiler option 1-11
- USING statement 1-12
- utility
  - ActiveX Diagnostic 4-31
  - chklock 3-48 to 3-49
  - dbl2xml 4-51 to 4-52
  - fconvert 3-62
  - Monitor
    - UNIX 4-23 to 4-30
    - Windows 4-21 to 4-22
  - servstat 4-15 to 4-20
  - synckini 4-14
  - Synergy DBL profiler 4-11
  - Synergy DBMS 3-34
  - Synergy Prototype 4-39 to 4-43
  - Synergy UI Toolkit Control Panel 4-3
  - Variable Usage 4-44

**V**

- V compiler option 1-8
- v compiler option 1-16
- ^VAL function 1-15
- variable
  - length
    - COUNTED option on isload 3-74
    - file 3-7
  - name, truncating 1-15
  - reporting nonusage 1-16, 4-44
- Variable Usage utility 1-16, 4-44
- /VARIANT compiler option 1-16
- ^VARIANT data reference operation 1-16
- /VAR\_REVIEW compiler option 1-16
- /VECTORS compiler option 1-16
- verbose mode when processing OLB 1-30
- verifying structure of ISAM file 3-62, 3-76, 3-79
- version, Synergy 1-53
- VIEW debugger command 2-48

**W**

- W compiler option 1-17
- w compiler option 1-17
- %WAIT routine 1-55
- warning
  - changing to error 1-17
  - disabling 1-17
  - error message
    - librarian 5-127
    - linker 5-122
  - library option 1-30
  - linker option 1-39
  - message 5-47
  - not generating 1-17
- warning disabled linker option 1-39
- /WARNINGS compiler option 1-17
- WATCH debugger command 2-49
- watchpoint
  - canceling 2-18, 2-29
  - displaying 2-43
  - saving 2-37
- WD compiler option 1-17
- wdtxt.ddf file 4-7
- width, compiler listing 1-17
- /WIDTH\_SIZE compiler option 1-17
- WINDBG debugger command 2-52
- window error messages 5-49 to 5-51
- windowing API 1-55
- Windows
  - elevated privileges 1-51, 1-54
  - Run dialog box 1-3
- Workbench
  - extended listing options 1-10
  - invoking commands from 1-3

**X**

- X compiler option 1-15
- x runtime option 1-50
- XDL file
  - contents 3-35
  - keywords
    - correspondence to FDL 3-97 to 3-100
    - description and values 3-90 to 3-97
  - rules 3-89
  - specifying 3-35
  - syntax checker utility 3-100
- xdlchk utility 3-100
- xfODBC, alternate keys and 3-80

## Z

xfpl.log file 4-20

### xfServer

- closing connection on remote port 4-21

- display status, OpenVMS 4-17

- free pool size, OpenVMS 4-18

  - checking and trimming 4-19

  - extend time 4-19

- global logicals, OpenVMS 4-18

- monitoring

  - UNIX 4-23 to 4-30

  - Windows 4-21 to 4-22

- purge free pool, OpenVMS 4-18

- server logicals, OpenVMS 4-19

- shutting down, OpenVMS 4-18

- system manager, OpenVMS 4-15

### xfServerPlus

- cycle log file, OpenVMS 4-20

- debugging and 2-8, 2-10

- displaying status information, OpenVMS 4-19

- ELBs and 1-42, 1-43, 1-46

- purging free pool, OpenVMS 4-20

XML, generating with dbl2xml 4-51 to 4-52

XUNDEF error 1-41

## Z

zoned store 1-8