Updating Synergy/DE

Version 9.5.3

Printed: December 2011

DCN UP-01-9503

Synergex
2330 Gold Meadow Way
Gold River, CA 95670 USA

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

Contents

Preface

About this manual

This manual is a record of changes made to Synergy/DE™ products. For each version of Synergy/DE, it lists new features and changes that may break code.

For information on the changes you may need to make to your system and code when you upgrade, see the *Synergy/DE Quick Migration Guide*.

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 an argument is omitted and the comma is inside the brackets, 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**).
- The term "environment variable" refers to logicals on OpenVMS, as well as environment variables on Windows and UNIX platforms.
- To "enter" data means to type it and then press ENTER. ("ENTER" refers to either the ENTER key or the RETURN key, depending on your keyboard.)

Other resources

The following Synergex® resources may be useful if you are moving to a new version of Synergy/DE:

- Synergy/DE release notes
- *Synergy/DE Quick Migration Guide* available in the Synergy/DE Online Manuals
- The *Migrating from VAX DIBOL to Synergy Language* guide available in the Synergy/DE Online Manuals
- The *Migrating Your Application to Windows* guide available in the Synergy/DE Online Manuals

Comments and suggestions

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

1

Synergy Language

Version 9

This section briefly describes new features in Synergy Language version 9 and the changes in this version that may break your code. For information on the changes you may need to make to your system and code when you upgrade, see the *Synergy/DE Quick Migration Guide*.

Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Synergy Language version 9.

- If you are using Synergy .NET, you must recompile all of your Synergy .NET code as we have optimized many compiler and runtime code paths for performance in 9.5.3. To force this recompile, the version of **synrnt.dll** has been changed in 9.5.3. (9.5.3)
- If your application uses global commons, global data sections, or public class fields that are accessed across assemblies, any time one of those elements changes, you must recompile all projects that reference the assembly containing them. As an example, in version 9.5.3, the UI Toolkit globals changed, so therefore any application that uses **Synergex.SynergyDE.tklib.dll** must be recompiled. To ensure that this occurs, we have changed the assembly version of the **tklib.dll** to force a recompile. We recommend that you use assembly versioning on your dependent projects as well. (Traditional Synergy does not have this problem because names are resolved at runtime, not at build time.) (9.5.3)
- With the addition of the IOHook class, changes were made to the underlying I/O routines in the runtime with regards to the error values. The value returned by %SYSERR and the *status* and *stv* parameters of the ERROR routine will be zero after each I/O operation. This is a change from previous behavior, where these values would have remained unchanged after a successful I/O operation, including writing to the terminal. Note that the %SYN_SYSERRTXT routine also uses the same %SYSERR value if the optional argument is not passed, which could be cleared by an I/O operation. The new %IO_ERROR routine enables you to retrieve the error values on a specific channel even though the general system error value may have been cleared. (9.5.3)
- The %SS_CLOSE routine has been enhanced to follow recommended techniques to ensure all data from a remote client is silently received and ignored on SS_CLOSE. This assumes that the remote application correctly closes down the socket when the protocol is finished. This forces the underlying TCP/IP transport to cleanly terminate and reduces dangling socket problems in the operating system. Under error conditions, or if an application just sends data and never expects any response, the optional purge flag should be specified on the SS_CLOSE to avoid the shutdown algorithm hanging. (9.5)

- Reversing the change made in version 9.3.1, the size of the Boolean type is once again 4. This was required so that traditional Synergy™ matches Synergy .NET, where the definition of Boolean is controlled by the .NET Framework. This modification may break your code. If you use Boolean types in Repository, see the related entry in "Features and fixes that may break your existing code" on page 6-2. (9.5)
- We changed the magic number in the 64-bit database project (DBP) files to be the same as the 32-bit magic number. This requires you to re-prototype all 64-bit files. (9.5)
- Previously, **gennet** incorrectly output a class with the name "Byte*" in certain situations. Prior to 9.5.1, the compiler didn't catch this error, and the code would compile and run. Because this error is not caught by the compiler, when recompiling code in 9.5.1 (and higher) that was previously generated by the gennet utility, you must re-run gennet, or many errors will be generated. (9.5)
- Passing ^NULL to an overloaded method previously selected the first overload. Now it gives an AMBSYM error, and you must cast ^NULL to the desired type to get the correct overload. This modification may break your code. (9.3.1b for Windows and UNIX, 9.5 for OpenVMS)
- (Windows) Windows 7 and all 64-bit Windows platforms now prevent exceptions from being thrown across the kernel (**user32.dll**) boundary. This means that 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 or throw an exception that could be caught in a routine on the other side of the callback without causing random hangs and crashes. We have updated the Synergy runtime to generate an $ERR_NOCHAIN error if a Windows callback attempts to perform a STOP chain. (We cannot, however, prevent the callback from throwing an exception that could be caught in a routine on the other side.) This modification may break your code. (9.3.1b)
- Prior to 9.3.1b, a prototype file created with the **-out** option that was in the second directory specified by SYNIMPDIR was not being loaded. We discovered the include guard processing was preventing the file from loading, so we added a flag to the prototype file that indicates when the include guard should be ignored so that it mimics the include guard processing of the old **.dbh** files. To pick up this change, prototypes must be regenerated. (9.3.1b)
- As of 9.3.1, ^ARGN returns $ERR_NULARG if a null argument ('') is passed, because a valid decimal descriptor can't have a length of 0. This modification may break your code. (9.3)
- (Windows) In 9.3.1, we added support for font orientation and escapement settings by adding two fields (orientation and escapement) to the end of the font_specs structure in **wndprint.def**. With these settings, you can rotate characters relative to the horizontal axis of the page, and you can rotate the text baseline relative to the horizontal axis of the page.

  If you use ^SIZE(font_specs) to allocate the font_specs structure passed to DWP_FONT, or if you use the structure definition in **winprint.def** directly, you will need to initialize the new fields to integer 0 to prevent randomly changing the escapement used by fonts in existing code. You can easily do that by passing the DM_NULL option on %MEM_PROC or by using the INIT statement on the structure. This modification may break your code. (9.3)

- We added the demand loading of prototypes to the compiler. To enable this, all prototype files are now binary files. You must re-prototype your application. (We use a **.dbp** file extension now instead of **.dbh**.) This greatly speeds up compiler load times and reduces dependencies between namespaces. (9.3)
- An application using an **a10** for an RFA may result in $ERR_RECNOT and $ERR_XFNOSPT errors being generated. All other RFA qualifier variables that exceed **a6** will now receive an $ERR_INVALRFA error. To suppress these conditions, set system option #56, which disables global RFA and restores prior behavior. (9.3)
- As of 9.3.1, the compiler generates an error on a relative range specification that accesses data that precedes the beginning of a field. Note that the ability to access characters that extend past (or in front of) the specified field has been deprecated. For backwards compatibility, the Synergy Runtime will continue to support this type of access. However, we recommend that you compile and run with **-qcheck** and make the appropriate changes to avoid future memory access violations. (9.3)
- As of 9.3.1, the signature of the virtual Equals method in System.Object has been changed to return boolean instead of **i1**. Existing methods that overload this method will need to be changed to match the new signature before recompiling. This modification may break your code. (9.3)
- (Windows, UNIX) As of 9.3.1, READ and READS from a relative file (or a file opened in the default mode with RECSIZ set) now generate an $ERR_TOOBIG error, as documented in the LRM, if the file record size is greater than the size of the data area. (This modification may break your code.) You can set system option #27 to temporarily get around this issue. (9.3)
- We changed the size of the Boolean type from 4 to 1. Boolean is now treated as its own separate, non-integer type. (9.1.5a)
- (UNIX) As a result of a change that eliminates random I/O failures, the FORK subroutine and the %HTTP_SERVER_RECEIVE function with the optional fork argument now close all channels opened by the parent in the forked child process. Therefore, files needed by the child must be reopened. This modification may break your code. (9.1.5a)
- For interoperability with .NET and other languages, indexes used by the System.Collections.ArrayList class are now 0-based instead of 1-based. If you use the System.Collections.ArrayList class (introduced in 9.1.3), you will need recompile your code with 9.1.5 and change your ArrayList references. See "Runtime" on page 1-19 below for detailed information. (9.1.5)
- To conform to a change by Microsoft to .NET version 2.0, the IndexOf and LastIndexOf methods in both System.Array and System.Collections.ArrayList now use the array element's Equals method instead of the target object's to determine a match. This modification may break your code if the System.Object.Equals method is overridden by objects in the array or the target object and they are different types. (9.1.5)

- Repository record, structure, or literal .INCLUDEs in a class now terminate by default. Previously these .INCLUDEs were open-ended unless you specified an ",end" qualifier to terminate the construct. This modification may break your code. (9.1.5)
- Arrayed fields larger than 256 MB are no longer supported and cause the compiler to generate a BIGSIZ error. This modification may break your code. (9.1.3)
- (Windows, UNIX) The runtime now detects duplicate common, global data section, and static records when ELBs are loaded. If the size is identical, a warning is printed in debug output. If the size is different, it is considered a fatal error. If you recompile your code, you will not see the duplicate static records. This modification may break your code. (9.1.3)
- As of 9.1.3, object libraries are portable between little-endian 64-bit machines on Windows and UNIX (system codes 021, 420, 428). All 64-bit Windows object libraries must be rebuilt if you are upgrading from a version prior to 9.1.3. This modification may break your code. (9.1.3)
- As of 9.1.3, object parameters cannot be optional in subroutines and functions that are not contained in classes. This modification may break your code. (9.1.3)
- If you are upgrading from version 9.1.1 or 9.1.1a to a later version, you will need to recompile and relink in order to use the debugger. (9.1.1b)
- (OpenVMS) As of 9.1.1a, the Synergy HTTP document transport API requires HP SSL Version 1.3 (based on OpenSSL 0.9.7e.). Previously, we linked against Version 1.1. This modification may break your code if the correct SSL version is not installed. (9.1.1a)
- In version 9.1.1a, we changed the handling of the Concat System.String class method to make it link without error. This modification may break your code. If your code currently uses the String class and includes addition (+) of string handles with other string handles or quoted strings, you must recompile that code. If you don't recompile, the order of strings will be reversed in the resulting string. (9.1.1a)
- We have changed the way in which .INCLUDE offsets are specified by the compiler. If the record is named, dot notation using the record, group, and field names is used. If the record is unnamed, a numeric offset relative to the overlay field is used. The use of the named reference could cause an "Ambiguous symbol" error to be reported by the compiler if you have a duplicate symbol in your code. (9.1)
- We removed %SYNSOCK from the runtime because, due to strong prototyping, system-supplied parameterized macros are no longer used. The %SYNSOCK subfunctions are still available as functions, as documented in the "Synergy Socket API" chapter of the *Synergy Language Reference Manual*. This modification may break your code. (9.1)

New features

With Synergy/DE 9.5, we introduced support for .NET. This includes a new compiler (**dblnet**), which enables Synergy code to be run under the Common Language Runtime (CLR). It also includes Visual Studio integration, which provides templates for Synergy/DE projects, a C#-to-Synergy code converter, and other features.

Analysis engine

- The .DEFINE resolution has been optimized to only use local fields and class fields. This means fields defined in imports will not give an .IFDEF true result, which is the same behavior as other languages. (9.5.3)
- Previously, calling a subroutine in the global namespace from within a class that has a subroutine of the same signature wasn't possible. We added the ability to specify "^global" as part of the subroutine name path to allow specifying the global namespace. For example,

  ```
  xcall ^global.sub1()
  ```

  (9.5)
- We added support for partial classes, with the following restriction: all declarations of a specific partial class must be compiled and prototyped together in one compilation unit. (9.3)
- A dynamic array of 0 elements can now be created. (9.3)
- We added the sbyte (signed byte) data type and changed the byte data type to represent an unsigned byte in .NET. (9.3)
- We added support for the .REGION and .ENDREGION preprocessor directives, which specify a collapsible/expandable block of code for Visual Studio .NET and Workbench. (9.3)
- You can now specify "(*)" on a parameter declaration to support the passing of pseudo arrays. If the **-qnet** compiler option is specified, only pseudo arrays can be passed to arguments marked with (*). If neither **-qnet** nor **-qstrict** is specified, (*) is currently ignored. (9.3)
- We added a new initial value syntax for the dimensioned NEW operation for dynamic arrays. For example, new i4[#] {1, 2, 3}. (9.3)
- You can now add attributes and documentation comments to your Synergy code, which will be processed into an XML file if you run the **dbl2xml** utility. See "Using Attributes to Define Synergy Methods" in the "Defining Your Synergy Methods" chapter of the *Developing Distributed Synergy Applications* manual for details on this feature. (9.3)
- We added prototype demand-loading to the compiler to speed up compiler load times and reduce dependencies between namespaces. (9.3)
- We now support bitwise operations within an enumeration. (9.1.5a)
- (Windows) We enhanced the performance of namespace import processing in the compiler. (9.1.3)

- A level 4 warning is now generated when an implied decimal is passed to an **n** or **d** parameter. (9.1.3)
- A level 4 warning is now generated if a simple assignment like a = b is attempted in a Boolean expression in an IF statement. (9.1.3)

Code behind

- We made several changes in the code-behind support to better support the design-time experience with the Microsoft, Infragistics, and DevExpress controls. (9.5.1a)

Code converter

- We made numerous improvements to the code converter, and generated code from C# is much cleaner with direct one-to-one statement mapping when available. (9.5.1a)

Code snippets

- We added more snippets and improved navigation within snippets, including auto expansion of enumerations in the using snippet. (9.5.1a)

Compilers (dbl & dblnet)

- We added a new function, ^ARGPRECISION, to return the precision of an implied argument. (9.5.3)
- We added a new define, SYN_VERSION, to **dbl.def** in preparation for version 10. (9.5.3)
- We added support for passing an alpha into an OUT/INOUT non-CLS structure parameter. We added W_ATOS warnings when passing an alpha to a non-CLS structure parameter and an E_ATOS error when an alpha that's too small is passed to a non-CLS structure. A warning occurs in traditional Synergy. (9.5.1b)
- We added a PASSUR warning (level 3 in **dblnet**, level 4 in **dbl**) that will occur when a decimal or implied-decimal type is passed into a MISMATCH alpha argument that is not marked as IN. (9.5.1a)

Compiler (dbl)

- We added a **-qnet** warning when ^DATATYPE(*literal*) is called on a MISMATCH alpha parameter. If ^DATATYPE(*variable*) is called, a level 4 warning will be generated. (9.5.1b)
- We added an INVFCALL warning (level 4) for functions that are called without a "%" or without parentheses. Because such functions are susceptible to other compile-time errors that may not be obvious, we recommend that you correct them. (9.5.1a)
- Structfields are now fully supported in group arguments. (9.5.1a)
- Methods whose signatures only differ by rank on fixed arrays are not allowed in .NET. We added the DUPMRNK **-qnet** warning to catch these. (9.5)

- We added the OVRCASE **-qnet** warning for instances when the case of an override method or property name differs from the case of the abstract method being overridden. (9.5)
- Instance arrays can now be passed as arguments and accessed as arrays in the called program. NOTE: Programs built with this change must be run in at least the same version of Synergy due to a required runtime change in conjunction with the compiler change. (9.5)
- (OpenVMS) The .ALIGN PAGE compiler directive has been changed to align on a 4096 boundary on OpenVMS I64 and an 8192 boundary on OpenVMS Alpha. All other platforms will still align on a 512-byte boundary. (9.5)
- We added a **-qnet** warning when using "new string(*alpha*)". (9.3)
- (Windows, UNIX) The maximum number of arguments that may be passed on a single line to **dbl** and **dblproto** from a redirected input file has been increased from 125 to 4000. The maximum number of characters that may be passed on a command line to **dbl** and **dblproto** has been increased from 32K to 128K. (9.3)
- The compiler now supports **d28** and **d28.28** variables. (9.3)
- We added the SYNUSERDEF environment variable, which specifies a file to implicitly include at the beginning of a compilation unit. This enables you to import multiple namespaces into a project without explicit IMPORT statements in your source code, as well as to define identifiers that will apply across all files in the compilation unit. (9.3)
- (Windows) We added support for the REMOVEHANDLER statement for code generated by the **gennet** utility. (9.3)
- In addition to structures, records, and groups, an enumeration type can now be .INCLUDEd from a repository. (9.3)
- Calling a subroutine as a function and calling a non-^VAL function as a subroutine won't be supported in .NET. Therefore, we added a **-qnet** warning in these cases. (9.3)
- We added support for the boolean, decimal, double, float, guid, int, long, sbyte, and short data types. (9.3)
- We added the INVTYPSIZ **-qnet** warning for when a .NET type (byte, sbyte, short, int, integer, long, decimal, double, or float) is used in a fixed array. (9.3)
- We added **-qnet** warnings for when an external field differs by type or size from its corresponding global field and when a .NET type is used in a common. (9.3)
- A W_NARROWING level 4 warning now occurs when a larger numeric is passed to a smaller numeric. (9.3)
- It's now possible to mark a structure field as public even if the structure is marked private. (9.3)
- We added support for **n** and **n.** to non-unique methods. (9.3)
- Using a call to a label outside a FOREACH loop causes premature termination of the loop. Due to implicit local data, a CALL to labels outside the loop's scope is now prohibited. (9.3)
- We removed a **-qnet** restriction that prevented parameter groups from being used in .NET. Parameter groups are now supported. (9.1.5a)

- We added new **-qnet** restrictions for non-Synergy types (i.e., object handles and .NET value types). The following are not allowed:
  - A non-Synergy type in a group within a structure
  - A non-Synergy type in a group where dimensioned group access is done
  - A non-Synergy type in a parameter group
  - An object handle in a structure that is used in a real array
  - A non-Synergy type in a structure used in a ^M routine

  (9.1.5a)
- We separated .NET warnings for APIs into their own warning, "Routine %s not supported in .NET" (NETAPI), so you can selectively disable it if you don't want to display .NET support warnings for API calls when compiling with the **-qnet** option. (9.1.5a)
- A new level 4 warning, "Routine hides data reference operation %s" (HIDEHAT), is now generated when a routine declaration hides a data reference operation. (9.1.5a)
- We modified the compiler output of debug information for group arguments to allow the expanded fields to be displayed when the group argument name is examined in the debugger. (9.1.5a)
- Method overloads that only differ by optional parameters now report a new error: "Class %s already defines a method %s that differs only by optional parameters" (DUPOPTS). This error is also reported if more than one method overload per ID has trailing optional parameters (including parameters defined explicitly and methods marked with VARARGS). (9.1.5a)
- We added a new statement, FOREACH, which iterates sequentially through elements in a collection. The syntax is

  `FOREACH` *loop_var* `in` *collection*
  iteration processing
  .
  .
  .

  where *loop_var* is a loop variable whose value is set to each element in sequence as FOREACH iterates through them and *collection* is one of the following:
  - A dynamic system array of rank 1
  - A real Synergy array of rank 1
  - The System.Collections.ArrayList class or a descendant
  - The Synergex.SynergyDE.Collections.ArrayList class or a descendant
  - A class generated by gennet that inherits from DotNetObject and can be converted to IEnumerable

  (9.1.5)

- We added the ability to specify the MISMATCH modifier on an alpha parameter for functions and subroutines, which allows passing a decimal or an implied-decimal argument to the alpha parameter. (9.1.5)
- We added a new compiler option, **-WD**, to allow specific warnings to be disabled. This option is especially useful if you are encountering a lot of level 4 warnings. The syntax is

  `-WD=`*error_no[,...]*

  where *error_no* is the number of the error that you want to disable. You can specify multiple errors, separated by commas.

  For example,

  ```
  dbl -W4 -WD=885,777 filename.dbl
  ```

  (9.1.5)
- The following routine parameters are now defined as MISMATCH alpha: INSTR's second and third arguments, GETLOG and SETLOG's second argument, and %XML_ELEM_SETATTRIBUTE's third argument. This allows decimal variables to be passed to the changed parameters without a compiler error or the use of the **-qrelaxed:interop** option. (9.1.5)
- The compiler now allows you to cast an alpha literal to a string when passing the literal to an object. (9.1.5)
- An OLYBD error is now generated when a class field that is not in a record overlays another field. (9.1.5)
- (UNIX) The compiler now allows .INCLUDEing from a repository that is accessed via *xf*Server. (9.1.3)
- We added a new compiler switch, **-qnoargnopt**, to relax integer optimization of **n** type arguments that are used in CASE and USING statements. Implied values passed to **n** arguments will be treated as their full implied value. (9.1.3)
- We added the **interop** option to the **-qrelaxed** compiler option. Using **-qrelaxed:interop** compiles classes generated by the **gennet** utility. Identifiers longer than 30 characters are truncated instead of generating a warning. (9.1.3)
- The CASE statement now allows implied selection variables and labels. Implied labels are only allowed with implied selection variables. (9.1.3)
- Arrays of structfields that contain either alpha or decimal fields can now have initial values. Each initial value must be the same size as the structfield, and if there are any initial values, all array elements must have an initial value. (9.1.1c)
- We added a new compiler option, **-qerrwarn** (or **/ERRWARN** on OpenVMS), that turns warnings into errors. (9.1.1b)
- We enhanced error handling for invalid I/O statement qualifiers. (9.1.1b)

- (OpenVMS) The OpenVMS help for SYNERGYDBL has been updated for version 9. (9.1.1b)
- Signatures of prototype and external functions are now displayed for PROTOMISMCH errors when the **-W4** compiler option is set. (9.1.1b)
- The error text for usage of the NEW modifiers is now more specific. (9.1.1b)
- We added a **-qrelaxed** compiler option to help with backwards compatibility regarding strong prototyping. We don't recommend that you keep this option on except for initial porting work. (9.1)
- We added a **-qstrict** compiler option, which enforces real array bounds checking at runtime. We recommend that all production code be compiled with **-qstrict**. (9.1)
- We added a new environment variable, SYNCMPOPT, which can be used to set compiler options for all subsequent compiles. You can use SYNCMPOPT to easily add or remove the new **-qstrict** or **-qrelaxed** option. (9.1)
- We added a **-qnet** compiler option, which turns on .NET compiler warnings for items that will not be included in Synergy/DE support for .NET. These include deprecated data types, syntax, APIs, compiler options, and alignment warnings. Warnings are output to standard error. (9.1)
- All definitions that were previously included via **dbl.def** are now built into the compiler. Any explicit references are now ignored. The **dbl.def** file is only provided in your distribution for reference purposes. You can still use the compiler option **-g** (on Windows and UNIX) or **/nogbldefs** (on OpenVMS) if you do not want these global definitions to be available. (9.1)
- We now cache repository .INCLUDE information for better performance when doing multi-module compilations. (9.1)
- The maximum number of include levels is now 65,535, up from 8. (9.1)
- We augmented the Synergy Language with a complete object-oriented implementation. See the *Synergy Language Reference Manual* for details. Three object-oriented example programs have been added to the distribution for your reference: **myclass.dbl**, **myooprogram.dbl**, and **mystructprogram.dbl**. (9.1)
- We added the @*class*, @*, and *structfield* data types. (9.1)
- We added the INIT statement to initialize a field or record to its declared default value (or the default value for the type if a declared default value is not provided). Note: This is not the cleared value but the initial value, including any initial values specified in the data division. (9.1)
- We now support arrayed arguments when passing structures mapped to handles to a subroutine. (9.1)
- We now support dynamic arrays for classes. You can declare a dynamic array of reference types whose upper bounds are determined at runtime by specifying a "#" for each array dimension (for example, [#,#]). (9.1)
- We added structured exception handling (TRY-CATCH-FINALLY). See the *Synergy Language Reference Manual* for details. (9.1)

- You can now specify DIMENSION when doing a .INCLUDE from a repository. (9.1)
- You can now specify accessibility options (PUBLIC, PRIVATE, or PROTECTED) when doing a .INCLUDE from a repository for records, structures, or groups. (9.1)
- Users can now specify direction (IN, OUT, INOUT) when doing a .INCLUDE from a repository to include a group as a parameter to a method, function, or subroutine. (9.1)
- We added the ^ARGTYPE function. See the *Synergy Language Reference Manual* for details.
- We now support several system classes compatible with .NET (System.String, System.Object, System.Exception). See the *Synergy Language Reference Manual* for details. (9.1)
- We added the following data types for future compatibility with .NET: decimal, string, byte, short, int, and long. These are automatically mapped to compatible Synergy types for version 9. (9.1)

Compiler (dblnet)

- We added support for declaring extension methods in Synergy .NET. Now you can call a method by importing the extension method's namespace. For example:

  ```
  someobject.MyExtensionMethodName(param1, param2)
  ```

  (9.5.3)

- We added support for declaring and using lambda functions. Use the LAMBDA statement to declare a lambda function. (9.5.3)
- We added support for ^INCR() and ^DECR(), to allow incrementing and decrementing within a complex path. For example,

  ```
  ^incr(myvar).ToString()
  ^decr(myvar).ToString()
  ```

  (9.5.3)

- We added support for type inference on DATA statements:

  [`DISPOSABLE`*]* `DATA` *variable* = *value*

  where *value* is a data reference to a local variable or any literal initial value. If *value* is a reference to a local variable, the type of *variable* is the same as the data reference. If *value* is an initial value, the type of variable is determined by changing the Synergy literal type to the same .NET literal type. (9.5.3)

- DATA statements no longer have to be enclosed in BEGIN-END blocks in Synergy .NET. (9.5.3)
- We added support for ^AS(), to convert an expression to the specified type. The syntax is

  `^AS(`*expression, type_name*`)`

  (9.5.3)

- When assigning a literal to a field or property whose type is System.Object, the compiler will automatically box the literal using .NET types. Also, casting a literal to an object will use .NET types to remain consistent with all automatic boxing. (9.5.3)
- If a value type is assigned to a System.Object type variable the compiler now automatically boxes the value type. (9.5.3)
- We added syntax for custom event-handler methods within an event. The syntax is

  `PUBLIC` *event_mod* `EVENT` *event, @delegate*

  (9.5.3)
- We added support for STATIC READONLY fields. (9.5.3)
- We added support for setting an attribute field value when using the attribute. (9.5.3)
- We added support for specifying the DllImport attribute on a method to denote an external method call in a Win32 DLL and setting the DllImport field values as needed. (9.5.3)
- We added event += and event -= syntax that is equivalent to ADDHANDLER and REMOVEHANDLER, respectively. (9.5.3)
- Multi-dimensional arrays of groups and non-CLS structures are now supported. (9.5.1b)
- We added support for specifying attributes on an enumeration value. (9.5.1a)
- Previously, an ID in the middle of a path whose name matched an operator caused an INVEXPR error. For example, cvar.band.fld1 treated “band” as an operator in the parser. We added the ^ID function to encapsulate an ambiguous ID, which enables the parser to recognize it as an ID. In the example above, the offending path would become cvar.^id(“band”).fld1. (9.5.1a)
- We added the MISIMP3 error to indicate when a method, event, or property in a class has a different case than the associated interface’s equivalent matching member. Previously a TypeLoadException occurred at runtime. (9.5.1a)
- Override methods and properties that don’t match the case of the original might not get run in a virtual calling scenario. If this situation is encountered, the case for the override method will be changed to match the base method, and a CHGCASE warning (level 3) will be issued. (9.5.1a)
- We added support for loading, using, and declaring generic delegate types, methods, and interfaces. (9.5.1a)
- We added support for the ^TYPEOF data reference operation to determine the data type in an attribute value. (9.5.1a)
- We added support for D_ADDR in a structure with ^M. (9.5.1a)
- The DATA statement can now appear anywhere within a BEGIN-END block. If it is used before it is declared, a USEBDECL error occurs. (9.5.1a)

- We added support for mixing positional and assignment attribute values in an attribute. Note that positional values must come first, or an UNSUPPORT error will occur. (9.5.1a)
- To support .NET, we added the **dblnet** compiler, which generates MSIL, enabling Synergy code to be run under the Common Language Runtime (CLR). (9.5)

Debugger

- (Windows) The debugger has been enhanced to include an optional /FULL switch on the SHOW CHANNELS command. When the /FULL switch is specified, the full Windows filename is returned for each open file. The /FULL switch is ignored on Windows systems prior to Vista. (9.5.3)
- The limitation of 255 source files on a debug compile of a compilation unit has been lifted. This is the default compilation method for Workbench projects where multiple source files are built into a single **.dbo** file. You must recompile any files in 9.5.3 to debug those routines in 9.5.3. (9.5.3)
- We enabled support for multidimensional arrays in the debugger. (9.5.1a)
- We added an internal member view to the arraylist debug display so members can be displayed. (9.5.1a)
- We added two new options to the SET command:
  - SET STOP ON causes a break in the debugger when a STOP statement is executed. This break is after any destructors are executed.
  - SET STOP OFF turns off breaking at a STOP statement.

  (9.5)
- We added a new option for the SHOW command. SHOW STOP displays the current state of breaking at a STOP statement. (9.5)
- We added the SHOW VARIABLE *var_list* command to show the type and size of a variable or list of variables. If present, the VARIABLE keyword must be the last keyword on the SHOW command line, followed by one or more variable names separated by spaces and/or commas. (9.3)
- We added the SET VIEW *num* command, which sets the default number of source lines that will be displayed before and after the current line when the VIEW command is specified without a count. (9.3)
- We added two new debugger commands, SET UNINITIALIZED ON|OFF|BREAK and SHOW UNINITIALIZED. (These were released as “experimental features” in 9.1.5b.) (9.3)
- You can now use the Edit > Mark and Edit > Copy features on the Synergy application window, the debugger window, and the Toolkit debugger window. This enables you to copy information displayed in one of these windows and then paste it into other applications for further debugging. (9.3)

- The debugger has been enhanced to allow the following:
  - Stack record variables to be accessed by @*index*
  - EXAMINE of object fields by @*objidx*.@*index* (but EXAMINE of @*objidx* is not allowed)
  - EXAMINE of ^m(@*strfldidx*,@*hndidx*)

  (9.1.5b)
- The debugger can now cast an object handle by a boxed object specification such as (@d)hnd or (@ns.str)hnd. (9.1.3)
- We added the command GO /DEBUG, which will proceed until a routine compiled with debug is entered. (9.1.3)
- You can now display all static fields in a class by using EXAMINE /STATIC *class_name*. (9.1.3)
- The debugger help text for EXAMINE has been updated to include the /STATIC switch. The text for /PAGE has been updated in the help for the EXAMINE, SHOW, and SHOW CLASSES commands. (9.1.3)
- The online Help text for the debugger has been updated for object support. (9.1.1b)
- (Windows) We enhanced the Synergy debugger window to be able to scroll to view more than 25 lines of text. By default, 300 lines of display are saved, but you can override this setting in **synergy.ini** or **synuser.ini** with the new DBG_BUFFER environment variable. Set DBG_BUFFER to the number of lines of display that you want the debug window to retain. (9.1)
- You can now log a debugger session to a file. To start logging, issue the command

  `LOGGING START` *filename*

  The filename may be quoted. To stop logging, issue the command

  `LOGGING STOP`

  (9.1)
- The debugger now allows a comma as the optional separator between elements in the EXAMINE and SHOW commands. (9.1)
- When you EXAMINE a named record or group, the debugger now shows all named members. If you want to display the record or group as the debugger did previously, you can use the EXAMINE /A option. (9.1)
- The debugger command GO/NODEBUG now cancels all watchpoints in addition to all breakpoints. (9.1)
- The debugger now allows /PAGE at the end of the command line for the following debugger commands: SHOW, HELP, and EXAMINE. (9.1)

Editor

- Hover-over support is greatly enhanced when editing. (9.5.1a)
- We made several fixes to improve stability and avoid Visual Studio crashing while editing. (9.5.1a)

Installation

- On a 64-bit operating system, 32-bit and 64-bit Synergy/DE now install a shortcut under the Start menu for the installation that will launch a command prompt with Synergy environment variables already sourced. (9.5.3)
- (Windows) We have upgraded to InstallShield 2012, and the distributed **setup_base.ini** file has changed. If you are using the shared installation feature of Synergy/DE version 8.1 or later, the version 9.5.3 installation will update your customized **setup.ini** file automatically with the changes. First, the version 9.5.3 installation will rename your current **setup.ini** file to **setup_old.ini**. It will then create a new **setup.ini** file by copying the distributed **setup_base.ini** file. Finally, it will merge your customized command line settings from **setup_old.ini** ("CmdLine=") into the new **setup.ini** file. (9.5.3)

  IMPORTANT: If you are using the shared installation feature of Synergy/DE version 7.5, the version 9.5.3 installation will *not* update your customized **setup.ini file**. Because **setup.ini** was an "installed" file in version 7.5, it will be removed when you uninstall 7.5 or upgrade. All customizations will be lost. If you are currently on version 7.5, *you must save off your **setup.ini** customizations before you install version 9.5.3.* After installing version 9.5.3 on the shared machine, copy your command line settings ("CmdLine=") from your saved-off **setup.ini** file to the newly installed **setup.ini** file.
- We modified the License Configuration screen in the Synergy/DE installations to clarify the licensing options being presented. (9.5.3)
- When installing Synergy/DE on a 64-bit machine, the License Configuration screen now defaults to "License Server or Standalone" as opposed to License Client. (9.5.3)
- We added support for hosting WCF services in IIS to Synergy/DE Core Components. (9.5.1a)
- Synergy/DE Client can now be uninstalled without having the host machine available. Note, however, that the registry entries for the **SPR32X?.ocx** and **tkgrid?.ocx** files will not be uninstalled, as there is no access to the uninstall methods of controls that reside on the host machine. (9.5)
- (Windows) When using a 64-bit machine on which both 32- and 64-bit Synergy/DE are installed, if you previously tried to run Composer from Workbench, you got a "composer.dbr not found" error, because SYNBIN was set to the path for the 64-bit directory and Composer is always 32-bit. The SDE installation now sets SYNBIN to SYNERGYDE32 when installing on a 64-bit machine. Make sure the SYNBIN setting in your own **synergy.ini** files points to the 32-bit installation directory. (9.5)

- (Windows) The 32-bit Synergy/DE Client installation now sets the SYNERGYDE32 environment variable; the 64-bit Synergy/DE Client installation sets SYNERGYDE64. (9.5)
- (UNIX) The **synxfpng** utility is now distributed with Synergy Language instead of Synergy *xf*Server. (9.3)
- (Windows) We now support the MSI command line switch **/q** and its variants, which enables you to control the user interface level of the installation. (9.3)
- (Windows) 64-bit Synergy/DE now includes "Files for shared installation" as an installation option. On server machines, the feature is included in the default selection of features. (9.3)
- (64-bit Windows) Professional Series Development Environment is now installed by default. (9.3)
- (Windows) You are no longer required to enter a SlickEdit license key when you install Professional Series Workbench. It is now pre-licensed. (9.3)
- (Windows) A 64-bit version of Synergy/DE Client is now available. (9.3)
- (Windows) 32-bit Synergy/DE Client can now be installed on a 64-bit operating system. (9.3)
- (Windows) The minimum supported Service Pack Level for XP is now 3. The minimum supported Service Pack Level for Vista is now 2. The minimum supported Service Pack Level for Server 2008 is now 2. These minimum service pack levels are enforced by the installations. (9.3)
- (Windows) Users upgrading from Synergy/DE version 7.5 and earlier will have to manually add *xf*Server services. Refer to the "Configuring xfServer" chapter of the *Installation Configuration Guide* for information on how to do this. (9.3)
- (Windows) The SDE component "Files for shared installation" is now installed by default on server platforms. (9.1.5)
- (Windows) Connectivity Series is now installed by default. (9.1.5)
- (Windows) Our installations now enforce operating system minimums. The following Service Pack levels are enforced:

Windows 2000	SP 4
Windows XP	SP 2
Windows 2003	SP 1

  (9.1)

- (Windows) We have upgraded to InstallShield 12. InstallShield 12 no longer distributes the InstallScript engine (**isscriptXXX.msi**). When you install or upgrade the "Files for shared installation" component, note that this file will no longer appear in the synergyde\client directory. This change affects the distributed **setup_base.ini** file. (9.1)

- (Windows) The **setup_base.ini** file has changed. If you are using the shared installation feature of Synergy/DE version 8.1 or 8.3, the version 9 installation will update your customized **setup.ini** file automatically with the changes. First, the version 9 installation will rename your current **setup.ini** file to **setup_old.ini**. It will then create a new **setup.ini** file by copying the distributed **setup_base.ini** file. Finally, it will merge your customized command line settings from **setup_old.ini** ("CmdLine=") into the new **setup.ini** file.

  IMPORTANT: If you are using the shared installation feature of Synergy/DE version 7.5, the version 9 installation will *not* update your customized **setup.ini** file. Because **setup.ini** was an "installed" file in version 7.5, it will be removed when you uninstall 7.5 or upgrade. All customizations will be lost. If you are currently on version 7.5, *you must save off your **setup.ini** customizations before you install version* 9. After installing version 9 on the shared machine, copy your command line settings ("CmdLine=") from your saved-off **setup.ini** file to the newly installed **setup.ini** file. (9.1)

- (Windows) All executable files (**.dll**, **.exe**, **.ocx**) that Synergex produces are now Authenticode signed. (9.1)

- (Windows) Our installations now automatically create an installation log file on target machines running Windows Installer Service 4.0. The logfile will be created in the user's %temp% directory with a filename of **MSI*xxxx*.log**, where *xxxx* is alphanumeric and randomly generated. (9.1)

IntelliSense

- We made many fixes to IntelliSense, including better support for generic IntelliSense. (9.5.1a)

Librarian

- Errors issued by the librarian now unmangle any method names that are displayed. Also, when the **-t** switch is specified to generate a table of contents, both the mangled name and the unmangled name are displayed when a method is output. (9.3)

- The performance of the librarian has been improved. (9.3)

Linker

- The linker has been modified to put the version of the linker into the header of the **.dbr** and **.elb** files. The **listelb** utility has a new switch, **-i**, which causes the version of the linker used to create the **.elb** file to be displayed. The **listdbr** utility also has a new **-i** switch, which causes the version of the linker used to create the **.dbr** file to be displayed. (9.3)

- Errors and warnings issued by the linker now unmangle any method names that are displayed. (9.3)

- We improved linker performance when linking programs with ELBs. (9.1.5a)

- We added a new linker option:

  `-R` *olb_file*

  Use this option to create an ELB when not all of the routines in an OLB file are desired. Only the routines in the specified OLB that are necessary to resolve the routines in the ELB will be included. You can specify **-R** multiple times to specify more than one OLB file. For example,

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

  (9.1.5)

- The linker now processes debug information in 16K buffers instead of 1K buffers. This change speeds up the linker on mapped Windows drives. (9.1.5)
- The linker now adds references to virtual methods and destructors to the list of external links when creating an ELB. This allows the linker to pull in external references when linking against an ELB. We also enhanced the **listelb** utility to list the external link references in an ELB. (9.1.5)

Project

- We added support for Microsoft macros for build commands and properties (those that aren't deprecated or specific to other languages) to the Environment Variables page of project properties. (9.5.1a)
- We added functionality to copy and paste and drag and drop existing files into folders within the project hierarchy. (9.5.1a)

Runtime

- We enabled GRFA support for READ, READS, WRITE, and WRITES on stream, sequential, relative, and block files. (9.5.3)
- A new clone method was added to System.Collections.ArrayList and to the Synergex.SynergyDE.Collections.ArrayList that performs a shallow copy of the objects in the original ArrayList into the newly created ArrayList. This means that the new ArrayList has references to the same objects that are referenced in the original ArrayList. (9.5.3)
- We added a new IOHooks class to support I/O statement hooks. See Synergex.SynergyDE.IOExtensions.IOHooks in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual* for details. (9.5.3)
- We added a new function, %ISLITERAL, that returns a 1 on Windows and UNIX if the passed-in argument is a literal or a runtime temp. The syntax is

  status `= %ISLITERAL(`*argument*`)`

  See %ISLITERAL in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual* for details. (9.5.3)

- (Windows, UNIX) We added a new runtime startup switch, **-dz**, to allow the runtime to break into the debugger on an untrapped error with a post-mortem debug session. (9.5.3)
- We added a Dispose method to Select and From classes for compatibility with Synergy .NET. (9.5.3)
- We added a new function, %IO_ERROR, to return error numbers on a specific I/O channel. The syntax is

  `%IO_ERROR(`*ch, [syserr], [sysstv]*`)`

  See %IO_ERROR in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual* for more information. (9.5.3)
- (.NET) We added support for specifying environment variables using an **app.config** file. (9.5.3)
- Performance of the Select class using large records has been improved. To enhance performance even more, a new From class flag (Q_NO_GRFA) has been added to disable the generation of a GRFA for each selected record. This flag should only be used if a GRFA will not be needed from the selected match set. To use the Q_NO_GRFA flag, use the lock and wait form of the From class constructor, and specify Q_NO_GRFA on the lock argument (using bitwise OR with a locking option). For example,

  ```
  fobj = new From(ch, Q_NO_GRFA, 0, rec)
  ```

  or

  ```
  fobj = new From(ch, Q_NO_GRFA|Q_NO_LOCK, 0, rec)
  ```

  (9.5.1a)
- (.NET) %TNMBR now returns the same value in Synergy .NET as it does in traditional Synergy when the TNMBR environment variable is not set: 0 if the process is an interactive user session and -1 if it is not attached to the user interface. (9.5.1a)
- When a client/server relative, sequential, or stream file is opened in output or append mode, and caching is enabled with SCSPREFETCH, the records written with WRITES or PUTS to the file are cached. The cache is flushed when the buffer is full, a READ/READS/FIND/FORMS/WRITE/PUT is performed, or a WRITES/PUTS with a GETRFA is performed. An explicit FLUSH statement flushes the buffer and turns off caching. (9.5.1a)
- A new OPEN option, BUFSTORE, has been added to enable the records being stored to be cached on the client side of an *xf*Server connection when system option #36 is not set. The buffered stores are flushed when the buffer is full, a READ/READS/FIND is performed, or a STORE with a GETRFA is performed. An explicit FLUSH statement flushes the buffered stores and turns off store caching. (9.5.1a)
- (.NET) Due to performance improvement changes, the version of the **synrnt** DLL was changed to force a code rebuild. (9.5.1a)

- Select now includes From class constructors that take a filename and optionally an open_option string. By default, a specified filename with the extension **.ism** is opened **I:I**. A filename that does not end in **.ism** is opened **I:S**. The open_option string argument can be used to override these defaults, as well as specify any of the options supported by the OPTIONS modifier in the OPEN statement. (9.5)
- The Select class now includes a Delete() method that deletes all records matching a selection. When using *xf*Server, all I/O is performed on the server. (9.5)
- We added a SparseUpdate method to the Synergex.SynergyDE.Select.AlphaEnumerator class. It works in conjunction with SparseRecord to limit the amount of data transferred to *xf*Server when updating a record. (9.5)
- The RFA returned in the onLock method of the Select.Event class now includes the CRC portion for GRFA use. (9.5)
- (.NET) We added the ^VARARGARRAY data reference operation to access and pass VARARGS undeclared arguments at runtime. (9.5)
- (Windows, UNIX) New *xf*Server transmission counters can be retrieved by using one of four new GETFA keywords:

  XFT Return total number of bytes sent and received on a channel opened to a remote file.

  XFS Return total number of bytes sent only on a channel opened to a remote file.

  XFR Return total number of bytes received only on a channel opened to a remote file.

  XFP Return total number of round-trip packets issued on a channel opened to a remote file. The statistics returned are the total accumulated, from the time of the OPEN to the time of the GETFA call.

  (9.5)
- We improved the performance of the QSORT routine. (9.3.1a)
- We added a new ^VAL function called %GETCRC to retrieve the internal CRC value from a GRFA. (9.3.1a)
- The Select class is now available for selecting records from a file based on fields within the record. See "System-Supplied Classes" in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual* for more information. (9.3)
- An additional type of RFA called a GRFA is now available. Unlike a normal 6-byte RFA, a GRFA is 10 bytes and consists of the 6-byte RFA and a 4-byte CRC of the record associated with the RFA. To use a GRFA, simply pass an **a10** (instead of an **a6**) variable to either the GETRFA or RFA I/O qualifiers. (9.3)
- The runtime now supports **d28** and **d28.28** variables. (9.3)

- We added uninitialized memory checking to the runtime. (This was released as an "experimental feature" in 9.1.5b.) This uninitialized memory checking occurs when running in debug mode and will help detect where your application is accessing uninitialized memory before writing to it (in assignment statements and "IF" tests). Uninitialized memory occurs for ^M and stack records. (9.3)
- Data field encryption has been implemented with these three new subroutines: DATA_SALTIV, DATA_ENCRYPT, and DATA_DECRYPT. (9.3)
- We added the %TIMEZONE function, which returns the time zone of the current machine. (9.3)
- %DATETIME has a new, optional argument that is the offset of the number of minutes east of Greenwich to be applied to the current local time. (9.3)
- The Synergy symbol table API now allows associated data to be longer than 100 bytes. (9.1.5b)
- (Windows) We added a new function, %SET_PRIORITY_CLASS, to allow a program to change its priority class if privileges allow. (9.1.5)
- If you use the System.Collections.ArrayList class introduced in 9.1.3, you will need to change your ArrayList references.

  For interoperability with .NET and other languages, indexes used by the System.Collections.ArrayList class are now 0-based instead of 1-based. (This class was introduced in Synergy/DE 9.1.3 with 1-based indexes.)

  To prompt your attention regarding this change, the 9.1.5 runtime will generate a fatal ALCOMPAT error when System.Collections.ArrayList is used. This error will go away when you compile your code with 9.1.5. However, you must still perform one of two steps to resolve the issue, or your code may contain errors (where it is depending on the indexes to be 1-based).

  1. If there's any chance that you will migrate your code to .NET in the future, you should convert all ArrayList index references to be 0-based. This will be a requirement for interoperability with other .NET languages. We recommend using the FOREACH statement with all FOR loops that reference ArrayLists.
  2. If future interoperability with .NET is of no concern, you can simply change all references to ArrayList to Synergex.SynergyDE.Collections.ArrayList, which remains 1-based.

  Typically, these references to ArrayList will appear in two places: on imports at the beginning of source files and on ArrayList handle declarations where you've specified @System.Collections.ArrayList (or @Collections.ArrayList if you've imported the System namespace).

  If you are choosing step 2 above, you can change the IMPORT statement to use the Synergex.SynergyDE.Collections namespace instead, and if you have handles referencing System.Collections, you can change those to reference Synergex.SynergyDE.Collections instead. (9.1.5)

- The IndexOf and LastIndexOf methods in both System.Array and System.Collections.ArrayList now use the array element's Equals method (instead of the target object's) to determine a match. This is to conform to a change by Microsoft to .NET version 2.0. This change has the potential for breaking existing code if the System.Object.Equals method is overridden by objects in the array or the target object and they are different types. (9.1.5)
- (Windows) We improved the performance of WRITES, FORMS, and DISPLAY to an O:S or O:R file. (9.1.5)
- We improved runtime performance of integer FOR statements and same data type stores (for example, A=A and D=D) and made other expression optimizations. See the *Synergy Language Reference Manual* for details on the new optimizations. You must recompile your code to take advantage of these new optimizations. (9.1)
- When accessing a memory handle as a dimensioned structure (when compiled with **-qcheck**), bounds checking now includes the size of the allocated memory, not just the size of the structure. (9.1)
- (Windows, UNIX) We now support relative files larger then 2 GB. On UNIX, this support is limited to modes other than update mode. (9.1)
- The SORT statement now generates a "File in use by another user" error if the input file to sort is open for writing by any process. (9.1)

Synergy ActiveX API

- We moved the documentation for AX_TIMEOUT from the *Synergy Language Reference Manual* to the "ActiveX Routines" chapter of the *UI Toolkit Reference Manual*. (9.5)
- When converting a double to a Synergy type in the ActiveX API, 15 fractional digits are now used. (9.3)
- We added the ability to send ActiveX debug information to an attached Windows debugger by setting the AXDEBUG environment variable to EXTERNAL. If AXDEBUG is set to YES on 64-bit Windows, it is interpreted as EXTERNAL. (9.1.5)

Synergy Configuration Program (synconfig)

- The Licensing tab now displays the backup server name when SynConfig is run on the primary server. (9.5)
- The "Server name" field in the Advanced License Manager dialog is now disabled if the "Be a backup server" check box is checked. This is to prevent accidentally becoming a backup server to another primary server when you meant to become a client of a new server only. To become a backup server of a different primary server, you must uncheck "Be a backup server," OK the dialog, re-access the dialog, change the server name, and then recheck "Be a backup server." (9.5)
- We added "(64-bit)" to the caption of the main **synconfig** dialog and to the About box when running the 64-bit version, so you can tell which version you're running. (9.3)

- We added two new options to the Connectivity Series tab for Oracle 11 and SQL Server (Native Client). (9.1.5b)
- We added the SC_MY9 license mnemonic to the Install Keys dialog. (9.1.3)
- (Windows) We added the SRUN9 license mnemonic to the Install Keys Manually dialog. (9.1.1b)
- The **synconfig** online help is now compiled HTML Help rather than WinHelp. (9.1)

Synergy DBMS

- (Windows, UNIX) We added the READ_CACHE environment variable to enable support for read caching for files opened in I, I:S, and I:R mode in conjunction with the OPTIONS:"/sequential" qualifier. This behavior will become the default in a future version, but you can enable it in 9.3 by setting READ_CACHE to any value. (READ_CACHE was released as an "experimental feature" in 9.1.5b.) Read caching is disabled by any I/O statement except READS.

  According to our tests, performance improves by the following factor with read caching enabled:

System	Improvement
UNIX	2 times
Windows XP and Vista (using a Vista non-system disk)	3 times
Windows Vista/Server 2008 (using the system disk)	10 times
Windows mapped drives	35 times

  When using *xf*Server, set READ_CACHE on the server. (9.3)

- The Select class (and other associated classes in the Select namespace) is now available for selecting records from a file based on fields within the record. See "System-Supplied Classes" in the "System-Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual* for more information. (9.3)
- (OpenVMS) **Fcompare** was enhanced to return an error code to DCL when an error in the operation is detected. (9.3)
- (Windows, UNIX) The **chklock** utility has been added to the ISAM utilities download file, **dbmsutil**. (9.3)

- (Linux) **Dbr.cis** (C-ISAM) now adds trailing and leading compression automatically for keys defined over 14 bytes long. (9.1.5b)
- (Windows, UNIX) We added a new environment variable, SDMS_AUDIT_FILENAME, which filters the audit log by limiting it to operations made on a specific file. (9.1.1b)

Synergy DLL API

- (UNIX) The Synergy DLL API has been updated to report the associated descriptive error text via informational errors. The system error code is unreliable or nonexistent on UNIX (an operating system limitation). Using ONERROR $ERR_CATCH to get the traceback will provide any text related to a failing %DLL_OPEN or %DLL_CLOSE. (9.1)

Synergy HTTP document transport API

- We enabled support for the *timeout* parameter for HTTPS requests in %HTTP_CLIENT_GET and %HTTP_SERVER_RECEIVE. (9.3.1a)
- If the URI contains user information, an authorization header is created with Base64 encoding of the user information. (9.3)
- (UNIX) Setting the fork value to 2 in %HTTP_SERVER_RECEIVE now causes the main accept loop to never time out, although it still passes the timeout to the rest of the HTTP operations. (9.1.3)
- (UNIX) %HTTP_SERVER_RECEIVE in fork mode now returns after the accept instead of after the first receive. The receive after accept now supports a timeout value. (9.1.3)
- We added a new argument (*dont_close*) to %HTTP_SERVER_SEND, which, if passed and nonzero, causes the socket not to be closed when %HTTP_SERVER_SEND completes successfully. (If %HTTP_SERVER_RECEIVE is called after %HTTP_SERVER_SEND does not close the socket, it does not perform a new accept.) (9.1.3)
- (UNIX) %HTTP_SERVER_RECEIVE now supports an optional tenth argument whose presence indicates the Synergy program will create a “forked” child to process the receive request, thus allowing for better scalability. The new syntax is

  ```
  status = %HTTP_SERVER_RECEIVE(instance_id, uri, method, [timeout], handle,
  &                             length, error, [headers], [log_file][, fork])
  ```

  See the *Synergy Language Reference Manual* (DCN SL-00-9104) for more details. (9.1.1b)

Synergy License Compliance Toolkit

- LM_LOGIN arguments #5 and #6 are now optional (as are arguments #7 and #8). (9.3)

Synergy .NET assembly API

- We added a second version of **gennet.exe**, called **gennet40.exe** (with an accompanying **gennet40.exe.config**), which can load both 4.0 and 2.0 CLR assemblies. (**Gennet.exe** can only load 2.0 CLR assemblies.) (9.3.1b)
- We added the SYNNET_CLR environment variable to specify the version of the .NET Common Language Runtime (CLR) that the Synergy .NET assembly API should use. (9.3.1b)
- Strongly typed parameters are now allowed on delegates when the ADDHANDLER statement is being used. (9.1.5)
- We added the Synergy .NET assembly API, which enables you to interface to .NET from a non-.NET environment, so your existing code can take advantage of its features without migrating to .NET. Using the .NET assembly API, you can load a .NET assembly; instantiate types defined in that assembly; and communicate with the methods, properties, fields, and events of those objects. See the "Synergy .NET Assembly API" chapter of the *Synergy Language Reference Manual* for more information. (9.1.3)
- The Synergy .NET assembly API has a debugging system equivalent to that of the Synergy ActiveX API. See "Debugging" in the introduction to the "Synergy .NET Assembly API" chapter of the *Synergy Language Reference Manual* and SYNNET_DEBUG environment variable in the "Environment Variables" chapter of *Environment Variables and System Options*. We recommend that you use SYNNET_DEBUG when debugging the failure to load an assembly. (9.1.3)

Synergy Prototype utility

- The Synergy Prototype utility has a new option, **-single**, which provides the ability to compile one file at a time. This is especially useful if you have many **.dbl** files that have no interdependencies, because it enables you to quickly create prototypes when a standard **dblproto** run takes too much time. (9.1.1b)
- In previous versions, if you had a function without a declared return type, the first FRETURN of an implied-decimal type returned ID and an FRETURN of an implied-packed type returned IP. We changed these to "D." and "P.", respectively. (An implied-numeric type returns "N.".) (9.1.1b)
- (OpenVMS) The asterisk character is now supported as a wildcard in input filenames. (9.1.1b)

Synergy windowing API

- (.NET) We added support for scaling the font used in a .NET application that makes use of low-level windows or the Toolkit. If SYN_RESIZE_SCALE is set to 1 in the environment, you may resize, maximize, or restore the form used for window/Toolkit display, and the font will be scaled accordingly. (9.5.3)
- (.NET) We added row leading (extra space between rows) to match the metrics used in traditional Synergy on Windows. You can control this leading via the MINIMIZE_LEADING environment variable. (9.5.1b)

- We added the ability to turn the horizontal and vertical scroll bars of a window on and off independently of one another. The new syntax for the WB_SBON and WB_SBOFF subfunctions of the W_BRDR routine is

  `xcall w_brdr(`*window,* `WB_SBON`*[, scroll_bar]*`)`

  and

  `xcall w_brdr(`*window,* `WB_SBOFF`*[, scroll_bar]*`)`

  where *window* is the ID of the window and *scroll_bar* is one of the following optional scrollbar identifiers:

  WBS_HORZ Visibility of horizontal scroll bar is affected.

  WBS_VERT Visibility of vertical scroll bar is affected.

  WBS_BOTH Visibility of both scroll bars is affected. (default)

  (9.3)

Synergy Windows printing API

- (.NET) We added support for the print previewer, which is accessed via DWP_PREVIEW in the %WPR_EXECUTE routine. The optional help routine argument to DWP_PREVIEW is ignored. (9.5.1a)
- We added the DWP_FILL subfunction to %WPR_PRINT, which enables you to fill a rectangle of a report page with a background color, optionally hatched. (9.3)
- We added the ability to specify the orientation and escapement of fonts. This enables you to rotate characters relative to the text baseline as well as to rotate the text baseline relative to the device's x-axis. (9.3)
- We added mousewheel scrolling support to the print preview window. (9.3)
- We added the ability to specify image transparency on a per-report basis with the DWP_TRANSPARENCY subfunction to %WPR_PRINT. It has the syntax

  `DWP_TRANSPARENCY,` *transparent_color[, threshold]*

  (9.3)

- We added the ability to query the status of the selected printer with the DWP_GETSTATUS subfunction to %WPR_INFO. It has the syntax

  status = `%wpr_info(`*report_handle,* `DWP_GETSTATUS)`

  (9.3)

- (Windows) Error checking is now performed on all WIN32 API calls in DWP_PRINT, and appropriate Synergy errors are generated in exactly the same way that LPQUE performs the error checking. (9.1.3)

Synergy XML API

- Special characters (ASCII 127 - 255) are now encoded using XML decimal encoding. (9.5.1a)
- We added a PARSER option to %XML_OPTION that enables you to keep leading spaces in element text instead of stripping them when calling %XML_ELEM_GETTEXT and %XML_ELEM_GETTEXTHANDLE. Before parsing the XML file with either %XML_PARSER_PARSEFILE or %XML_PARSER_PARSESTRING, call

  ```
  %XML_OPTION("PARSER", SYNPARSER_KEEP_LEADING_SPACE)
  ```

  To strip the leading spaces, use the default parser option:

  ```
  %XML_OPTION("PARSER", SYNPARSER_DEFAULT)
  ```

  (9.5)
- The Synergy XML API is less likely to run out of memory with large XML files, and it uses about the same memory footprint as commercially available DOM parsers. In addition, we improved performance in loading large XML files. (9.1.5)

Utilities

- (OpenVMS) We made several enhancements to the **servstat** program. The servers are numbered in the list, making it easier to select the one you want to view. In addition, the list now distinguishes between *xf*Server and *xf*ServerPlus. The display status shows additional information when encryption is enabled: whether master or slave, the cipher level (low/medium/high), and the certificate file being used. (9.5.3)
- The **dblproto** utility has a new **-N** option, which processes decimal type arguments as numeric (equivalent to setting system option #28). (9.5.1a)
- **Synckusr** now indicates if the user being checked is part of the administrators group. (9.5)
- The **dblproto** utility now displays a warning message (for example, "%DBLPROTO-W-SKIP: Error on Method: skipping MyClassMethod") when it skips processing part of a file when **dblproto** encounters an error. (This tracker was fixed in Windows and UNIX in version 9.3.1b.) (9.5)
- We added command line redirection to **dbl2xml** to enable you to specify a list of command lines in a single file. For example,

  ```
  dbl2xml <file.lst
  ```

  or

  ```
  dbl2xml -T <file.lst
  ```

  Note that **-T** is only supported on Windows and UNIX. (9.5)
- The **dbl2xml** utility no longer causes a segmentation fault in the C++ runtime. (This tracker was fixed in Windows and UNIX in version 9.3.1b.) (9.5)

- The maximum number of arguments that may be passed on a single line to **dbl** and **dblproto** from a redirected input file has been increased from 125 to 4000. The maximum number of characters that may be passed on a command line to **dbl** and **dblproto** has been increased from 32K to 128K. (9.3)
- We have improved error handling for read-only files in **synctl**. These conditions now issue a "File is read-only" message and return to the menu. (9.3)
- We added command line redirection to **dblproto** such that a list of command lines can be specified in a single file. For example,

  ```
  dblproto <file.lst
  ```

  or

  ```
  dblproto -T <file.lst
  ```

  (9.3)
- **Dblproto** now outputs the filename when a compiler error occurs. (9.3)
- We added the **dbl2xml** utility, which processes attributed Synergy code into an XML file that can be imported into the SMC. This feature enables you to populate the SMC without using the MDU to perform data entry, and it should make it easier to keep your code and SMC in sync. (9.3)
- We significantly reduced memory usage when **dblproto** is run on large projects. (9.1.5a)
- The **synbackup** utility now supports relative files. (9.1.5a)
- (Windows, UNIX) The **synbackup** utility now displays descriptive usage information. In addition, you can now specify a wait time of -1 to indicate that **synbackup** should wait until the backup can be performed. (9.1.5)
- (UNIX) The **synbackup** utility now generates a unique shared memory key instead of a fixed "999" ID. The purpose of this change is to avoid conflicts with other non-Synergy programs using the same key. In addition, users without administrative privileges can now perform query operations (**-q**). (9.1.5)
- A ToString method is generated with the NEW modifier for all **gennet** classes. This method maps to the ToString of the object in the .NET common language runtime. (9.1.5)
- **Gennet** has a new command line option, **-impdir** *directory*, that enables you to specify a directory or logical for which **gennet** will build an import list. The new **gennet** syntax is

  ```
  gennet -output output_file -log log_file
  assembly [assembly...][ -nodep][ -impdir directory]
  ```

  Gennet adds the specified directory or logical to the end of the generated IMPORT statement, as follows:

  ```
  IMPORT System DIRECTORY 'MYLOGICAL:'
  ```

  (9.1.5)

- When using code generated by **gennet**, you can now cast from an object property, a field, or a method return value to a strongly typed object. (9.1.5)
- We added the **gennet** utility, which generates Synergy classes that wrap the classes defined in a .NET assembly. **Gennet** has the following syntax:

  `gennet -output` *output_file* `-log` *log_file assembly [assembly …][* `-nodep`*]*

  See "The Gennet Utility" in the "General Utilities" chapter of *Synergy Language Tools* for more detailed information. (9.1.3)
- The Variable Usage utility is now launched with **dbs** instead of **dbr**. (9.1.3)
- We added a new prototype generation utility (**dblproto**) that allows you to strongly prototype your own routines and functions. See the *Synergy Language Reference Manual* for details. (9.1)

Visual Studio integration

- We introduced Visual Studio integration, which provides templates for Synergy/DE projects, a C#-to-Synergy code converter, and other features. (9.5)

Build

- The output path on the Build property page now supports environment variables using the $(*xxx*) format (e.g., $(BIN)\myLoc). (9.5.3)

Debugger

- (NET) We added two new Synergy-specific debugging commands for the Watch and QuickWatch windows:
  - ^ADDR(*expression*), which displays address data for a descriptor type variable.
  - ^M(*[structure_field,] handle*), which displays the value of a specified location in the memory handle.

  Additionally, we removed support for three other commands that are no longer necessary: show globals, show statics, and show address. The information displayed by the show globals and show statics commands is now displayed automatically by the debugger, and the show address command has been replaced by ^ADDR. (9.5.3)

IntelliSense

- IntelliSense now shows parameter direction (INOUT/OUT/REF) where appropriate. (9.5.3)
- If a project has already built its IntelliSense symbols, when you shut it down and reopen it, those symbols will be used until the next IntelliSense build has completed. (9.5.3)
- We added support for extension methods. (9.5.3)

Project

- We added support for ClickOnce deployment: we added a Publish page and a Runtime Settings page to the Project Designer (i.e., project properties), and we added support for the Publish Wizard (which you can open from the Publish page). (9.5.3)
- We added a new option to the Interop property page that enables you to turn off adding the XFPL support methods to a generated component class. By default, this option is checked, to include the XFPL support methods. (9.5.3)
- Synergy Language Integration now offers better support for the Windows Service project type. The designer for the service is functional and can be used to add the Installer. We have also added item templates for Installer and Service types with included designer support. (9.5.3)
- We now support Add Service Reference. When in a project, right-click the project node or References folder and select "Add Service Reference". (9.5.3)
- We now support datasets, which enables you to generate dataset code by using the DataSet wizards. (9.5.3)
- We now support Add Service Reference using **svcutil**. (9.5.1a)

*xf*NetLink Synergy

- We added support for encryption of network packets sent and received between *xf*ServerPlus and *xf*NetLink Synergy. (9.3)
- To support encryption, we added an optional argument to %RXSUBR and RX_SETRMTFNC. For methods requiring slave encryption, "/encrypt" is added to the end of the method ID argument. (9.3)
- The *xf*NetLink Synergy client routines no longer sleep for 1 second during a close connection request (RX_SHUTDOWN_REMOTE). (9.1)

Version 8

This section briefly describes new features in Synergy Language version 8 and the changes in this version that may break your code. For information on the changes you may need to make to your system and code when you upgrade, see the *Synergy/DE Quick Migration Guide*.

Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Synergy Language version 8.

- (UNIX) The SHELL subroutine now correctly handles shell environment variables ($DBLDIR). This modification may break your code if you pass Windows-style commands to the UNIX shell containing "\\". You may be able to workaround this issue by using "\\\\" instead. (8.3)
- (Windows, UNIX) In preparation for Synergy/DE version 9, changes were made to our executable file headers. As a result, all **.elb** and **.dbr** files generated with version 8.3 are not backward compatible. The minimum **.elb** and **.dbr** version that is upward compatible with version 8.3 is 6.1. This modification may break your code. (8.3)
- Also, in preparation for Synergy/DE version 9, internal changes were made to our object file format. As a result, all **.dbo/.obj** and **.olb** files generated with version 8.3 are not backward compatible. This modification may break your code. (8.3)
- (Windows, UNIX) We added the ability to link an ELB to other dependent ELBs. As a result, all **.elb** files generated with version 8.3 are not backward compatible. The minimum ELB version that is upward compatible with version 8.3 is 6.1. This modification may break your code if you have ELBs that were built with a pre-6.1 version. See "Compiler, linker, librarian" on page 1-36 for more information. (8.3)
- (Windows) If you are on version 7.5 and are using the shared installation feature, you must save off your **setup.ini** customizations before you install version 8.3. See "Installation" on page 1-38 for more information. (8.3)
- (Windows) We improved Synergy/DE's color support to accommodate Windows system colors. This modification may break your code. It includes changes to the return values of the WI_PALET subfunction to W_INFO that may break your code, and it includes changes to the default palette that may change the appearance of applications that use the Synergy Language windowing subroutines. See "Synergy Language windowing subroutines" on page 1-55 for more information. (8.3)
- (UNIX) The behavior of the INTRAFILELOCKS environment variable has changed. This modification may break your code. See "Runtime" on page 1-46 for more information. (8.3)

- The Synergy HTTP document transport API now automatically escapes and unescapes certain characters in URIs. If you had already implemented your own escape mechanisms prior to this feature being added, this modification may break your code. See "Synergy HTTP document transport API" on page 1-53 for more information. (8.3)
- (32-bit systems) The runtime generates a new error, $ERR_BIGALPHA, if an alpha string concatenation results in a combined size of 65536 or greater. This modification may break your code. (8.3)
- The compiler will now generate a BIGNUM error on an implied-decimal field if the places to the left of the decimal are greater than 18. If the implied-decimal field is defined with a space between the **d** and the field size, and the number of places to the right of the decimal point is specified as 10, a BIGNUM error may be generated if the field size is greater than 19. This modification may break your code. (8.3)
- (Windows, UNIX) The %DLL_SUBR function has been removed. This modification may break your code. See "Synergy DLL API" on page 1-53 for more information. (8.3)
- If you wrap the Synergy/DE installations, please contact Synergex for a current copy of the documentation. Our upgrade to InstallShield 10.5 affects the **setup.ini** file that you may be redistributing. Read the **readme.txt** file that is part of the wrapped installation documentation. (8.3)
- (Windows, UNIX) The following modification may break your code. The **fconvert** utility is now affected by the ISAMC_REV environment variable in one additional scenario. See "ISAM" on page 1-43 for details. (8.1.7a)
- (Windows) If you use the "shared installation" feature or wrap your Synergy/DE installations, the following modification may break your code. We have upgraded to InstallShield 9. InstallShield 9 requires a new version of the InstallScript engine and requires modifications to the **setup.ini** file. See "Installation" on page 1-38 for details. (8.1.7)
- (Windows) We changed the default value of the ActiveX container property ClipChildren to TRUE. See "Runtime" on page 1-46 for details. (8.1.7)
- (Windows) The system code for 64-bit Windows has been changed from 301 to 103. This is to support a more consistent system-code naming scheme on Windows and OpenVMS as we add new platforms during the coming year. (The UNIX system-code naming convention is slightly different with regards to 64-bit platforms.) (8.1.5b)
- (Windows, UNIX) The COPY subroutine now generates a "File in use" error if the destination file exists and is open by another process. Since this was previously permitted, an existing application may now fail due to this change. (8.1.5)
- (UNIX) On those UNIX systems (including later Linux systems) that return hex TTYNAME (for example tty5a23), the %TNMBR value is now correctly determined. This modification may break your code as it may change the terminal numbers returned on a particular system. (8.1.5)
- (UNIX) When the **/etc/ttys** file is missing, %TNMBR now correctly creates a terminal number from the device name even it contains unrecognized characters. This modification may break your code as it may change the terminal numbers returned on a particular system. (8.1.5)

- (Windows) If you use the "shared installation" feature, the following modification may break your code. InstallShield now distributes the **isscript8.msi** file instead of the **isscript.msi** file with all installation packages. When you install/upgrade the "Files for shared installations" component, note that this new file will appear in the synergyde\client directory and the old one will not.

  Because this **.msi** file is referenced within the **setup.ini** file that is associated with the Synergy/DE Client installation, the contents of the distributed **setup_base.ini** file have been modified. If you are using the shared installation feature of version 8.1.1, you will need to merge your **setup.ini** command line settings with our 8.1.3 **setup_base.ini** file changes. After installing Synergy/DE on the shared machine, copy your command line settings ("CmdLine=") from your **setup.ini** file to the new **setup_base.ini** file. Then, do a "save as" to rename **setup_base.ini** to **setup.ini**, overwriting your old file in the process.

  Important: Because **setup.ini** was an "installed" file in version 7.5, it will be removed when you uninstall 7.5 or upgrade. If you are currently on version 7.5, *you must save off your* ***setup.ini*** *customizations before you install version 8* and/or *re-add them to the version 8* ***setup.ini*** *file after installation*. (8.1.3)

- If an *xf*Server file specification (@) is provided for any filename passed to the Windows printing API, an error $ERR_FILSPC is now generated. (8.1)
- (Windows 2000/XP) We changed the default installation directory for Synergy/DE to C:\Program Files\Synergex\SynergyDE. The new default will be used on new installations. If you are upgrading, the installation will default to your current Synergy/DE directory. (8.1)
- (Windows) The Synergy/DE installation no longer creates, nor updates, a **synergy.ini** file during installation. A default **synergy.ini** file (containing entries referencing %SYNERGYDE%) is now distributed in the dbl directory. This file should be copied to another directory and SFWINIPATH set to that directory, as it will be removed on an uninstall or upgrade. Also, because the distributed **synergy.ini** file is intended to be an example file, it will not be copied to **old_synergy.ini** before a new version is installed. If you have a previous version of Synergy/DE, you may need to make changes to accommodate the above enhancement:
  - If you are using SFWINIPATH to point to the directory where your **synergy.ini** file resides, and it is not synergyde\dbl, no changes are required.
  - If you are using SFWINIPATH to point to the directory where your **synergy.ini** file resides, and it *is* synergyde\dbl, you should copy the file to another directory (preferably outside the synergyde tree) and set SFWINIPATH to that directory.
  - If you are using the **synergy.ini** file from the Windows directory (where pre-7.5.3 installations placed the created file), you should move it to another directory (not synergyde\dbl and preferably outside the synergyde tree) and set SFWINIPATH to that directory.

  (8.1)

- (Windows) The Synergy/DE Client installation no longer supports a local **synergy.ini** file. (Previously it created and/or modified a local file under certain circumstances.) Now, it sets a local SFWINIPATH environment variable pointing to the **synergy.ini** file on the shared machine, as specified in the **setup.ini** file. Local settings are written to the new **synuser.ini** file on the client machine.

  If you are using a local **synergy.ini** file on your Synergy/DE Client machines, you should remove the system-specific settings from the local **synergy.ini** file and place them in a **synergy.ini** file on the shared machine. Set SFWINIPATH on each client to point to the **synergy.ini** file on the shared machine. For each client, save the user-specific settings from that client's **synergy.ini** file in a file named **synuser.ini** and place it in the local application data directory on the client machine. (8.1)

- (Windows) Customized **setup.ini** files in the synergyde\client directory will be lost when upgrading from version 7.5. (This is the directory created by the "Files for shared installation" feature.) (8.1)

- (Windows) All *xf*Server, *xf*ServerPlus, and SQL OpenNet services are now stopped and unregistered when you uninstall Synergy/DE. To retain your registered services, you should upgrade rather than uninstall/re-install. When performing an upgrade, all services are stopped but not unregistered. Note that if you are upgrading to a different directory, you will need to reregister and restart the services manually. Also note that if you are upgrading from version 7, those installations did not have this same behavior and may have been erroneously unregistering the services upon uninstall. (This is important because an "upgrade" actually runs the "uninstall" logic of the previous installation.) (8.1)

- (Windows) The syntax for wrapped installations has changed. If you wrap any of the Synergy/DE installations, please contact Synergex for an updated copy of the documentation and a description of the changes. (8.1)

- We moved the Synergy HTTP document transport API into the Synergy runtime. This means that you no longer need to link with **synxml.elb** to use this API. In making this change, we modified all HTTP API routines to be macros that invoke the F_SYNHTTP runtime routine. If you are using the HTTP API, you must recompile and relink with version 8. (8.1)

- Invalid MATCH, DIRECTION, POSITION, and LOCK values now signal an E_OUTRNG error. (8.1)

- Non-Synergy programs using the Synergy Database API directly to access SDMS_ISINFO will receive a DE_INVOPT error when requesting FILENAME until they are recompiled with the new **sdms.h** header file. (8.1)

- We removed the .INCLUDE of **object.def** within **activex.def**, because it was not needed for the ActiveX interface. (8.1)

- When .INCLUDEing from a repository, where the structure contains nested groups that all define a prefix, the compiler now concatenates the prefixes from all levels instead of appending only the lowest level prefix to the fields. Previously, non-unique field names were possible, which is what the prefix feature was designed to prevent. (8.1)

- (Windows) The %TTSTS routine now correctly returns 1 if the channel is not open to a terminal or input is redirected. This modification may break your code if you depended on the previous behavior. (8.1)
- (Windows) The PARSE routine now places any *\\server\share_name* portion of a UNC path specification in the *device* return variable instead of the *directory* variable. (8.1)
- Parent/child hierarchies are no longer supported in the Synergy namespace API. (8.1)

New features

- (Windows) Version 8.1.3 has been certified for use with Windows Server 2003. Supported products for this platform include Synergy/DE, *xf*NetLink COM, *xf*NetLink Java, *xf*NetLink .NET, and Online Manuals. (8.1.3)

C interface

- (Windows, UNIX) We added a feature to the DLL interface to assist you in creating your own C DLLs that interact with the runtime C interface. This eliminates the need to relink the runtime with your C routines. (8.1)
- (Windows) When you use the C interface, the **windows.h** file no longer needs to be included when **xcall.h** is included, as it will be included through **machine.h**. (8.1)

Compiler, linker, librarian

- (Windows, UNIX) In preparation for Synergy/DE version 9, changes were made to our executable file headers. As a result, all **.elb** and **.dbr** files generated with version 8.3 are not backward compatible. The minimum **.elb** and **.dbr** version that is upward compatible with version 8.3 is 6.1. This modification may break your code. (8.3)
- Also, in preparation for Synergy/DE version 9, internal changes were made to our object file format. As a result, all **.dbo/.obj** and **.olb** files generated with version 8.3 are not backward compatible. This modification may break your code. (8.3)
- (Windows, UNIX) We added the ability to link an ELB to other dependent ELBs. You can now specify one or more ELBs as input files to the **dblink -l** command, and these ELBs will be automatically opened at runtime when the newly created ELB is opened. **Dblink** also has a new **-r** option that can be specified in conjunction with **-l** (i.e., when creating ELBs). The **-r** option does not allow unresolved references to routines or symbols in an executable library.

  In addition, the **listelb** utility has two new options: **-e**, which displays the list of ELB names only, and **-l**, which includes information for each of the ELBs linked to the specified ELB.

  As a result, all **.elb** files generated with version 8.3 are not backward compatible. The minimum ELB version that is upward compatible with version 8.3 is 6.1. This modification may break your code if you have ELBs that were built with a pre-6.1 version.

  Be careful when rebuilding existing ELBS to be linked to other ELBs, as it can affect the order of routine resolution at runtime. (This is a result of automatically opening linked ELBs.) (8.3)

- (Windows) When a backup license is retrieved, it always includes a mandatory timeout (starting at 21 days). The Synergy compiler will no longer display the copyright message during the first two weeks that a backup license server is enabled, only during the third and final week. (8.1.7c)
- (OpenVMS) An unaligned access was removed from the compiler, resulting in a slight speed improvement. (8.1.7a)
- We improved the performance of the compiler when performing multiple .includes from a repository. This performance improvement is most notable on OpenVMS. (8.1.7a)
- (Windows, UNIX) Previously, when an .ALIGN directive resided within a GLOBAL DATA SECTION that did not have a ",INIT", and the object file was linked into an ELB, unpredictable results could occur because the compiler had erroneously emitted data over other pieces of compiler information. In the reported situation, a CASE statement was not executing properly. This has been corrected. *We strongly recommend that all source modules that contain a global data section that does not have a ",init" but does contain an .align directive be recompiled*. (8.1.5)
- Previously, when the .PROC statement was not in the primary source file, the line numbers in a traceback could be inaccurate and sometimes negative. This has been corrected. *We recommend that programs that do not have the .proc statement in the primary source file be recompiled*. (8.1.5)
- An .INCLUDE from the repository now allows the specification of a field prefix. The syntax is PREFIX="*prefix*". (8.1)
- (Windows, UNIX) The maximum number of ELBs that can be linked against is now 256. (8.1)
- (64-bit systems) The compiler now allows fields and named records to exceed 65,535 on 64-bit systems (system codes 301, 021, 320, 304, and 309). (8.1)
- (OpenVMS) We added optimizations for text file processing in the compiler. (8.1)
- Additional review level values have been added to the variable usage reporting facility of the compiler. They are now addable bit flags:

  1 = global variable references
  2 = label references
  4 = include file references
  8 = primary file only
  16 = process local routines for references

  (8.1)
- The variable usage reporting facility of the compiler has been enhanced to identify variables and labels referenced between the BEGIN_LOCAL_ROUTINE and END_LOCAL_ROUTINE compiler directives when the review level is specified as 2. (8.1)
- The compiler has been modified to *not* generate an error on exact duplicate EXTERNAL FUNCTION declarations. (8.1)
- We improved the performance of the compiler when .INCLUDEing from the repository. (8.1)

Debugger

- (Windows, UNIX) The OPENELB command now opens not only the specified ELB but also all ELBs that are linked to it. See "Compiler, linker, librarian" on page 1-36 for more information. (8.3)
- The debugger command SHOW DYNMEM now displays the handle numbers associated with the dynamic memory handles. (8.3)
- We added the DYNMEM option to the SHOW debugger command. SHOW DYNMEM displays all dynamic memory segments that are currently in use. NOTE: If you're using UI Toolkit, we recommend using this after the U_FINISH routine. If the amount of output is large, it is better to use the remote Telnet debugger with a recall buffer. (8.1.7)
- (Windows, UNIX) The remote debugger is now supported across program chains. (8.1.7)
- (Windows, UNIX) The Synergy Language debugger can now debug remotely. This may be desirable when, for example, the runtime is running non-interactively under *xf*ServerPlus or with an HTTP server application, or in any situation where the local debugger interferes with the focus of the application being debugged. The command to invoke remote debugging is

  `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 120), and *program* is the name of your compiled and linked Synergy Language program. To be able to access symbols, you still need to compile and link with the **-d** option, as you would for local debugging. (8.1.5)
- We added the OPENELB debugger command, which makes the subroutines in an ELB available to the debugger and the executing program. The syntax is

  `OPENELB` *elb_spec*

  where *elb_spec* is the file specification of the ELB to open. (8.1.5)

Installation

- (Windows) If you are currently running a shared installation of version 8.1 or a shared installation of version 7.5 without Connectivity Series, *you do not have to upgrade or uninstall/reinstall the clients* for version 8.3. You simply need to upgrade (or uninstall/reinstall) the shared machine.

  If you are currently running a shared installation of version 7.5 *with* Connectivity Series, you must set the VORTEX_HOME environment variable added in version 8.1 on each client. You can either set VORTEX_HOME manually or you can upgrade or uninstall/reinstall each client. We *strongly* recommend that you set the environment variable manually to avoid the burden of upgrading or uninstalling/reinstalling all of the clients.

  If you use *xf*ODBC and would like to upgrade the MDAC version on the clients, you can install the latest version (2.8) from the Synergy/DE CD or from the Annual License Maintenance Downloads page of the Synergex secured website. (8.3)

- (Windows) We have upgraded to InstallShield 10.5. InstallShield 10.5 distributes the **isscript1050.msi** file instead of the **isscript9.msi** file with all installation packages. When you install/upgrade the "Files for shared installation" component, note that this new file will appear in the synergyde\client directory and the old one will not. This change affects the distributed **setup_base.ini** file. (8.3)
- (Windows) The **setup_base.ini** file has changed. If you are using the shared installation feature of Synergy/DE version 8.1, the version 8.3 installation will update your customized **setup.ini** file automatically with the changes. First, the version 8.3 installation will rename your current **setup.ini** file to **setup_old.ini**. It will then create a new **setup.ini** file by copying the distributed **setup_base.ini** file. Finally, it will merge your customized command line settings from **setup_old.ini** ("CmdLine=") into the new **setup.ini** file.

  IMPORTANT: If you are using the shared installation feature of Synergy/DE version 7.5, the version 8.3 installation will *not* update your customized **setup.ini** file. Because **setup.ini** was an "installed" file in version 7.5, it will be removed when you uninstall 7.5 or upgrade. All customizations will be lost. If you are currently on version 7.5, *you must save off your **setup.ini** customizations before you install version* 8.3. After installing version 8.3 on the shared machine, copy your command line settings ("CmdLine=") from your saved-off **setup.ini** file to the newly installed **setup.ini** file. (8.3)
- (Windows) We no longer support installing on the following systems: 98, Me, NT. (8.3)
- (Windows) On a new installation, the Synergy/DE and Synergy/DE Client installation programs no longer set the VSLICKCONFIG environment variable. On an upgrade installation, these programs retain the previous VSLICKCONFIG setting. (8.3)
- (Windows) We removed Synergy color settings and the palette setting from the distributed **synergy.ini** file. In previous versions, the Synergy runtime defaults were used only if you removed these overrides or used a different **synergy.ini** file. (8.3)
- (OpenVMS) The installation has been modified to allow either one or two servers to be configured at install time: one *xf*ServerPlus and/or one *xf*Server. (8.3)
- (OpenVMS) The installation will no longer put /FREE_POOL=0 in the SYNERGY_STARTUP.COM file when *xf*Server data access is not requested; nor will it put /XFPL_FREE_POOL=0 in the SYNERGY_STARTUP.COM file when *xf*ServerPlus is not requested. This setting was used to "turn off" the unwanted service, but it was causing the desired process to fail on start-up because, starting with 8.1.7a, a free pool setting of 0 is not permitted. (8.3)
- (OpenVMS) The installation script no longer includes the /nosecure and /inactive_limit switches in **synergy_startup.com** when *xf*ServerPlus is configured. (8.3)
- (Windows) The **genxml** utility is now also distributed with *xf*Series. Previously, if you installed only *xf*Series, you were unable to use the MDU's Export Methods feature because it requires **genxml**. (8.3)

- (Windows) We now only install **gdiplus.dll** on Windows 98, Me, NT, and 2000. (8.1.7c)
- (Windows) We updated the version of the **gdiplus.dll** installed with Synergy/DE to the level required by Microsoft security hotfix MS04-028. (8.1.7c)
- (Windows) The Synergy/DE and Synergy/DE Client installations have been modified to abort if the target system has Synergy/DE *xf*ODBC Client installed. (8.1.7c)
- (Windows) Version 8.1.7a includes the initial release of the new *xf*ODBC Client installation. This installation does not require or install any Synergy/DE components, other than those components required for an *xf*ODBC client. (8.1.7a)
- (Windows) If you use the "shared installation" feature or wrap your Synergy/DE installations, the following modification may break your code. We have upgraded to InstallShield 9. InstallShield 9 distributes the **isscript9.msi** file instead of the **isscript8.msi** file with all installation packages. When you install/upgrade the "Files for shared installations" component, note that this new file will appear in the synergyde\client directory and the old one will not.

  Because this **.msi** file is referenced within the **setup.ini** file that is associated with the Synergy/DE Client installation, the contents of the distributed **setup_base.ini** file have been modified. (Other changes have been made to the **setup_base.ini** file as well.) If you are using the shared installation feature of Synergy/DE version 8, you will need to merge your **setup.ini** command line settings with our 8.1.7 **setup_base.ini** file changes. After installing Synergy/DE on the shared machine, copy your command line settings ("CmdLine=") from your **setup.ini** file to the new **setup_base.ini** file. Then, do a "Save As" to rename **setup_base.ini** to **setup.ini**, overwriting your old file in the process.

  IMPORTANT: Because **setup.ini** was an "installed" file in version 7.5, it will be removed when you uninstall 7.5 or upgrade. If you are currently on version 7.5, *you must save off your setup.ini customizations before you install version 8*. After installing version 8 on the shared machine, follow the instructions above to merge your **setup.ini** command line settings into the version 8.1.7 **setup.ini** file. (8.1.7)
- (Windows NT) As instructed in the Microsoft KB article regarding registry entries that should be set for Windows NT Terminal Server, we now set the MsgQBadAppSleepTimeInMillisec entry to 0, rather than 1. (1 is the default.) (8.1.7)
- (Windows) We now distribute Microsoft's redistributable **gdiplus.dll** with Core Components. This DLL contains the GDI+ API, used by the new runtime image support feature, as documented in "Synergy Windows Printing API" on page 1-56. (8.1.7)
- (Windows) We now include our company name in the Copyright information in the Properties dialog for versioned files. (8.1.7)

- (Windows) We now add Synergy/DE environment variables to the end of the PATH, rather than the beginning. Previously, we inserted them at the beginning to ensure that, should PATH contain orphaned Synergy/DE environment variables from bugs in earlier uninstallations, the first one encountered (and hence used) would always be the most current one. Starting in version 8.1, Synergy/DE has been correctly removing environment variables from PATH during uninstallation, so installation can now safely add settings to the end. (8.1.7)
- (Windows) The following files are no longer distributed with the Examples component: **wbldism.***, **wisload.***, and **wstatus.***. (8.1.7)
- (Windows) We no longer distribute the **axctrls.pas** file with Professional Series. This was for use with controls built in Delphi 3. We recommend you use Delphi 4 or higher. (8.1.7)
- (Windows) The Synergy/DE installation (Connectivity Series component) and Synergy/DE Client installation now install the current MDAC version only if the target system does not have MDAC 2.71 Service Pack 1 or higher or SQL Server Cluster already installed. The currently distributed MDAC version is 2.8. On NT, 2000, and XP (with no service packs), the user will be prompted to reboot. If the target system already has the minimum required MDAC version for Synergy/DE, but you would like to upgrade to version 2.8, the **mdac_typ.exe** file is available on the Annual License Maintenance Downloads page and the Synergy/DE software CD. (8.1.7)
- (Windows) We now distribute the **chklock** utility on Windows. (8.1.7)
- (OpenVMS) We now install **synxfpng** with Synergy Language (in [dbl.bin]) instead of with *xf*Server/*xf*ServerPlus (in [server]). In addition, the installation now creates a verb for **synxfpng** so that you no longer have to create a symbol for it. (8.1.7)
- (Windows) For local or shared installations of Synergy/DE, the installation/upgrade of MDAC is now associated with the Connectivity Series component, rather than with Core Components. (8.1.5)
- (Windows) The Synergy/DE installation (Connectivity Series component) and Synergy/DE Client installation now check for and install MDAC version 2.71 SP1 if necessary. The exceptions to this are on systems where SQL Server Cluster is installed, and on XP and 2003, where MDAC is part of the operating system and can only be updated via an operating system service pack. (8.1.5)
- (Windows) We no longer install/upgrade MDAC on systems where SQL Server Cluster is installed. Microsoft KB article 820754 states, “Do not install MDAC 2.6 or later on any SQL Server 7.0 clustered nodes.” Since MDAC is required for correct Connectivity Series operation, you must ensure that you install at least SQL Server 2000 SP3a in addition to the August security patch before running Connectivity Series on your SQL Server Cluster. (8.1.5)

- (Windows) InstallShield now distributes the **isscript8.msi** file instead of the **isscript.msi** file with all installation packages. When you install/upgrade the "Files for shared installations" component, note that this new file will appear in the synergyde\client directory and the old one will not. See "Features and fixes that may break your existing code" on page 1-32 for more information. (8.1.3)
- (Windows) Additional logging has been added to the Synergy/DE and Synergy/DE Client installations to aid in debugging. (8.1.3)
- (Windows) In the "Files for shared installation" feature, we now distribute a **setup_base.ini**. This file is copied to **setup.ini** if a file of that name does not already exist. (The **setup.ini** file is modified by the user before client installations are performed.) Because **setup.ini** is "copied" and not "installed", it is not removed on an uninstall or upgrade, thereby preserving your customizations. When we have modifications that need to be made to your **setup.ini** file, (such as a new minimum version specification for the Windows Installer), those changes will be reflected in **setup_base.ini**. At that time we will document that the file has changed, and your will be instructed to make the appropriate change to your **setup.ini** file. (8.1)
- (Windows) All installations now check for and install Windows Installer Service version 2.00.2600 if necessary. Additionally, we distribute the updated **instmsia.exe** and **instmsiw.exe** files. (8.1)
- The contents of the distributed **setup_base.ini** file have been modified (from when they were distributed as **setup.ini** in version 7.5). If you have customized the **setup.ini** file, you will need to merge those changes with the changes introduced in version 8. Important: because **setup.ini** was an "installed" file in version 7.5, it will be removed when you uninstall 7.5 or upgrade. *You must save off your customizations before you install version 8 and/or re-add them to the version 8* ***setup.ini*** *file after installation*. Make sure to edit the new **setup.ini** file and not simply replace it with the one you saved off, because other information in **setup.ini** has changed in version 8. (8.1)
- Synergy/DE Client (introduced in version 7.5) was designed so that as new versions of Synergy/DE were released, only the shared machine would need to be updated, not the clients, provided that no new or modified environment variables or registry entries had been introduced. If you are currently running version 7.5.1 clients (Synergy/DE Client) with a 7.5.1 shared installation and do not use Connectivity Series, *you do not have to upgrade or uninstall/re-install the clients for version 8*. You simply need to upgrade (or uninstall/re-install) the shared machine.

  If you *are* using Connectivity Series 7.5.1, there is an environment variable, VORTEX_HOME, that needs to be set on each client. You can either set the new environment variable manually, or you can upgrade or uninstall/re-install each client. We strongly recommend that you set the environment variable manually to avoid the burden of upgrading or uninstalling/re-installing all the clients. (8.1)

- (Windows) If you do choose to upgrade or uninstall/re-install your version 7.5.1 clients to version 8, you must follow the steps below:

  1. Uninstall Synergy/DE Client on all machines accessing the shared 7.5.1* installation.
  2. Save off your customized **synergyde\client\setup.ini** file from the shared machine.
  3. Upgrade the 7.5.1* installation on the shared machine to version 8. (Optionally, you can do an uninstall followed by an install.)
  4. Update the CmdLine= line in the new **synergyde\client\setup.ini** file on the shared machine to what it was in your saved off **setup.ini** file. *Do not copy the file in its entirety.* The **setup.ini** file installed with version 8 contains other required modifications.
  5. Install Synergy/DE Client (from the version 8 client directory) on all machines referenced in step 1 above.

  If you do not follow the steps above, and you upgrade your shared machine "before" you upgrade the clients, you will be required to specify the location of the original (version 7.5.1) **sdeclient.msi** file that was used for the original installation. This file can be found on the version 7.5.1 CD. Additionally, this upgrade of the clients cannot be done in silent mode (a feature of wrapped installations) as all user-interface is suppressed. (8.1)

- (Windows) The Synergy/DE Client installation no longer creates shortcuts for the release notes for License Manager, *xf*Server, and *xf*ServerPlus, as these products cannot be run from the client. (8.1)
- (Windows) We removed the DBLOPT=3 (ISAM file I/O caching) setting from the distributed **synergy.ini** file. If you are setting this in your own **synergy.ini** file, you can now remove it, as that is now the default behavior on Windows. (8.1)

ISAM

- (UNIX) As promised in the 8.1.7 release notes, the behavior of the INTRAFILELOCKS environment variable has changed. By default, intraprocess file locks are now enforced. This means that a file-sharing conflict between two files open by the same process now generates a "File in use" error.

  In version 8.1.5 and 8.1.7, *not* enforcing intraprocess file locks was the default. You could, however, set the INTRAFILELOCKS environment variable to 1 to enable enforcement or just to test it, which allowed you to make any necessary code modifications before version 8.3 was released.

  In version 8.3, you can revert back to the prior default behavior of not enforcing intraprocess file locks by setting the INTRAFILELOCKS environment variable to 0. However, this is not recommended. (8.3)

- If the **fcompare** utility detects a repository field that is defined as decimal when the actual key in the ISAM data file is defined as alpha, it now generates a warning if a range of values that contains negative numbers is defined. (8.3)

- If the **fcompare** utility detects an unsigned field that contains signed data, an "Unsigned field [x] contains signed data" error is generated. This error occurs when the **-dv** option is used with either **-r** or **-g**. (8.3)
- (OpenVMS) Sometimes when **fcompare** generates an error, the program will now return an error status. (8.3)
- (OpenVMS) The ISAMC subroutine now signals an error when the primary key is specified as modifiable. (8.3)
- (Windows, UNIX) ISAM data file corrections applied when using **isutl -ra** have been enhanced for better success on files with a large amount of corruption. (8.1.7a)
- (Windows, UNIX) The following modification may break your code. The **fconvert** utility is now affected by the ISAMC_REV environment variable when the input ISAM file is lower in revision than ISAMC_REV and no description file is used. Previously, ISAMC_REV was only used to create the new file with a higher revision when a description file was specified. (8.1.7a)
- (Windows, UNIX) A new option, **-lx**, has been added to **isutl -v** to better verify a file that is in use. Use of the **-l** option turns off file locking and does not block file access but can result in a false failure when the file is in use. The **-lx** option uses cooperative locking but doesn't require exclusive access; however, all file operations will be blocked until verification is complete. On UNIX only, input-mode reads will not be blocked. (8.1.7)
- (Windows, UNIX) To eliminate a potential ISAM index conflict, a one-minute timer has been added to ISAM update file access. Index blocks that are in memory longer than one minute will be re-read on the next READ. (8.1.7)
- (UNIX) ISAM file opens now detect files on NFS mount points and add additional locks on the data file during a STORE operation to ensure data integrity with concurrent access. (8.1.7)
- (OpenVMS) The ISINFO subroutine now outputs "FIXED" when FOPTS is specified for a file that contains fixed-length records. (8.1.7)
- (Windows, UNIX) Attempting to create a multiple fixed-length record ISAM file with data compression now generates an error. (8.1.7)
- (Windows, UNIX) When using **fconvert**, if a fixed-length output file is specified with multiple input files and the first input file is a fixed-length record file, the record size of the output file will be implied from the record size of the first input file unless the **-r** *recsiz* option is specified for the output file. (8.1.7)
- (Windows, UNIX) **Fconvert** now generates a warning if any output record is truncated due to the input record size being greater than the output record size. (8.1.7)
- (Windows, UNIX) Creating an ISAM file from an XDL file that contains "multiple" as the record format value is now supported. The **ipar** utility now generates valid XDL output when processing a multiple-fixed-length-record ISAM file. The ISINFO routine now generates valid file options when processing a multiple-fixed-length-record ISAM file. Also, the **xdlchk** utility now recognizes "multiple" as a valid XDL record format value. (8.1.7)

- (Windows, UNIX) In the **fcompare** utility, we now include the actual duplicate insert location (ATEND or NOATEND) in the error text when a key in an ISAM file has duplicates but is defined in the repository to not have duplicates. (8.1.7)
- (Windows, UNIX) To improve **isutl** performance (especially when using the **-o** option), the default SORTMEM value has been increased from 1024 to 10240 for 32-bit files and from 10240 to 20480 for terabyte files. This effectively gives the SORT operation 10 MB of memory (20 MB for terabyte files), if necessary. On large, unordered files (greater than 3 million records) with a record size greater than 250, a 30- to 50-percent improvement can be expected when running **isutl -ro**. (8.1.5d)
- (Windows, UNIX) The **fconvert** utility now requires the **-r** option to specify the output file record size when converting an ISAM file with variable-length records to a relative file. (8.1.5d)
- (Windows, UNIX) The **isutl** and **ismvfy** utilities now detect and correct an invalid RFA flag combination that occurred when an ISAM file was created using an XDL file with fixed-length records, static RFA, and no compression (as reported in tr#20747). (8.1.5d)
- (Windows, UNIX) The **isutl** utility now determines (during verify **-v**) the best recovery method for a fixed-length segment ISAM file (one without compressed or variable-length records) when the file contains one or more records of the wrong size. (This is a problem that may occur during a system crash.) Synergy/DE version 7 and higher protects against embedded bad segments by inhibiting STORE, but pre-version 7 files may require indexed recovery using **fconvert** instead of **isutl -r**. (8.1.5)
- (Windows, UNIX) The TBYTE (terabyte) option can now be added to a file by using the **isutl** utility. A new keyword-based option, **-q**, has been added to **isutl -r**. The form is -qfile=*opt[,opt...]* where *opt* can be one or more of the following (or an abbreviation thereof): "compress", "static_rfa", "tbyte", "page=*pagesiz*", and "density=*density%*". (The "density=*density%*" option is similar to the **-p** option, except that it also permanently changes the file density setting.) (8.1.5)
- (Windows, UNIX) A new **isutl** option, **-i**, has been added to launch the information advisor. The information advisor displays helpful advice based on the file organization or content. This advice ranges from preventative actions to performance suggestions. When used by itself ("isutl -i *file*"), the utility does a quick check of static information. To get a full analysis, use the new option with the verify command ("isutl -vi *file*"). (8.1.3)
- (Windows, UNIX) The **isutl** utility re-index command (**-r**) now resets duplicate qualifiers when explicitly ordering the data file (with the **-o** option). Resetting duplicate qualifiers may be required on files where large volumes of records with duplicate key values are deleted and stored. As a result, if qualifiers are not reset, a FILFUL error could eventually be generated by a STORE and have nothing to do with file size. Use **isutl -vm2** to report this. (8.1.3)
- (Windows, UNIX) A new **-t** *tempdir* option has been added to **fconvert** to control where temporary files are created. By default, all temporary files are now created in the current directory. (8.1)

- (Windows, UNIX) When **isutl** is run, all temporary files are now created in the current directory unless a valid temporary directory is specified. (8.1)
- The **fcompare** utility now treats user-defined fields like alpha fields for the purposes of key comparison when running in catalog mode. (8.1)

Runtime

- We added the ^ARGDIM function to retrieve dimension information from a routine argument. (8.3.1d)
- (Windows, UNIX) When an *xf*Server client program performs a remote sort and an error occurs while opening the input file, the filename is now displayed as part of the error. (8.3.1d)
- (Linux 64-bit) Version 8.3.1a is supported on Red Hat Enterprise Linux 4 for AMD64/Intel EM64T. Note: If you choose to rebuild the runtime, you will need the Intel ICL compiler. We recommend that you create a shared object containing your C routines instead, however, and access it via the Synergy DLL API. (8.3.1a)
- (Windows) We added an option to the **dbssvc** service runtime to display the current version:

  ```
  dbssvc -v
  ```

  (8.3.1a)
- (Windows) If a service using the **dbssvc** service runtime is stopped in the middle of an ISAM file update/store/delete, it now waits for that operation to finish before shutting down the service. (8.3.1a)
- (Windows) We added a new flag, D_DETACHED, to the SPAWN subroutine for the **dbs** service runtime. D_DETACHED forces a detached process when SPAWN is called from **dbs** running in a console window. (8.3)
- We added the %SYN_ATEXIT function to register a routine to be called when the following occurs:
  - (Windows) The program executed by **dbssvc** is stopped.
  - (UNIX) A process is stopped by SIGHUP or SIGTERM(-15).
  - (OpenVMS) A process is stopped using the $FORCEX system service.

  (8.3)
- *xf*Server now has the ability to prefetch records in order to increase throughput. By setting the new environment variable SCSPREFETCH, records from successive xfServer READS operations will be prefetched on the client. A new system option, #55, enables you to map Q_NO_LOCK to Q_NO_TLOCK to take advantage of this feature without changing your code. (8.3)
- (UNIX) %SYN_SETDIR now supports UNIX systems in addition to Windows and OpenVMS. (8.3)

- (Windows, UNIX) We added the %SYN_GETSTATE and %SYN_SETSTATE functions, which enable you to determine the current backup mode and alter application behavior when update I/O is frozen due to a backup. (8.3)
- We added the SYN_REPORTEVENT routine to enable a Synergy Language program to write entries to the Windows event log (or to syslog() on UNIX or the operator log on OpenVMS). This routine is most beneficial for use by services or detached programs. (8.3)
- (Windows, UNIX) The OPENELB, %XADDR, and RCB_SETFNC routines now open not only the specified ELB but also all ELBs that are linked to that ELB. See "Compiler, linker, librarian" on page 1-36 for more information. (8.3)
- (Windows) We added a new option, D_NOWINDOW, to the SHELL and SPAWN subroutines. This option enables an XCALL SHELL or SPAWN to be done with no DOS or command prompt window showing anywhere (including the task bar). (8.3)
- For performance reasons, we now use our own API for reading the **synergy.ini** and **synuser.ini** initialization files. (8.3)
- (Windows) We added a new service runtime, **dbssvc**, that allows you to run Synergy programs as Windows services, under control of the Service Manager. (8.3)
- The **dbs** service runtime no longer reads the **synuser.ini** file when the SFWINIPATH environment variable is defined. (Previous versions of the Synergy/DE release notes reported that *xf*ServerPlus scalability was affected by the use of **user32.dll**, a DLL known to limit the desktop heap on Windows. It was reported that the API that reads **synergy.ini** uses **user32.dll**. As it turns out, it is the API that determines the location of **synuser.ini** that uses **user32.dll**. In addition, **dbs.exe** is normally run as a service, and as such, the user hive is not loaded. The user hive must be loaded in order to determine the user's local application data directory. For these two reasons, **dbs.exe** no longer reads the **synuser.ini** file.) (8.3)
- An $ERR_BIGALPHA runtime error (number 14) is now generated on 32-bit systems if the results of an alpha string concatenation are greater than 65535 bytes. This modification may break your code. Previously, if concatenation resulted in a single data item exceeding 65535, the size of that new data item was the resulting size modulo 65535. In other words, an expected combined size of 65540 resulted in only 4 characters. (8.3)
- (UNIX) XCALL RUNJB no longer waits 2 seconds (or the value of JBWAIT) for each job. Henceforth, the JBWAIT environment variable is no longer necessary and therefore is being retired. Several other minor RUNJB problems have also been corrected. Terminal modes are no longer affected by RUNJB. Leading spaces in the command string are now stripped prior to executing the command. (8.3)
- (Windows) XCALL RUNJB("*myjob*"), where *myjob* is a Synergy program, now runs. You no longer need to include the **dbr** command (i.e., dbr *myjob*). (8.3)
- (Windows) Event logging is now possible for non-privileged users of the runtime (**dbr**), and errors that occur when writing to the event log are ignored. (8.3)
- (Windows) A Windows system error is now prefaced with "System error:" to differentiate it from Synergy/DE error text in a fatal traceback. (8.3)

- FIND(Q_SEQ) on a relative or sequential file will now generate an $ERR_FILOPT error. (8.3)
- (Windows, UNIX) %DATETIME has been changed to always fill with zeros instead of spaces if the system resolution for the milliseconds is not six digits. (8.3)
- (Windows) An event log for **dbs.exe** is now available for non-privileged users. (8.1.7e)
- (Windows) The descriptions for **dbr.exe**, **dbs.exe**, and **rsynd.exe** are now unique which will assist with automatic Windows XP Service Pack 2 firewall unblocking. (8.1.7a)
- (Windows) We added a new routine, %SYN_CHARTOSTR, which converts a C-style character pointer (char *) to a Synergy Language alpha type. (8.1.7a)
- A new intrinsic compile-time function, %SYSID, has been added which returns the system ID number of the operating system where the source is being compiled. (8.1.7)
- The performance of the truth test for an alpha variable that exceeds 64 characters has been significantly improved. (8.1.7)
- The default Synergy runtime stack has been increased from 64K to 256K. This allows improved performance when using stack records. On UNIX, the default MAXMEM value has been increased from 1 MB to 1.25 MB to take this into account. on Windows, a MAXMEM minimum of 4 MB is now enforced, even if an explicit MAXMEM value has been specified. (8.1.7)
- A small performance improvement has been made in decimal-to-integer conversions for the STORE (=) statement. (8.1.7)
- (Windows, UNIX) While sorting a file (within the Synergy SORT statement or the sort phases of **isutl** and **fconvert**), if the amount of memory requested cannot be allocated, the minimum SORTMEM amount will be attempted, and if that cannot be allocated, a “Not enough memory” error will be generated. The SORTMEM environment variable sets the maximum amount of memory sort will use. The minimum SORTMEM value is 512 (512K), and the maximum is 32768 (32 MB). (8.1.7)
- The READ and READS statements have a new qualifier: NOFILL. This qualifier causes the data for the record read to be transferred to the user’s data area without filling with trailing blanks. This was used in *xf*Server and the XML API to improve performance. (8.1.7)
- (Windows) The following modification may break your code. We changed the default value of the ActiveX container property ClipChildren to TRUE. This usually reduces flicker when controls are repainted. However, if you find that parts of your control are not being erased properly, set the property to FALSE. (8.1.7)
- (UNIX) The RENAM subroutine can now rename files across file systems. (8.1.7)
- (Windows) The SS_RECVBUF routine now retries transient service failures. (8.1.7)
- (Windows, UNIX) The **stv** argument to the ERROR subroutine and the %ERNUM function is now available on non-OpenVMS systems and returns the last SDMS error number when the last operation was an I/O statement. (8.1.5d)

- (Windows) The following modification may break your code. The system code for 64-bit Windows has been changed from 301 to 103. This is to support a more consistent system-code naming scheme on Windows and OpenVMS as we add new platforms during the coming year. (The UNIX system-code naming convention is slightly different with regards to 64-bit platforms.) (8.1.5b)
- (Windows, UNIX) The **listdbr** utility supports a new **-x** option, followed by the name of one or more ELBs. When this option is specified, **listdbr** loads every module in the main routine and the specified ELBs to see if any modules are missing. (8.1.5)
- (Windows) We reduced the memory footprint of **dbs.exe** by delay-loading the spooling DLLs and certain related system DLLs. This may also reduce desktop heap contention. (8.1.5)
- (Windows) We slightly reduced service runtime overhead when **dbs** is idle. (8.1.5)
- (UNIX) In version 8.1.3, we began enforcing file locking within a process. However, we have since decided to introduce this change more gradually and have therefore modified its implementation to be an optional behavior in 8.1.5, which is controlled using the new environment variable, INTRAFILELOCKS.

  When INTRAFILELOCKS is set to 1, file locking will be enforced among channels within the same process and will generate "File in use by another user" errors as appropriate. (This is the standard behavior on Windows and OpenVMS.) This behavior corrects potential file problems that could result in loss of data.

  Beginning in version 8.3, the behavior described above will become the default behavior. At that time, the INTRAFILELOCKS environment variable can be set to 0 to override this default behavior and enable you to continue executing version 8 and earlier applications that violate these sharing rules.

  In version 8.1.5, INTRAFILELOCKS can be used to test the future version 8.3 behavior and enable you to modify your code before version 8.3 is released. (8.1.5)
- We modified our distributed example programs to use %DLL_CALL instead of %DLL_SUBR. Because we no longer recommend the use of %DLL_SUBR, version 8.1.5 is the last version that supports it. If you are using the Synergy DLL API function %DLL_SUBR, we recommend that you change your code to use %DLL_CALL instead. If you cannot easily modify your code to change all %DLL_SUBR calls to %DLL_CALL, you can instead declare a parameterized macro that redefines %DLL_SUBR as %DLL_CALL. See the example macro below.

  ```
  .define dll_subr(handle, function, arg1, arg2, ...arg20) dll_call
  (handle, DLL_TYPE_STDCALL, function, arg1, arg2, ...arg20)
  ```

  (8.1.5)
- We improved the performance of %TRIM, %ATRIM, and %TRIMZ on strings over 128 bytes. On very large strings, performance is four times greater. (8.1.5)

- The %SS_SOCKET function now validates that the socket argument is at least the same size as the native TCP/IP socket defined by the operating system. For all platforms except system code 301 (Windows 64-bit), this is an **i4**. (8.1.5)
- Synergy/DE no longer requires DNS access (by no longer using gethostbyaddr()) when using an IP address in any of the server or client products. This may result in improved reliability and resilience to DNS problems when using IP addresses instead of names. (8.1.5)
- (Windows) The **makedbr.bat** file now includes instructions for rebuilding the runtime using Visual Studio .NET 2003. (8.1.5)
- (UNIX) The following modification may break your code. If a process doing a SORT already has the output file open, a "File in use" error will be generated. Since this was previously permitted, an existing application may now fail due to this change. This behavior was made optional in version 8.1.5. (8.1.3)
- (UNIX) The following modification may break your code. File locking within a process (that is, when a single process has the same file open on more than one channel) now works the same as file locking between processes. Previously, using the OPEN statement or the routines DELET, RENAM, ISAMC, or ISCLR on a file that had already been opened by the same process on another channel was permitted (albeit risky); now, such an operation will correctly result in a "File in use" error. This behavior was made optional in version 8.1.5. (8.1.3)
- We now support up to 1024 open channels. (UI Toolkit is still limited to 255.) Operating-system limitations may prevent 1024 channels being opened. On OpenVMS, the user parameter FILLM and the system parameter CHANNELCNT need to be increased to at least 1048. On UNIX, there is a hard and a soft limit to the maximum channels a process may open. Both are kernel parameters, and both must exceed 1024 + the number of **dbr** executables, ELBs, and shared objects used by **dbr** + 3 for **stdin**, **stdout**, and **stderr**. (On UNIX and Windows, remember that each Synergy ISAM file requires two channels.) (8.1.3)
- (Windows) The PARSE routine now clears the node argument even if there is no node to return. (8.1.3)
- (Windows) Synergy/DE now supports the concept of a "system-specific" initialization file (**synergy.ini**) versus a "user-specific" initialization file (**synuser.ini**). Settings in the user-specific file override settings in the system-specific file. For more information on this, refer to the "Environment Variables" chapter of *Environment Variables and System Options*. This enhancement enables better support in Terminal Services and Windows XP "limited user" environments. (8.1)
- We added a new function, %SYN_UNAME, which returns the username from the operating system. (8.1)

- The GETFA routine has been enhanced to return the following information:

  RMT—0 if file is local or 1 if remote
  VER—The 7-byte alpha version of the SDMS system (local or remote)
  NDN—1 if the file resides on a BIGNDN system or 0 if LTLNDN
  OST—The operating-system type of the file system: 1 = UNIX, 2 = OpenVMS, 3 = Windows
  ISZ—The native integer size of the operating system: 0 = 32-bit, 1 = 64-bit, 2 = 128-bit

  (8.1)

- (Windows) We have a new icon for the Synergy Runtime. (8.1)
- (Windows) For several versions, the distributed **synergy.ini** file has set system option #3 (ISAM I/O file caching), which improves ISAM performance. This option is now the default and therefore does not need to be set in **synergy.ini** anymore. (8.1)
- We added two new socket API functions: %SS_GETSERVBYPORT returns the service name given the port number, and %SS_GETSERVBYNAME returns the port number given the service name. (8.1)
- (Windows) The profiler can now process a pre-7.5.1 **profile.dat** file. Additionally, the profiler program can now process a UNIX profile on Windows. (8.1)
- (Windows, UNIX) The PARSE subroutine now returns a remote host name in the node argument. The host name returned is in the form "@host". (8.1)
- (64-bit systems) On 64-bit systems (system codes 301, 021, 320, 304, and 309), a subscript can now be greater than 65534. (8.1)
- We added a new optional argument to %SYN_SYSERRTXT to allow the system error text to be provided from a passed error code as well as the current %SYSERR error code. (8.1)
- The SLEEP statement now processes implied variables such that millisecond precision can be specified on those systems that support it. (8.1)
- The performance of the assignment opcode (STO) and some XCALL/function calls has been restored to their pre-7.5.1 level. (8.1)
- We added a new parameter to NSPC_ADD to allow users to request an access code for the item being inserted. Passing this parameter is equivalent to calling NSPC_ADD without the parameter, followed by an NSPC_MOVE with the returned access code as the *old_entry* parameter and the requested access code as the *new_entry* parameter. (8.1)
- (Windows) If the SYNERGYDE environment variable does not include a trailing slash, the runtime now appends one. This ensures that 7.5.1 Synergy/DE clients can continue to run with version 8 shared installations without having to upgrade or uninstall/re-install. (8.1)

- The ^B/^O/^X data characteristics operations have been modified to produce an **i4** value if the result fits in 32 bits or an **i8** if not. Also, an optional second argument with valid values of 1, 2, 4, and 8 has been added to specify the size of the resulting integer. Additionally, a 32-bit value with the high bit set is returned in an **i8** with the upper 32 bits clear. (8.1)
- (Windows, UNIX) We added support for the following %ISINFO options: PAGESIZE, KEYDEPTH, CREATEDATE, CLEARDATE, REINDEXDATE, and VERIFYDATE. (8.1)
- (Windows) **Dbr** and **dbs** can now be used as a scheduled task to emulate a batch job. Scheduled tasks using **dbr** that are run while you are 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 run while no user is logged in will operate as if TNMBR is set to -1 (detached). In this mode there is no user-interface, and Toolkit user-interface calls should not be made. You can test for %TNMBR.lt.0 to detect this condition.

  If you use a scheduled task and want to review any error log output when the user is not logged in, redirect stdout to a file. For example, "dbr my_prog > my_out.log". (8.1)
- (Windows) A comment that begins with a semicolon (;) is now allowed in the [colors] section of **synergy.ini** or **synuser.ini**. (8.1)
- (Windows) The *parent_id* argument of the JBNO routine now returns the session ID when Windows Terminal Services is enabled and 0 when it isn't. The session ID is common between processes running from the same Terminal Services Client and unique between other clients. (8.1)
- (Windows, UNIX) When a tempfile creation failure occurs and auditing is enabled, a line with additional information is now output to the audit log. (8.1)

Synergy Configuration Program (synconfig)

- We now distribute a manifest file that enhances the user interface when running on XP. (8.3)
- We modified the SlickEdit Key dialog to allow you to paste the full key into the first edit control and thereby populate all of them, honoring any hyphens that occur in the pasted input. Also, when typing a key, filling the field to its maximum number of characters will advance to the next control. (8.3)
- We added the new security options for *xf*Server to the xfServer Information dialog. (8.3)
- We added the new *xf*Server verbose logging option to the xfServer Information dialog. (8.3)
- We removed the "xfServer data access" panel from the xfServerPlus Information dialog because *xf*Server data access on the *xf*ServerPlus port is no longer supported. (8.3)
- We added support for Oracle 10 and SQL Server 2000 (ODBC) to the Connectivity Series tab. (8.1.7)

- We added new “Installation type” and “Synergy/DE version” fields to the “Contact Information” and “Registration Information” dialogs. We also removed the “Fax number” field from the E-mail dialog, and the Company name in both dialogs no longer defaults to the Licensee name. (8.1.5)
- We added new options to the *xf*ServerPlus service dialog for remote debugging. See *Developing Distributed Synergy Applications* for details. (8.1.5)

Synergy DLL API

- (Windows, UNIX) As planned, the %DLL_SUBR function has been removed. Since version 8.1.5, we have recommended that all calls to %DLL_SUBR be replaced with calls to %DLL_CALL. To ease migration, however, %DLL_SUBR has been redefined as a macro for %DLL_CALL(DLL_TYPE_STDCALL). You must recompile your code. If your DLL does not conform to the __STDCALL or __WINAPI calling convention, this modification will break your code. (8.3)

Synergy HTTP document transport API

- When calling %HTTP_CLIENT_GET or %HTTP_CLIENT_POST, you can now turn off server verification by passing an HTTPS URI without specifying a CA_file. (Because this poses a security threat, we recommend that you not do it unless you only need encryption without verification.) (8.3.1d)
- We added an optional alpha argument to the %HTTP_CLIENT_POST, %HTTP_CLIENT_GET, and %HTTP_SERVER_SEND routines that allows you to specify the HTTP version number that will be put into the document header. (8.3.1d)
- The maximum length of a URI in the Synergy HTTP document transport API is now 2000 bytes. URIs longer than 2000 bytes will be truncated. The limit on individual HTTP headers is 499 bytes. Headers longer than 499 bytes will be truncated. Segmentation faults will no longer occur if the URI or HTTP header is too long. (8.3.1a)
- We added the ability to send HTTP requests using a relative URI for %HTTP_CLIENT_GET and %HTTP_CLIENT_POST. (8.3.1a)
- The Synergy HTTP document transport API (specifically %HTTP_CLIENT_POST, %HTTP_CLIENT_GET, and %HTTP_SERVER_RECEIVE) now supports user information in a URI. The URI can have the format

  `http://`*username*`:`*password*`@`*host*`:`*port*`/`*somefile*`.html`

  For security reasons, we do not recommend passing the *username* and *password* as clear text in an HTTP packet. (8.3)

- We added automatic escaping of certain characters in a URI by %HTTP_CLIENT_GET and %HTTP_CLIENT_POST, and automatic unescaping of any escape codes in a URI by %HTTP_SERVER_RECEIVE. (See the list of characters that get escaped in the %SYN_ESCAPE_HANDLE documentation in the "System Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual*.) If you had already implemented your own escape mechanisms prior to this feature being added, this modification may break your code. To revert to pre-v8.3 behavior and disable automatic escaping and unescaping, set the new HTTP_NOESCAPE environment variable to any value. (8.3)
- We added two functions that enable you to escape and unescape characters in a string within a handle: %SYN_ESCAPE_HANDLE and %SYN_UNESCAPE_HANDLE. They have the syntax

  sts = %SYN_ESCAPE_HANDLE(*handle, rules, length*)

  and

  sts = %SYN_UNESCAPE_HANDLE(*handle, rules, length*)

  where *handle* is the memory handle that contains the string, *rules* is either XML_RULES or HTTP_RULES, and *length* is the length of the string and, upon return, the length of the escaped or unescaped text. (8.3)
- (Windows) The Synergy HTTP document transport API now retries transient service failures. (8.1.7)
- In the Synergy HTTP API, logicals are now supported on all logfile, certfile, and cafile arguments. (8.1.3)
- You can now use an uppercase string (e.g., "LOCALHOST") in the URI argument of HTTP_CLIENT_GET or HTTP_CLIENT_POST. (8.1.3)
- Version 8 introduces support for HTTPS by providing an interface to OpenSSL. This is accomplished via extensions to the Synergy HTTP Document Transport API. Currently, this feature is supported on all Synergy platforms except NCR. Synergex does not distribute OpenSSL with Synergy/DE. It is your responsibility to get OpenSSL for your particular operating system. Red Hat and SCO Linux systems will require the openssl packages that come with operating system support. For OpenVMS, CompaqSSL v-1.0a requires version 7.2 of the operating system. For all other operating systems, download OpenSSL version 0.9.7 from **http://www.openssl.org/** and build the shared libraries. Note: If building on AIX, you will need to make build script modifications. Contact Synergy/DE Developer Support for assistance. Refer to the "Synergy HTTP Document Transport API" chapter of the *Synergy Language Reference Manual* for more information. (8.1)

Synergy Language windowing subroutines

- The W_INFO subfunctions now give a subscript error if there is insufficient space in the data division for the returned data. A common mistake is for WI_ACTIVE not to pass an array large enough for MAXWINS windows. (8.3.1d)
- (Windows) We have reduced the amount of resources required to create a Synergy window. (8.3.1a)
- (Windows) We improved Synergy/DE's color support to accommodate Windows system colors. You can now assign Windows system colors to the various UI elements in applications that use the Synergy Language windowing subroutines by assigning the system colors to Synergy color palette entries. To implement this, we made the following changes:
  - We increased the number of colors that Synergy/DE supports to 512 (color 0 through color 511), and we assigned Windows system colors to color 256 through color 285. (Colors 0 through 255 are user-defined colors, as they have been in previous versions, and colors 286 through 511 are reserved for future Windows system colors.
  - We removed Synergy color settings and the palette setting from the distributed **synergy.ini** file. In previous versions, the Synergy runtime defaults were used only if you removed these overrides or used a different **synergy.ini** file.
  - We changed the foreground and background colors for the first and third color palette entries in the default palette to bring default palette and renditions settings closer to standard Window settings. The foreground color is now 264, which is the Windows system color for text for window objects, and the background color is now 271, which is the Windows system color for the face of 3D objects.

  This modification may break your code. To accommodate the increased number of Synergy colors, the WI_PALET subfunction to W_INFO may now return values that are too large for **i1** variables (which were sufficient for all color values in previous versions). Check your code to make sure variables used to store these values are no smaller than **i2** or **d3**. (We recommend using **i4** variables.)

  Also note that the changes to the first and third entries of the default color palette and the absence of color and palette definitions in the distributed **synergy.ini** file may change the appearance of an application if you install Synergy/DE on a new system. (8.3)
- We added a new subfunction to W_PROC: WP_TOFRONT. This subfunction brings a window to the front of the display without changing its position. (8.1.7a)

Synergy socket API

- We added a new optional argument, SS_PURGE, to %SS_CLOSE to avoid the socket shutdown logic when using the FORK subroutine. The new syntax is

  status = %SS_CLOSE(*socket[,* SS_PURGE*]*)

  (8.3)

- (Windows) The socket API routines poll for data every second during RECV and ACCEPT operations and now use the message poll loop to ensure the application does not stop responding. (8.3)

Synergy Windows Printing API

- In addition to the existing Print Setup dialog, we have now made the standard Windows Print dialog available from the Windows printing API via a new optional argument to the DWP_PRINTDLG subfunction of %WPR_INFO. This gives users access to features such as Print to file, Page ranges, # of Copies, and Collation. (8.3)
- You can also access the Print dialog from the print previewer if you have enabled the new optional "Print..." menu entry. To print directly from the previewer without accessing the Print dialog, select File > Print now. (Note that "Print now" was previously called "Print", but we renamed it to avoid confusion with the new "Print..." entry.) (8.3)
- We changed the default behavior of the arrow keys in the print previewer to advance 5% of the visible area. You can override this percentage by setting the PRINT_PREVIEW_SCROLL environment variable in the environment to a number between 1 and 100. (8.3)
- We added the ability to search for text in the print previewer. (8.3)
- We added the DWP_SETPAGES option to %WPR_PRINT, which gives you the ability to programmatically specify a range of pages to preview and print. (8.3)
- You can now save the metafile created by the Windows printing API without previewing or executing the print job first. You can do this by passing the optional *keep_file* argument to the DWP_DELPRINTER subfunction of %WPR_INFO. (8.3)
- Two example programs that showcase the new Windows printing API features are being distributed in the dbl\examples directory. They are called **tprt.dbl** and **tprt2.dbl**. (8.3)
- A clearer informational error message is now generated when DWP_GETPRINTER fails, indicating the printer device name in error. (8.1.7a)
- We added new image support to the Synergy Windows printing API. JPEG, GIF, Exif, TIFF, and PNG formats are now supported in addition to BMP. (8.1.7)

Synergy XML API

- The performance of the XML parser for large documents has been significantly improved. The XML API can now load very large documents 100 times faster than before. (8.3.1d)
- Memory is now less fragmented as a result of using the XML API. In addition, an XML string no longer requires a minimum of 4096 bytes. (8.3.1d)
- We can now handle larger XML documents by reducing memory fragmentation. A "Not enough memory for desired operation" error is no longer generated when a large number of elements are added to an XML document. (8.3)
- The Synergy XML API can now parse, manipulate, and serialize multiple XML declarations in an XML document. This change affects %XML_PARSER_PARSEFILE, %XML_PARSER_PARSESTRING, %XML_DOC_TOFILE, %XML_DOC_TOSTRING, and %XML_DOC_GETDECLARATION. %XML_DOC_GETDECLARATION has a new, optional parameter. Its syntax is now

  sts = %XML_DOC_GETDECLARATION(*doc, declar[, declar_name]*)

  %XML_DOC_SETDECLARATION has been superseded by a new routine, %XML_DOC_ADDDECLARATION, which adds a declaration to an XML document. It has the syntax

  sts = %XML_DOC_ADDDECLARATION(*doc, declar_name, declar*)

  We also added %XML_DOC_REMOVEDECLARATION, which removes the specified declaration from an XML document and has the syntax

  sts = %XML_DOC_REMOVEDECLARATION(*doc, declar_name*)

  (8.3)

- We added several new routines to the Synergy XML API to improve manipulation of the DOM tree:
  - %XML_ATTR_GETVALUEHANDLE returns the handle that holds the value for the attribute. It has the syntax

    handle = %XML_ATTR_GETVALUEHANDLE(*attr*)

  - %XML_ATTR_SETVALUEHANDLE assigns a handle as the value for an attribute. It has the syntax

    sts = %XML_ATTR_SETVALUEHANDLE(*attr, handle*)

  - %XML_ATTRLIST_ADDLIST enables you to add each of the attributes in a list to the end of another attribute list. It has the syntax

    sts = %XML_ATTRLIST_ADDLIST(*destid, sourceid*)

- %XML_ATTRLIST_INSERT enables you to insert an attribute in an existing attribute list. It has the syntax

  sts = %XML_ATTRLIST_INSERT(*attr_list, attr, index*)

- %XML_ELEM_ADDATTRIBUTES enables you to add each of the attributes in a list of attributes to a given element. It has the syntax

  sts = %XML_ELEM_ADDATTRIBUTES(*element, attr_list*)

- %XML_ELEM_ADDCHILDREN enables you to add each of the elements in a list of child elements to a parent element. It has the syntax

  sts = %XML_ELEM_ADDCHILDREN(*parent, elem_list*)

- %XML_ELEM_GETTEXTHANDLE returns the handle that contains the text for an element. It has the syntax

  handle = %XML_ELEM_GETTEXTHANDLE(*element*)

- %XML_ELEM_REMOVEATTRIBUTES enables you to remove all attributes from a given element. It has the syntax

  sts = %XML_ELEM_REMOVEATTRIBUTES(*element*)

- %XML_ELEM_REMOVECHILDREN enables you to remove all child elements from a given element. It has the syntax

  sts = %XML_ELEM_REMOVECHILDREN(*element*)

- %XML_ELEM_SETTEXTHANDLE assigns a handle as the text for an element. It has the syntax

  sts = %XML_ELEM_SETTEXTHANDLE(*element, handle*)

- %XML_ELEMLIST_ADDLIST enables you to add each of the elements in a list to the end of another element list. It has the syntax

  sts = %XML_ELEMLIST_ADDLIST(*destid, sourceid*)

- %XML_ELEMLIST_INSERT enables you to insert an element in an existing element list. It has the syntax

  sts = %XML_ELEMLIST_INSERT(*elem_list, element, index*)

(8.3)

- We added escaping and unescaping of certain characters in XML data to the Synergy XML API. Use the new %XML_OPTION function to turn escaping and unescaping on or off:

  sts = %XML_OPTION(ENCODE, *value*)

  where *value* is one of the following:

SYNESCAPE_OFF (default)	Don't escape or unescape any items.
SYNESCAPE_ESCAPE	Escape text and attribute values according to XML escape rules for %XML_ATTR_SETVALUE, %XML_ELEM_SETTEXT, %XML_ELEM_SETATTRIBUTE, %XML_PARSER_PARSEFILE, and %XML_PARSER_PARSESTRING.
SYNESCAPE_UNESCAPE	Unescape text and attribute values according to XML escape rules for %XML_ELEM_GETTEXT and %XML_ATTR_GETVALUE.

  (See the list of characters that get escaped in the %SYN_ESCAPE_HANDLE documentation in the "System Supplied Subroutines, Functions, and Classes" chapter of the *Synergy Language Reference Manual*.)

  For performance reasons, %XML_ELEM_GETTEXTHANDLE, %XML_ELEM_SETTEXTHANDLE, %XML_ATTR_GETVALUEHANDLE, and %XML_ATTR_SETVALUEHANDLE do not honor %XML_OPTION with regard to escaping and unescaping text or attribute values. If you use these functions, escaping and unescaping certain characters (for example, ampersands and double quotation marks) is your responsibility. See %SYN_ESCAPE_HANDLE and %SYN_UNESCAPE_HANDLE for more information.

  IMPORTANT: If you call %XML_ELEM_GETTEXTHANDLE or %XML_ATTR_GETVALUEHANDLE and then do an escape or unescape on the handle, any document that contains that element or attribute will be modified. Therefore, if you don't want to modify your document, you should make a copy of the handle before performing any escaping or unescaping. (8.3)

- We added two functions that enable you to escape and unescape characters in a string within a handle: %SYN_ESCAPE_HANDLE and %SYN_UNESCAPE_HANDLE. They have the syntax

  sts = %SYN_ESCAPE_HANDLE(*handle, rules, length*)

  and

  sts = %SYN_UNESCAPE_HANDLE(*handle, rules, length*)

  where *handle* is the memory handle that contains the string, *rules* is either XML_RULES or HTTP_RULES, and *length* is the length of the string and, upon return, the length of the escaped or unescaped text. (8.3)

- The Synergy XML API no longer generates an error when it parses an XML declaration that it doesn't recognize. Instead, it just parses and ignores it. (8.1.7a)
- With %XML_ELEM_GETTEXT, if two pieces of text are separated by a *child*, we now insert a space between the concatenated text to be compatible with the implementation that would occur if the *child* didn't exist. (8.1.7a)
- We improved the performance of the Synergy XML API. (8.1.7)
- The Synergy XML API now supports embedding an XML document within another XML document using the CDATA syntax. (8.1.7)
- The Synergy XML API now allows input lines within an XML file to be up to 65,534 characters in length. (8.1.7)
- The Synergy XML API now supports text fields and attribute values up to 65,534 characters in length, and element and attribute names up to 256 characters in length. (8.1.7)
- The maximum token size for input from an XML file or schema is now 10,000. This is the same as the maximum element text size. (8.1.5d)
- There has been a slight reduction in memory required by the Synergy XML API as a result of the new use of stack records. (8.1.5)
- We now document an additional routine in the Synergy XML API: %XML_ELEM_TOSTRING. (8.1.3)
- We added the following routines to the Synergy XML API to support removal of elements and attributes from an element: XML_ELEM_REMOVECHILD, XML_ELEM_REMOVEATTRIBUTE, XML_ELEMLIST_REMOVE, and XML_ATTRLIST_REMOVE. (8.1.3)
- We added the Synergy XML API, which gives you direct access to XML from your Synergy programs. It enables you to
  - Parse an XML file or string into a memory-based XML document that represents a DOM tree.
  - Iterate through the DOM tree to access the XML data.
  - Assemble Synergy data into a memory-based XML document.
  - Write the contents of a memory-based XML document to a file or XML string.

  Refer to the "Synergy XML API" chapter of the *Synergy Language Reference Manual* for more information. (8.1)

Utilities

- It is now possible to profile a program that is chained to. We recommend that you edit the resulting **profile.dat** file and remove the "chained from" program data before running **profile.dbr**. (8.3.1d)
- (UNIX) We have improved error messages for the backup mode. This affects **synbackup**, **irecovr**, **isutl**, and **fconvert**. (8.3.1d)
- (Windows, UNIX) We added the **synbackup** utility, which provides a way for all cooperating processes to freeze update I/O while Synergy databases are being backed up. (8.3)
- The Synergy Language profiler now supports a routine exclusion list so that specific routines can be excluded from the resultant profile trace. See "The Synergy Language Profiler" in the "General Utilities" chapter of *Synergy Language Tools* for more information. By default, the profiler now uses an exclusion list for the UI Toolkit input routines (**WND:tkexclude.txt**). (8.3)
- The **synckini** utility now gives you the option to create the **synuser.ini** file if it doesn't exist. (8.3)

*xf*NetLink Synergy

- We changed the way that *xf*NetLink and *xf*ServerPlus use sockets to establish communication. Version 8.3 *xf*NetLink clients will connect to *xf*ServerPlus using the new method; pre-8.3 clients and clients running in debug mode will continue to use the old method. This change was made to prevent problems that occurred when there were multiple firewalls in a WAN. (8.3)
- You will no longer receive an error when calling a Synergy method that has a collection parameter from an *xf*NetLink Synergy client. This enables you to use the same method definitions for *xf*NetLink Synergy and *xf*NetLink Java or .NET. On the *xf*NetLink Synergy side, write your code as though you were passing variable-length data. That is, create a memory area on the client and pass the memory handle (as the parameter that is defined as a collection in the MDU) to *xf*ServerPlus using the RCB_*xxx* routines. For more information, see "Returning a Collection of Structures" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual. (8.1.7a)

Version 7

This section briefly describes new features in Synergy Language version 7 and the changes in this version that may break your code. For information on the changes you may need to make to your system and code when you upgrade, see the *Synergy/DE Quick Migration Guide*.

Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Synergy Language version 7.

- In versions 7.5.1 and 7.5.1a, the ^SIZE and %LEN operators generated incorrect object code when used on a subscripted or ranged variable. We have corrected this. This modification requires a recompile of code compiled with 7.5.1 or 7.5.1a. (7.5.1b)
- (Windows) If the printer device is incorrect when using the Windows printing API or the LPQUE statement, $ERR_NOTAVL instead of $ERR_FNF is now reported. This modification could break your code depending on how your error trapping is coded. (7.5)
- (Windows, UNIX) An attempt to open an ISAM file explicitly with no extension (e.g., “filename.”) no longer results in a segmentation violation (v7.3.3). In Synergy/DE version 7.5.1 and higher, an attempt to create, open, or otherwise access an ISAM file using the “filename.” form results in an “Invalid file specification” error (E_FILSPC). (7.5)
- (Windows) The Windows printing API now uses **.emf** as the meta file extension, using it to both create and re-open a meta file when no extension is specified. Previously, **.spf**, **.ddf**, or nothing was used as the default extension when none was explicitly specified. (7.5)
- (Windows, UNIX) The Synergy sort now requires the input file, output file, and temporary file to reside on the same machine. Formerly, due to the way client/server sorting was initially implemented (all file I/O went across the network), these files could reside on different machines. The new Synergy sort for client/server handles all I/O on the target machine. All file specifications *must* either include the same host name or omit the host name. (7.3)
- The %ATRIM and %TRIM functions now return an error if the variable passed is numeric. This will break your code if you are passing numeric variables to %ATRIM and %TRIM. (7.1)
- (Windows) We added an optional argument to AX_INPUT. The *wait_time* argument sets the timeout value for an ActiveX control. Because the default behavior is to use the global timeout value, specifying *wait_time* may change existing behavior. Previously, no timeout occurred in AX_INPUT. Thus, if any existing program depends on AX_INPUT never timing out, that program needs to specify D_FOREVER for *wait_time*. (7.1)

New features

ActiveX (Windows)

- We modified the user interface of the ActiveX Diagnostic Utility. (7.1.5)
- We added the following ActiveX API functionality:
  - Support for indexed and design-time properties, as well as the ability to put a caption on a container (with the CAPTION control property). (7.1)
  - A new ActiveX debugging option that logs events to a file instead of a window. To activate this option, set AXDEBUG=FILE=*filename*. (7.1)
  - The AX_TIMEOUT subroutine, which sets a time-out value for an ActiveX control. (7.1)
  - A new control property, ICON, which sets the container's icon. We also added a new read-write ambient property, AMB_USERMODE, which sets the user or design mode state. (7.1)

Compiler/Linker/Librarian

- The Synergy compiler now allows real array specifications in replacement text for macro arguments. (7.5.1f)
- The Synergy compiler now supports the following C-style comparative operators: ==, !=, >, >=, <, <=, !, ||, &&, ~, |, &, |=, and &=. (7.5)
- Macros are now allowed to be used as parameters to other macros, including themselves. (7.5)
- We added two new compiler options that report unused variables.
  - **-qvar_review** (or **/VAR_REVIEW** on OpenVMS) generates a file that reports unused variables.
  - **-qreview_level** (or **/REVIEW_LEVEL** on OpenVMS) specifies the level of variable usage reporting. (Level 0 is unreferenced include files and variables only, level 1 is level 0 plus unreferenced global variables and labels, and level 2 is level 1 plus process local routines.)

  (7.5)
- The compiler no longer requires empty parentheses when no parameters are specified in a .DEFINE parameterized macro reference. (7.5)
- The compiler listing file has been changed to display the source file number by default instead of the listing level number. A new compiler switch has been added to get the alternate behavior. On Windows and UNIX, the switch is **-qlist_level**. On OpenVMS, the switch is **/INCLUDE_LEVEL**. (7.5)
- The compiler now allows ",GROUP=*name*" syntax when including from a repository. (7.3)

Debugger

- The debugger now supports the following operators within the "set watch" syntax: ==, !=, >, >=, <, and <=. (7.5)
- We added a new feature to the debugger: SAVE *filename*. This allows the current breakpoints, watchpoints, and other settings to be written to the specified filename. (7.3)

Environment variables

- (UNIX) We added the JBWAIT environment variable to control the amount of time to wait for a failed job issued by the RUNJB subroutine to start. (7.1)

Error trapping

- Normally, a routine uses an ONERROR statement to trap errors. If no ONERROR statement (or error list for an individual statement) exists, a fatal traceback is generated by the runtime. This is an increasing problem for products such as *xf*ServerPlus, which need to be able to control all exceptions to provide information to the caller. To facilitate this control, Synergy/DE has added catch functionality to the runtime via the new $ERR_CATCH error literal. Error catching enables a routine to set an ONERROR trap that will be called whenever an untrapped error occurs in a routine lower in the call tree. (7.3)

Installation

- (Windows) The Synergy/DE and Synergy/DE Client installations now check for and install MDAC version 2.7 if necessary. (7.5.1c)
- (Windows) The "Administrative Image/Run from Network" installation feature of version 7.3 has been superseded by the new "Synergy/DE Client" installation program that can be installed as part of a shared Synergy/DE installation. (7.5)
- (Windows) Version 7.5 installations will test for and, if needed, install Windows Installer Service version 2.0. (7.5)
- A new file, RESTRICT.TXT, is now distributed with Synergy/DE. This file contains all the documented product restrictions previously located in the individual product release note files. This file is distributed in the dbl directory. (7.5)
- (Windows) The version number has been removed from the Synergy/DE Start menu entry in order to more cleanly support "installation upgrades" in future versions. (7.5)
- (Windows) Support has been added for wrapping the Synergy/DE version 7.3.3 and higher installation within the installation of your own application. You can create a silent installation, which requires no user input via dialog boxes, or a full user interface installation, in which customers will see the same dialogs that you see when installing Synergy/DE. For more information on using this feature, call your Synergy/DE account manager at 800-366-3472 (in North America) or 916-635-7300. (7.3.3)
- (Windows) The licensing of the Visual SlickEdit component of Professional Series Workbench has been moved to the Synergy Configuration program. (7.3.3)

- (Windows) The Connectivity Series Type prompt has been removed and SQL OpenNet server is no longer registered during installation. To register, start, and stop SQL OpenNet server, start the Synergy Configuration program (from the Start menu, select Programs > SynergyDE > Utilities), and go to the Connectivity Series tab. For more information, refer to the Synergy Configuration Program online help. (7.3.3)
- (Windows) A shortcut to the UI Toolkit's script utility has been added to the Start menu. (7.3.3)
- (Windows) Connectivity Series can now be installed in the "Run from Network" configuration. This allows SQL Connection or ODBC client access, SQL stand-alone access, or ODBC stand-alone access. (7.3.3)
- (Windows) We added the SYNERGYDE environment variable, which specifies the Synergy/DE directory. (7.3)
- (Windows) The distributed **synergy.ini** file now sets the DEFAULT font to a fixed-space, sans serif font. (7.3)

ISAM

- (UNIX) We now support terabyte ISAM files on Red Hat Linux. (7.5.1f)
- (UNIX) File sharing has been enhanced for Sun Solaris (32-bit & 64-bit) to take advantage of a Sun-specific file sharing facility. This will enhance performance under extreme user load (4000+), allowing conventional file locking to be turned off via DBLOPT 33 and at the same time retaining file protection by denying access to files for "opened exclusive" access. DBLOPT 33 must not be set if a prior Synergy version or non-Sun Synergy remote access is combined. (7.5.1f)
- (Windows, UNIX) A new environment variable, SDMS_AUDIT_ROUTINE, has been added to print routine names for the sdms audit log. (7.5.1f)
- (Windows, UNIX) A new **fconvert -8** option can be used to suppress the binary data warning when 8-bit ASCII characters are expected. This option must be specified for each input file where suppression is desired. Use of this option cancels detection of 8-bit binary data and possible premature record termination. When the **-8** option is not used, the offset of the first binary character detected is now displayed as part of the warning message. (7.5.1f)
- (Windows, UNIX) The **isutl** utility now detects bad RFA vectors when verifying. (7.5.1e)
- (Windows, UNIX) A small reduction in system call overhead has been made for every keyed read to an ISAM file that does not perform a wait greater than 1 second or a wait forever. (7.5.1e)
- (Windows, UNIX) A small performance increase has been obtained in **isutl** and **fconvert** when loading files. (7.5.1e)
- (Windows, UNIX) The performance of loading an empty ISAM file using **fconvert** or **isutl -f** when DBLOPT 36 is set has been improved. (7.5.1e)

- (Windows, UNIX) The **fconvert** utility now optimizes the load of a new or empty ISAM file when specifying an ISAM file as input. All input file types now perform optimized load logic. (7.5.1c)
- (Windows, UNIX) The **isutl -l** option now verifies a read-only file. (7.5.1c)
- (Windows, UNIX) A new command line option, **-p**, has been added to **isutl** that allows a Rev 2 or 3 ISAM file to be quickly updated to Rev 5. Rev 5 files reap the same performance benefits as Rev 4, but are recognized only by Synergy/DE version 7.5 or later. If you need your Rev 2 or Rev 3 ISAM files to remain compatible with version 7.1 or 7.3, you should convert them to Rev 4 using **irecovr** or **fconvert**. Do not patch them to Rev 5. (7.5)

  Although we introduced Rev 5, the default revision on file creation is still 4. This means that only files explicitly patched (**-p**) or created with ISAMC_REV=5 will be created as Rev 5. (If necessary, a Rev 5 file can be patched back to Rev 2 or 3 with no adverse effects.) (7.5)
- (Windows, UNIX) The **isutl** utility has been enhanced to log all operations to a log file. Logging can be disabled using an environment variable. (7.5)
- (Windows, UNIX) The **isutl** utility has been modified to allow verification of a file without requiring exclusive access. The new option is **-l**. (7.5)
- (Windows, UNIX) We added a fast-load option (**-f**) to **isutl**, which loads a sequential file into an empty ISAM file very quickly. It can potentially reduce the time required to load an ISAM file from hours to minutes. (7.5)
- (Windows, UNIX) The **fconvert** utility has been optimized for speed. (7.5)
- (Windows) If an interactive run of **isload** generates an error, the error now stays displayed until a keyboard entry is made. (7.5)
- (Windows, UNIX) We enhanced the index checking done by **ismvfy** and **isutl -v** to better handle files with excessive index corruption. (7.5)
- We made the following improvements to the error/warning generation for **fcompare**.

  The **fcompare** utility now generates

  - a warning during data validation if a field defined as decimal contains a space.
  - an error if you specify the conversion setup file option (**-c**) in conjunction with the system catalog option (**-g**).
  - INFO messages only if the new **-i** command line option is specified.

  (7.3.3)
- The **fcompare** utility now supports version 6 repositories. (7.3.3)
- The **fcompare** utility now supports conversion setup files specified by the SODBC_CNVFIL environment variable. (7.3.3)
- The **fcompare** utility now allows you to specify a conversion setup file with the **-c** command-line option. (7.3.3)
- (OpenVMS) The **-k** option to **bldism** is now supported on OpenVMS. (7.3)

- Key names that are in a par file which is used as input to **fconvert** can now contain spaces. (7.3)
- If the largest key in an ISAM file is greater than 60, the file is now created with a default page size of 2,048. You can override this default by explicitly specifying a different page size. (7.3)
- We added the Synergy File Compare Utility (**fcompare**), which can perform two basic functions: compare an ISAM*xf* file definition to repository or system catalog metadata, and compare ISAM*xf* file data to repository or system catalog metadata. (7.3)
- We added the ISAM File Maintenance Utility (**isutl**), which performs all of the functions performed by **ismvfy** and **irecovr**, but faster and more efficiently. In addition to better utility performance, **isutl** optimizes and packs your index and orders your data file by preferred key, which can increase READ/READS performance. Note, however, that this utility does not currently support remote files. (7.3)
- We added a new **-a** option to the **irecovr** utility. This option applies necessary changes or corrections to the original data that is suspected of being corrupted. Use this option with caution: improper use of this option can result in loss of data. (7.1)
- (Windows, UNIX) We changed the default value for the ISAMC_REV environment variable. The default is now 4, which causes Synergy Language version 7 ISAM files to be created. (7.1)
- We added support for the following ISAM features:
  - Up to 255 keys, each of up to 8 segments
  - Ascending or descending key attributes per segment
  - Additional key types, including decimal, integer, unsigned, and nocase
  - Portable integer data
  - Specification of key and file density
  - File page size control
  - Specification of a null value on numeric as well as alpha keys
  - Terabyte files on Windows NT/2000 systems with the NTFS file system and various UNIX machines

  These enhancements involve changes to the syntax of the ISAMC subroutine and the **bldism** utility. (7.1)
- We added the %ISINFO routine to return ISAM file status and key information. (7.1)
- We added an ISAM definition language (XDL) that enables you to create keyword files to supply to **bldism**. (7.1)

Runtime

- In the HTTP document transport API, the HTTP_SERVER_SEND routine now handles documents over 64K. (7.5.1e)
- (Windows, UNIX) We added a new LOCK:Q_NO_TLOCK option for READS on a file open for update to perform like READS on a file open for input. A READS on file open for update by default probes the index tree and locks the record. With Q_NO_LOCK, it probes the tree but doesn't lock the record. Now, with Q_NO_TLOCK, it does NOT probe the tree or lock the record, similar to open for input. (See the /sequential open qualifier documentation for similar behavior.) (7.5.1e)
- (Windows, UNIX) We improved performance when loading routines. (7.5.1e)
- (Windows, UNIX) The PAK subroutine now sets the sign in packed data in the same manner as %PACKED and the packed data type. (7.5.1e)
- (Windows NT/2000/XP) The profiler now shows more accurate profile results using the elapsed time of the high granularity system clock. To re-enable the 7.5.1 behavior of processor time, which is only updated every 20 milliseconds, use the environment variable PROFILE_PROCESSOR_TIME. If PROFILE_PROCESSOR_TIME is used on Windows 98/Me, it is ignored. (7.5.1e)
- (Windows) The profiler now approximates the count in milliseconds when PROFILE_PROCESSOR_TIME is not set so the results are similar for both profiler types, given a compute program. (7.5.1e)
- In the HTTP document transport API, "Content-Type: text/xml" will now be sent as the default content type for all HTTP transactions. (7.5.1c)
- (Windows, UNIX) If an OpenVMS FDL is used for **fconvert** or the OPEN statement O:I with the FDL qualifier, the resulting ISAM file now has static_rfa set, compatible with OpenVMS. (7.5.1c)
- (Windows) In the Windows printing API, if SFWINIPATH points to an invalid location, the print preview settings are now saved to the default **synergy.ini** file location. (7.5.1c)
- (UNIX) When using a terminal other than a vt100-series terminal (TERM=vt100/vt220/etc or TERM=ansi) that recognizes ANSI color sequences to control color attributes, you can now set ANSICOLOR=1 in the environment and the runtime will use its pre-defined sequences for generating color. This eliminates your need to define these sequences in the termcap file and use the **dbr.tc** form of the Synergy runtime. (7.5.1a)
- The C interface has been modified to change all function and XCALL return values from **int** to INT_REG. C interface users on Digital UNIX should re-build their C functions and their **rxrefs.c** file. (7.5)

- The HTTP document transport API defines four new environment variables: PROXY_HOST, PROXY_LOCAL, PROXY_PORT, and PROXY_SUBNET. (7.5)
- (Windows) The Windows printing API now supports printer names up to 256 characters in length. If you use %WPR_GETDEVICE to obtain the printer name using the "printer" field of the "print_specs" structure, we recommend that you now use the "extended_pr_name" field instead. (7.5)
- (OpenVMS) The default temp file location for a SORT statement is now SYS$SCRATCH, if the logical exists. (7.5)
- (Windows) The service runtime (**dbs**) now displays version information if the ENTER key is pressed at the "dbs>" prompt. (7.5)
- (Windows) The default MAXMEM value is now 8mb. (7.5)
- (Windows) COM ports above COM4 are now supported. LPT ports above LPT4 are now supported. The NUL: device is now supported as an output-only device. Devices can be directly opened using \\.\com1 as well as COM1:. (7.5)
- (Windows) On Windows NT/2000/XP, a **.dbr** or **.elb** file in use by a program can now be renamed (*not* deleted). This helps users replace files that are in use when sharing **.dbr** or **.elb** files without requiring all users to log off first. These files cannot be renamed if they have been opened by a Windows 98/Me user. (7.5)
- (Windows) The terminal number is now guaranteed to be -1 if the program is not attached to the desktop (for example, a service or a scheduled task). (7.5)
- In versions prior to 7.5, the returned system code value on XP would have been 15 (the Windows 2000 system code). It is now 18 (Windows XP). (7.5)
- We added three new routines:
  - %SYN_FREECHN returns the next available channel.
  - %SYN_GETDIR returns the current directory path.
  - %SYN_SYSERRTXT returns a formatted error message from the last system error.

  (7.5)
- (Windows, UNIX) We added %SYN_XNAME. This routine returns a function name for a segment address. (7.5)
- (Windows, OpenVMS) We added a new routine, %SYN_SETDIR. This routine sets the current directory path. (7.5)

- The new HTTP document transport API enables you to send and receive documents via HTTP from within your Synergy program. Using this API, a Synergy program, posing as an HTTP client, is able to send or receive XML data from an HTTP server. Conversely, a Synergy program can pose as an HTTP server to receive XML data from an HTTP client and then send XML data to that HTTP client to satisfy the client's request.

  The HTTP document transport API consists of the following routines:

  - %HTTP_CLIENT_GET (sends a request to retrieve a document to an HTTP server)
  - %HTTP_CLIENT_POST (sends a request to post a document to an HTTP server)
  - %HTTP_SERVER_CREATE (creates an HTTP server class object)
  - %HTTP_SERVER_DELETE (frees memory for the HTTP server)
  - %HTTP_SERVER_RECEIVE (receives a request from a client)
  - %HTTP_SERVER_SEND (sends a response to a client)

  (7.5)
- (Windows) A client can now retrieve the last RFA or KRF used on the channel in a client/server configuration, which allows the file to be reopened and repositioned on server connection failure. (7.5)
- The FATAL subroutine now supports logical names. Additionally, if this subroutine is called and the fatal program does not exist, the error message from the runtime is now correctly displayed instead of a file not found message. (7.5)
- (Windows) The service runtime, **dbs.exe**, now interprets **synergy.ini** if SFWINIPATH is used. (7.5)
- (Windows, UNIX) A DISPLAY statement to a file now works when using the service runtime, **dbs.exe**. (7.5)
- (Windows) The function %SS_GETHOSTNAME now sets %SYSERR so you can detect the error code if the routine returns SS_ENULL. Additionally, %SS_SELECT now correctly sets the error code %SYSERR if the routine returns SS_ENULL. (7.5)
- (Windows) If the printer device is incorrect when using the Windows printing API or the LPQUE statement, $ERR_NOTAVL (instead of $ERR_FNF) is now reported. Also, if $ERR_NOTAVL is reported, %SYSERR now indicates the Windows system error code on Windows NT/2000/XP systems instead of reporting ERROR_BAD_DEVICE. Additionally, if an LPQUE error occurs when using *xf*Server, $ERR_LPQERR is now returned. (7.5)
- (Windows) We added support for new return values from the ENVRN subroutine for Windows XP and Windows XP 64-bit. The **dbl.def** file contains .defines for these systems as well. (7.5)
- (OpenVMS) When doing a READ or READS with LOCK:0 on a file open in update mode, the number of SPINLOCKs has been reduced if the operating system is OpenVMS Alpha Version 7.2-1H1 or later. (7.5)
- (Windows) The terminal bell is now sounded on a terminal services session. (7.3.3)

- (Windows) The default temp file (**DL*xxxxx***) is now placed in TEMP: if the TEMP environment variable has been set. (7.3.3)
- The *case* argument to the LOCALIZE subroutine now affects the S_ routines (S_BLD, etc.) as well as the UPCASE and LOCASE statements. (7.3.3)
- (Windows, UNIX) The SERIAL subroutine now returns the full 12-byte registration string if an **a12** argument is used. **a10** and smaller arguments return the same string they've previously returned. (7.3)
- (OpenVMS) The following syntax is now supported when an FDL file contains the ISAM filename specification.

  `OPEN(ch, O:I, "", FDL:`*fdl_file*`)`

  (7.3)
- (OpenVMS) An OPEN statement with POSITION:Q_EOF is now recognized on OpenVMS. All other POSITION qualifiers for the OPEN statement are still ignored. (7.3)
- (OpenVMS) The RECSIZ:-1 or RECSIZ:0 qualifier for an OPEN statement is now ignored instead of generating an IRCSIZ error. (7.3)
- (Windows, UNIX) xcall GETFA(ch, "RSZ", siz) is now supported and returns the record size of a file opened on the specified channel. For non-ISAM files, see the RECSIZ OPEN qualifier for information on how RECSIZ will be interpreted. (7.3)
- Search lists are now allowed as logicals in the program file name field for the RUNJB subroutine. (7.3)
- (Windows) We added the weight definitions for DWP_FONT as defined in **wingdi.h**. (7.3)
- (Windows, UNIX) We added the %DLL_CALL routine which calls a DLL subroutine. This routine offers a choice of calling conventions and supersedes the %DLL_SUBR routine. (7.3)
- We added a new %ERR_TRACEBACK function, which generates error traceback in conjunction with %ERR_CATCH. (7.3)
- We added the subfunction DWP_PAPERSOURCE to the %WPR_SETDEVICE and %WPR_GETDEVICE Windows printing functions, to set and retrieve the paper source for a printer. (7.3)
- We added two new subfunctions to WP_OPTION, a subfunction of W_PROC: WPO_KEEPBRDR, which prevents window borders from being turned off due to the size of the window, and WPO_HIDEBRDR, which restores the default behavior. A new environment variable (or **synergy.ini** setting) KEEP_BORDER=1 causes the WPO_KEEPBRDR setting to be used initially. (7.3)
- The key specification in the ISAMC subroutine now has an alternate form. You can specify either a variable that contains a single key specification (with one *key_spec* for each key in the ISAM file, as in previous versions) or a dimensioned argument (whose number of elements is equal to the number of keys) that defines each key being created. (7.1)

- We added a routine call block API that enables you to create and use Synergy routine call blocks (RCBs). This API includes the following routines: %RCB_CALL, %RCB_CREATE, RCB_DELETE, RCB_INSARG, RCB_SETARG, RCB_SETARGS, and RCB_SETFNC. (7.1)
- We added a new attribute and a new skip attribute for the windowing subroutines: ATTR_ITAL and SKIP_ITAL. These attributes are synonymous with ATTR_BLNK and SKIP_BLNK, respectively. (7.1)
- We added two new qualifiers to W_BRDR: WB_SBON and WB_SBOFF, which enable and disable scroll bars, respectively. (7.1)
- We added two new subfunctions to %W_INFO: WIF_NORMALX and WIF_NORMALY. These subfunctions return the normal position of the top left corner of a window, which is the restored position for the window. (7.1)
- The %JBNO routine now returns the first Ethernet adapter address in the adapter address argument if multiple adapters are present. (7.1)
- We added a new flag, D_GAINFOCUS, to the SHELL and SPAWN routines. This flag ensures that focus for the Synergy application is restored when SHELL or SPAWN finishes. (7.1)

Synergy Configuration Program

- (Windows) The new *xf*ODBC, SQL Connection, WebBuilder, and Backup License Server product codes have been added to the drop-down selection list of the Product Code field in the Install Keys Manually dialog. (7.5)
- The Enter Visual SlickEdit Key dialog has been modified to support version 6 SlickEdit keys. (7.5)
- (Windows) The License Information panel of the Licensing tab is now read-only. To modify your current license configuration, select the Advanced button. The Advanced License Manager dialog supports the reconfiguration of a license, the specification of a backup server, and the specification of license manager logging. (7.5)
- (Windows) The xfServer data access panel of the xfServerPlus Information dialog has been modified to include a third button, “Disable xfServer data access.” See REL_XFPL.TXT for more information on this new feature. (7.5)
- (Windows) The xfServer Information dialog has been modified to include a new “Network Optimization” panel that includes a checkbox labeled “Compress data packets.” Selecting this check box sets the SCSCOMPR environment setting in the registry. (7.5)
- (Windows) The Connectivity Series tab has been modified to include SQL Server 2000 as a radio button. (7.5)
- (Windows) The new *xf*ODBC and SQL Connection product codes have been added to the drop-down selection list of the Product Code field in the Install Keys Manually dialog. (7.5)

- (Windows) The licensing of the Visual SlickEdit component of Professional Series Workbench has been moved to the Synergy Configuration program. To configure the Visual SlickEdit license, select the "Enter Visual SlickEdit Key" button from the Licensing tab. After the key has been entered, Workbench appears briefly as it completes its installation. (7.3.3)
- (Windows) The starting, stopping, and registration of the SQL OpenNet server has been moved from the installation to the Synergy Configuration program. (7.3.3)

Utilities

- (Windows, UNIX) The **chklock** utility now supports terabyte files. (7.5)
- (Windows) The **setruser** utility now supports a **-d** command line option that deletes the RUSER value from **HKEY_CURRENT_USER\Software\Synergex** in the registry. (7.5)
- (Windows) We have made the screen output from **syncksys** scrollable. (7.5)
- (Windows) We added a new Synergy Configuration Program to the Utilities subfolder of the Synergy/DE menu. This program configures License Manager, SQL Connection, *xf*Server, and *xf*ServerPlus. (7.3)

Miscellaneous

- (UNIX) We changed the WY-60 (wyse60) terminal entry in **termcap.src** so that the default attribute is the dim or half-bright mode, and we added the bold attribute as the default brightness mode. Also, we removed the alt character set/reset sequence from the clear attribute. This eliminates some line drawing problems. (7.3)

Version 6

This section briefly describes new features in Synergy Language version 6 and the changes in this version that may break your code. For information on the changes you may need to make to your system and code when you upgrade, see the *Synergy/DE Quick Migration Guide*.

Features that may break your existing code

The following changes may break code when you update to Synergy Language version 6.

- Specifying a null or blank filename in the DELET subroutine now generates a "Bad file name" error ($ERR_FILSPC).
- (UNIX) If you use the signal subroutine in your C code that is linked to the runtime, you must either call d_signal (a runtime routine) as documented in **xcall.h** or use sigaction (a C routine). Use of signal can cause random and catastrophic errors at runtime depending on which UNIX system you are using.
- (UNIX) There are new logging options in **rsynd**.
- (Windows) We no longer save the colors to **synergy.ini** on program shutdown, due to the program changing the palette in a way that could be inappropriate for other applications. Currently, the only way to update the color settings in **synergy.ini** is by using Composer or manually editing **synergy.ini**.
- (Windows) The executables **dbl**, **dblink**, and **dblibr** are now console applications rather than Windows applications. If you have been using START with the /w[ait] switch in your Windows 95 batch files for compiling and linking, you no longer need to do so. If you continue to use the START /w switch, an MS-DOS box that you will have to close manually will appear.
- (Windows) We now provide limited support for License Manager server on Citrix. (Citrix remains a limited support platform because it is based on Windows NT 3.51.)
- (Windows, UNIX) We now generate an error on certain invalid RFAs on UNLOCK, READ and READS operations.
- (Windows, UNIX) In the W_BRDR subroutine, the title is no longer turned on automatically.
- (Windows, UNIX) We fixed a problem where READS of an ISAM record on a channel open for update did not always see newly created records. However, this can cause the performance of READS operations on files open for update to be slower than on previous releases. We added the new SEQUENTIAL OPEN qualifier to allow you to control this.
- (Windows, UNIX) We added a new program, **synxfpng**, to ping the Synergy/DE *xf*Server to aid in troubleshooting TCP/IP network problems.
- (OpenVMS) We removed from the Synergy runtime several routines that were previously used by the WAIT subroutine in Synergy Language 5.3.*n* and the pre-3.7 version UI Toolkit but are no longer supported. If you have not relinked your programs since version 5.3.12, you must do so to run with Synergy Language version 6.

New features

Compiler

We made the following changes to the Synergy compiler:

- The compiler switch **-qexpand** (or **/EXPAND** on OpenVMS) will cause lines that contain macro replacement to have their expanded form placed in the listing file following the regular listing line.
- We added support for parameterized macros to the .DEFINE compiler directive.
- The Synergy compiler now supports the inclusion of groups from the S/DE Repository.
- We added a new **-qimplicit_functions** (**/IMPLICIT** on OpenVMS) option to allow the Synergy compiler to assume %NAME is a %VAL function without requiring an external function definition.
- We added a new **-qtrim** (**/TRIM** on OpenVMS) Synergy compiler option to strip trailing arguments from XCALL routines.

Debugger

We made the following changes to the Synergy debugger:

- You can now set a watchpoint in the debugger to fire on a declared value and condition rather than just when the contents of a variable change.
- (Windows, UNIX) The SHOW MEMORY debugger command now displays the number of segment reclamations that have occurred during program execution.

Environment variables

We made the following changes to environment variables:

- We added the SYNCENTURY environment variable, which specifies the two-digit year that will be used to determine the default century. SYNCENTURY defines a “sliding window” for Synergy/DE default century methods by allowing certain years to use the current system century and other years to use the next or previous century.
- (UNIX) We added the environment variables RSYNDLOG and RSLOGMAX, which specify an alternate *xf*Server log file and a maximum log file size, respectively.
- (Windows) We added the ACTIVEX_LIST environment variable, which specifies whether to use the ActiveX list control or the UI Toolkit list by default when performing list processing.
- (Windows) We added the environment variable AXDEBUG to enable a debug display system when using the ActiveX API.
- (Windows, UNIX) We added the environment variable ISAMC_REV, which enables you to use the ISAMC subroutine to create files compatible with Synergy DBL 5.7.9.

Initialization settings on Windows

We made the following initialization setting changes:

- We added a new [fonts] section to the **synergy.ini** file for defining fonts. In this section, you can define a font for the debugger, as well as a global font, an alternate font, and any other font you desire.
- We added two new initialization settings, FONT_GLOBAL and FONT_ALTERNATE, to the **synergy.ini** file so you can specify the global and alternate fonts on a per application basis.

ISAM files

We made the following changes for ISAM files:

- (Windows, UNIX) We added Synergy ISAM index caching. This can improve performance on ISAM file access at the expense of memory. This is most noticeable on Windows systems.
- (Windows, UNIX) We now allow an ISAM file to be opened for input even if the user only has read-only access to both partitions of the file.
- (OpenVMS) We now support READ ^LAST on Synergy ISAM files. This feature is only supported with version 6.1 and higher of OpenVMS.
- (OpenVMS) We now support READS REVERSE on Synergy ISAM files. This feature is only supported with version 6.1 and higher of OpenVMS.

ISAM utilities

- The **irecovr** utility now puts appropriate NODUP and IRCSIZ errors into an exception file. We also added a new option (**-x**) that enables you to specify the name of the exception file that will be created if an error occurs.
- We improved the performance of the **irecovr** utility when loading files.

Runtime

We made the following changes to the runtime:

- We added the new **/sequential** and **/cache** OPTIONS qualifiers for the OPEN statement.
- (Windows) The Synergy runtime and *xf*Server now allow access and creation of files that contain spaces in the name. UNIX clients also allow spaces when accessing a Windows NT *xf*Server.
- (Windows) We now support the Q_EXCL_RO option, which allows read-only access to files.
- (Windows, UNIX) We added a new service runtime (**dbs**) that has no debugging and only limited terminal support. This runtime is much smaller and starts up faster. It is ideal for use in server applications and services where no terminal I/O is required.
- (OpenVMS) We added a new OPEN with pipe syntax.

```
OPEN (1, U, "|DCL command")
```

Subroutines and functions

We made the following changes to Synergy subroutines and functions:

- We added the %SS_FATAL function, which notifies the server that a program is terminating abnormally, to the Synergy socket API.
- The MODNAME subroutine now has an optional fourth argument for the source file number to go with the source file line.
- The ERROR subroutine and %ERNUM function now have an additional optional argument for the source file number.
- We removed documentation of the QUICKSOCKET subroutine. The subroutine itself will be removed in Synergy Language 7.1. We recommend you use the Synergy Language socket API instead.
- We added the %SS_SELECT function, which waits for data to arrive, to the Synergy Language socket API.
- We added a name space API that enables you to store sets of names and their associated data.
- We added a new set of subroutines to enable you to use TCP/IP socket programming from Synergy Language.
- (Windows) We added an ActiveX API that enables you to use ActiveX controls within Synergy Language.
- (Windows) We added a Windows printing API that provides the ability to perform Windows-style printing.
- (Windows) We added a third argument to the SHELL subroutine for customizing the process creation options on Windows.
- (Windows, UNIX) If a call to the WAIT subroutine for time and keyboard input is performed, and the wait time is less than 10 seconds, one-tenth-of-a-second sleeps are used instead of one-second sleeps. This improves response time for character input but may not work on some UNIX systems that do not have appropriate timers.

System options

- We added system option #54 to enhance the runtime operation when used with the **-qcheck** compiler option.

Windowing subroutines on Windows

- We added the ability to set the pixel position of the application window, as well as the state of the application window or a Synergy window.
- We added the ability to set the Windows cursor for a Synergy window, the application window's container, and the application space. We also added the ability to capture the mouse.

*xf*Server on Windows

- We now support multiple *xf*Server services.

Version 5.7

This section briefly describes new features in Synergy Language (Synergy DBL) version 5.7 and the changes in this version that may break your code.

If you are updating to Synergy DBL 5.7 on a Windows or UNIX system, you must recompile and relink all programs. If you are updating on an OpenVMS system, we recommend that you recompile and relink.

Features that may break your existing code

The following changes may break code when you update to Synergy DBL 5.7.

- The runtime now produces an $ERR_NOCURR error when a program tries to update a record that it has unlocked.
- %DATETIME now returns blanks for the microsecond digits that are beyond the accuracy of the operating system.
- There is a potential for data corruption due to a documentation error in W_INFO.
- The W_INIT subroutine now reports errors if the terminal channel specified is not valid.
- .START with no options no longer “turns on” listing.
- The compiler now reports errors on USING statements with invalid ranges.
- In certain cases, the runtime now reports $ERR_TOOBIG instead of $ERR_KEYNOT.
- The runtime no longer emits escape sequences for terminals it does not recognize as being able to handle VT escape sequences.
- The **bldism** utility no longer strips spaces from non-quoted key names.
- Using GETS on a serial device now correctly ignores the lower-case FLAGS setting.
- You cannot use **i8** variables as ^VAL function return values on non-64-bit machines.
- Special device names TI: and KB: have been removed from the runtime. Other DOS-specific special terminal/device names, CON:, AUX:, COM1:, LPT1:, PRN:, and LP:, have been conditionalized for DOS only. To make TI or KB or other DOS-specific special terminal/device names work like TT:, you will have to set up an environment variable (or logical for OpenVMS) that maps them to the terminal. Similarly, you will have to set up environment variables on non-DOS systems to map the DOS device/terminal names to the appropriate devices.
- Integers are now limited to sizes of 1, 2, 4, and 8 at run time and compile time (except in CLEAR, IF var, and IF .NOT. var statements).
- For performance reasons, Synergy DBL no longer supports using any intrinsic function that returns an alpha data type from non-DBL routines.

- The size of literal fields are now enforced. For example, in the following code **initial_value** is maintained as a **d3**:

  ```
  literal
     initial_value ,d3    ,2
  ```

  In previous versions, it was reduced to a **d1** because its value has a size of 1. (This change prevents errors when passing literals to routines expecting arguments of a certain size.)

  Note the following effect. If you specify

  ```
  record
     another_var   ,d2    ,initial_value
  ```

  the compiler will generate an "Initial value too long" error because the length of the initial value is now 3 instead of 1. Fix your code by changing the literal field to a **d1** or **d2**, or .DEFINE it.

- On a READ or a FIND, if you specify an invalid key number to the KEYNUM qualifier or you specify an implied key that is longer than any of the keys in the file, the runtime will now generate an "Illegal key list specified" error ($ERR_BADKEY). Previously, it would not generate an error and would default to using the primary key. If you set system option #45, an error will be generated if a key is specified without a key of reference, is within the record, and does not match or partially match a defined key. System option #35 implicitly sets option #45.
- If two **i4** variables are operated on and the result overflows an **i4**, an **i8** temporary result is not used on non-64-bit systems.
- Modules compiled with 5.7.0 or higher cannot be run with runtimes earlier than version 5.7.0 (because of changes we made to support multiple source line debugging and to improve performance of the USING and FOR statements).

New features

Compiler

We made the following changes to the Synergy compiler:

- (OpenVMS) If you install Synergy DBL to invoke the compiler with the same verb that invokes the Compaq DIBOL compiler, you should now specify /SYNERGEX instead of /DISC to specify the Synergy DBL compiler. Support for the /DISC qualifier will be phased out at some point in the future.
- We added support for the %DATECOMPILED and %ARGNUM compile-time functions.
- Synergy DBL now reduces ^D(literal) (as well as ^I and ^A) at compile-time, rather than calling a function at runtime.
- We added support for the ^DATATYPE(*arg_no*) data reference operation. It returns the data type of a variable or argument. The return values D_TYPE_I, D_TYPE_A, D_TYPE_D, D_TYPE_ID, D_TYPE_P, and D_TYPE_IP are defined in **dbl.def**.

- We added the /reentrant compiler option (**-E** on Windows and UNIX systems), which permits recursion by specifying that all routines in the files being compiled can be re-entered (as if you specified the REENTRANT qualifier on the SUBROUTINE and FUNCTION statements for these routines).
- We added the /profile compiler option (**-u** on Windows and UNIX systems), which enables profiling of the routines in the files being compiled if you set system option #40 or #41 when you run your program. See "Runtime" on page 1-86 for more information.
- We added the /decscope compiler option (**-s** on Windows and UNIX systems), which specifies that ^SIZE (%SIZE) of a real array or a pseudo-array will result in the size of the whole array. By default, Synergy DBL returns the size of one element. Additionally, ^SIZE(real_array) is now reduced at compile-time.
- We added identifiers to the **dbl.def** file for code portability:

```
.define D_ADDRSIZE            ;The size of a char * pointer
                              ; and return size from %xaddr.
.define D_NATINT ,I4          ;SIZE of native int in C.
.define D_NATLNG ,I8          ;SIZE of native long in C.
.define D_ADDRSIZ, 8          ;SIZE of char* pointer.
.define D_MAXINT, I8          ;Max supported int size.
.define D_AXP                 ;You are running on an ALPHA
                              ; processor.
.define DBLV57                ;You are running with 5.7 or
                              ; later of Synergy DBL.
.define D_GUI                 ;You are running under Synergy
                              ; for windows.
```

- You can now debug programs that .INCLUDE procedure division statements.
- The RANGE qualifier for a USING statement can now contain compile-time expressions such as %CHAR.
- The compiler generates faster code for integer FOR and USING statements. With this release, USING is now the fastest selection statement in the runtime and is as fast as an unlabeled CASE statement.
- If you specify XRETURN in a main routine, the compiler will now generate a "Not within function or subroutine" error (NOTINSUB).
- (Windows, UNIX) To match the /ALTSTORE option on OpenVMS, we added the **-v** compiler option, which supports VAX-DIBOL-compatible zoned stores.
- The .IF compiler directive indicates that the statements that follow are to be compiled if the specified compile-time expression results in a true value. For example:

```
.IF compile_time_expr
    statements
.ENDC
```

- You can specify a library location for the .INCLUDE directive with this syntax:

  ```
  .INCLUDE 'name' LIBRARY 'lib_name'
  ```

  where *lib_name* is a logical defined as a search list. If *lib_name* has an extension, it will be stripped. If *lib_name* isn't specified, DBLLIBRARY is used.

- We added the /ALTSTORE compiler switch to provide support for VAX-DIBOL-compatible zoned stores by translating spaces to zeros during alpha-to-numeric and decimal-to-decimal stores. You no longer have to rebuild the runtime to get this support.
- We added the /find_lock compiler option (**-f** on Windows and UNIX systems). When this option is specified, the FIND statements in the files being compiled will default to locking their records as VAX DIBOL did.
- If you specify the /variant compiler option (**-v** on Windows and UNIX systems) with no value, it will default to 1.
- The compiler will not automatically size an integer variable or literal to **i8** if its initial value exceeds the maximum signed 32-bit value. Instead, if the initial value exceeds this maximum, the compiler generates an error. If it doesn't exceed this maximum, the compiler creates an **i4** variable or literal.
- Synergy DBL supports EXTERNAL and GLOBAL LITERAL statements.
- If you define a literal that is a valid field type/size specification, the compiler will not replace a field's type/size specification with that literal. In the following example, the compiler does not replace the "**a12**" in the fld definition. (If **a12** were defined as a .DEFINE identifier instead of a literal, the compiler would replace the "**a12**" in the fld definition.)

  ```
  literal
      a12    ,a*    ,"abcd"

  record
      fld    ,a12
  ```

- GROUPed subroutine and function arguments have been implemented. For example, a subroutine header can be declared as the following:

  ```
  .subroutine myroutine
  arg1 ,a
  group arg2 ,a
      fld1 ,d5
      fld2 ,[8]a3
  endgroup
  arg3 ,a
  etc...
  ```

  where **arg2.fld1** will reference the first five characters of the currently passed second argument, **arg2.fld2[3]** will reference characters 12 through 14, and so forth. Note that, with the exception of the argument itself being a dimensioned group, all valid syntax for a GROUP is allowed (multi-dimensioned fields, sub-GROUPs, dimensioned sub-GROUPs, and so forth).

- .INCLUDEing from the ICS Data Dictionary now processes ALIAS structures.
- The compiler now allows identifiers of 31 characters and filenames of 255 characters.
- Synergy DBL defaults the size of integer literals to **i4**. For example, in the following definition, lit would be created as an **i4**:

  ```
  literal
      lit  ,i  ,2
  ```

- If the routine name specified in a .MAIN compiler directive starts with “MAIN$”, “MAIN$” will not be appended to the beginning of the specified name.
- We implemented the NOSUFFIX qualifier on the COMMON statement.
- Synergy DBL allows you to overlay a named literal with another literal.
- We now support the packed (**p**) data type, with the following rules and constraints:
  - A GROUP cannot be type **p**.
  - A LITERAL cannot have a **p** field within its scope.
  - Whenever a **p** or **p.** field is used in an expression, it will be converted to a **d** or **d.** field, respectively.
  - Because of the above conversion rule, the only way to pass an expression to a subroutine or function in packed form is to assign it to a packed field prior to its usage.
- If you compile a source file that is not in the current directory, as shown in the example below, the object and listing files will be placed in the current directory:

  ```
  dbl SRCDIR:file
  ```

Data compression on OpenVMS

If you are updating from versions prior to 5.3.9, you may want to re-create and reload your ISAM files to take advantage of data compression. In previous versions, the ISAMC subroutine did not use compression. By compressing your files, you can reduce their sizes by 50 to 75 percent. You may also increase access speed to these files.

As an alternative to recreating and reloading your files, you can do the following:

1. Create an FDL file:

   ANALYZE/RMS/FDL/OUT=*filename*.FDL *filename.ext*

2. Edit the FDL and add data record and data key compression:

   EDIT/FDL/ANALYSIS=*filename*.fdl/NOINTERACTIVE *filename*.fdl

3. Reconvert the file:

   CONVERT/FAST/NOSORT/FDL=*filename* *filename*.ism *filename*.ism

4. Repeat steps 1 through 3 to finally reduce the size as a result of the EDIT/FDL optimization.

Debugger

We made the following changes to the Synergy debugger:

- We implemented the ability to debug and list program modules that are .INCLUDEed. These modules now show the line number in the form *module.line*. You can also set breakpoints to these lines.
- GROUP, %REF, and %VAL routine arguments can now be examined, and all except %VAL can be deposited.
- (OpenVMS) You can set breakpoints in shared images.
- You can examine integers as alphas and alphas as integers, and you can get the address of a variable by using the "!" or "@" sign after the variable name.
- We implemented watchpoints.
- We added a new debugger that supports source line access, line numbers, breakpoints, examination by offset, and named access to fields, including complete variable path specifications.
- We changed the debug compiler option (**/debug** on OpenVMS and **-d** on all other systems) to do the opposite of what it used to do. Previously, the debug option disabled the creation of symbolic access tables. Now symbolic access tables are not created by default, and the debug option will cause the symbol information to be output. You must also set the debug option in the linker if you want symbolic name and source file access.
- The debugger EXAMINE command will now display all non-printing, alpha field data as a ".", and the LIST command will do the same except for BS, HT, LF, VT, FF, and CR.

Errors

We made the following changes to Synergy errors:

- (Windows, UNIX) If more than 20 errors are reported by a terminal input operation with no intervening successful operation, the runtime now reports an $ERR_DEVNOTRDY rather than an $ERR_IOFAIL error.
- We switched the error numbers for $ERR_FILOPT and $ERR_IOMODE. FILOPT is now 21, and IOMODE is 108. We suggest you use error mnemonics instead of error numbers for more portable code. (5.1)

- The following error numbers have been changed: (5.1)

Error	Old number	New number
$ERR_DBLRTLERR	2005	102
$ERR_FILOPT	–	21
$ERR_IOMODE	–	108
$ERR_NOCALL	2002	2
$ERR_NOMEM	2001	9
$ERR_RECEXTCAL	326	5
$ERR_TIMOUT	325	111

- Synergy DBL generates a "Value out of range" error ($ERR_OUTRNG) when a qualifier on statements such as OPEN, LPQUE, and SORT is out of range.
- The following errors have been added:

Literal	Number	Message
$ERR_EXQUOTA	106	A system quota has been exceeded.
$ERR_DEVNOTRDY	107	A device is not ready (or is off line).
$ERR_QUEUENOTAVA	122	Invalid queue specified for LPQUE.

QUEUENOTAVA is a catchall for most print/batch symbiont errors. Use %SYSERR to retrieve the system error code to decode these errors further.

Installations on OpenVMS

- **DBLINSTALL.COM** now allows you to install the runtime only (without the compiler), and it allows you to install the runtime resident.

Message manager on OpenVMS

- We added the file **DBLDIR:msgmgrcmd.com** to our distribution. This file is required for message manager start-up.
- The Synergy message manager now writes fatal errors to the log file **DBLDIR:dblmsgctl.errp**.

Runtime

We made the following changes to the Synergy runtime:

- If you use ^REF routine arguments with data type integer and no size, you must now specify a size.
- If you specify the RFA:*nnn* qualifier in a WRITE statement, the runtime now reports an error.
- A WRITE ^LAST qualifier now generates an error.
- If you OPEN a mailbox device in O:P mode, the runtime will now generate an error.
- If you specify a WRITE with no record number to a relative file, the runtime now writes the current record instead of the next record.
- The FLAGS subroutine's runtime option flag 4 is no longer set if the terminal has SCOPE characteristics when system option #39 is set. System option #5 still sets the FLAGS subroutine flag 4 when option #39 is set.
- We added the $ERR_ARGMIS error when an argument is missing to complement NULARG.
- We added /stream to the OPEN statement OPTIONS qualifier string.
- We improved performance of data reference operations (^) and subscripting in the runtime.
- We added support for **i8** integers on non-64-bit systems.
- IF (dvar) and IF (.NOT.dvar) now allow "**d**" data type variables longer than 18 digits.
- We added the error "Invalid file organization" ($ERR_FILORG, #103), which is generated if you open a file whose organization is different than that specified in the OPEN mode.
- We added the ability to profile your programs.
- (OpenVMS) Performance of XCALLs to Synergy DBL subroutines has been improved on Alpha systems. Performance of XCALLs to Synergy DBL subroutines in shared images has been improved on VAX systems.
- Performance of XCALLs to Synergy DBL subroutines that do not have an ONERROR statement has been improved. This modification requires recompilation to take advantage of the improvement. This change will be most apparent on OpenVMS VAX systems.
- (DOS) The Btrieve interface is now supported in the DBL runtime. Unlike previous Btrieve support, this does not require a separate set of DBL executables.
- (UNIX) We added /nodelay to the OPEN OPTIONS string.
- (UNIX) The O_NDELAY flag is now turned off after a device is opened so that I/O to the device does not automatically generate a "Failure during I/O operation" error.
- (Windows, UNIX) We implemented FIND ^FIRST for non-ISAM files on Windows and UNIX systems.
- (OpenVMS) We added /stream to the OPEN OPTIONS string.
- We now support the ## (round without truncation) operator.
- We now support the // (divide giving implied result) operator.

- We added values for the BUFNUM, BUFSIZ, RECTYP, SHARE, and DEQ qualifiers to the OPTIONS qualifier of the OPEN statement.
- You can specify the following syntaxes on the CLOSE statement:

  ```
  CLOSE chan, chan, chan
  CLOSE chan THRU chan
  ```

- You may specify a label on an EXIT statement. This label must be a label on a BEGIN statement in which the EXIT statement is contained. For example, the following

  ```
  EXIT lbl
  ```

  branches to the first statement beyond the END statement corresponding to the BEGIN statement with lbl on it.

- The FIND statement syntax allows all valid VAX DIBOL and Synergy DBL syntax. The following is the basic FIND syntax:

  ```
  find(chan, find_args) [io_error_list]
  ```

  Find_args can be any of the following:

  key_spec

  qual_list

  key_spec, qual_list

  rec_buf, key_spec

  rec_buf, key_spec, qual_list

  rec_buf, , qual_list

  , key_spec

  , qual_list

  , key_spec, qual_list

  , , qual_list

  where

  - *key_spec* is either ^FIRST, ^LAST, or an expression whose result is the key to find.
  - *qual_list* is a qualifier list.
  - *rec_buf* is a variable for the record buffer.

  Note that *find_args* cannot be

  rec_buf, qual_list

  In this case, *rec_buf* will be incorrectly interpreted as a *key_spec*. Instead, you must specify

  rec_buf, , qual_list

- Synergy DBL supports the FLUSH statement.
- The IO qualifier for the OPTIONS qualifier on an OPEN statement will override the OPEN mode specified.
- You can specify an alpha expression for the mode on an OPEN statement. For example:

  ```
  open(ch, "i:i", file)
  ```

  You can specify i:* for the mode on an OPEN statement. It is equivalent to i:i.
- You can specify the following qualifiers on the OPEN statement:

  CONTIG
  BUFNUM
  DEQ
  BKTSIZ
  BLKSIZ
  RECTYP
- You can specify a value with the XRETURN statement. For example:

  ```
  xreturn value
  ```

Subroutines and functions

We made the following changes to Synergy subroutines and functions:

- If you use the WAIT subroutine or %WAIT function, you must relink your applications before or with 5.7.4. In 5.7.4, support for the old WAIT subroutine will be removed, and failure to relink will cause access violations. (This only applies if you have not relinked since 5.3.12.)
- Changes to the DELET and RENAM subroutines cause additional checking of filenames to occur. A filename containing spaces now generates an error $ERR_FILSPC (17).
- We improved the performance of string arithmetic, %STRING, %CHAR, %ATRIM, and "S_" subroutines.
- We implemented the following new subroutines:

ASTRST	Restores the contents of work areas used as a result of an AST.
ASTSAV	Saves the contents of work areas used as a result of an AST.
DBL$SETCTL	Modifies the operation of control characters.
ERRMOD	Returns the module in which the most recent error occurred.
GETDFN	Returns the default file specification.
LOCALIZE	Localizes currency.
SETDFN	Sets the default file specifications.
STTY	Controls terminal settings.

- We implemented a form of routine overloading using extensions to XSUBR and a new function %XADDR.
- The GETLOG subroutine now only translates logical names, rather than logical names and DCL symbols.
- Arguments passed to %ATRIM are now restricted to alpha data types. This modification corrects a situation that resulted in incorrect errors and unpredictable results. Arguments passed to %TRIM were previously restricted to non-packed types.
- (Windows, UNIX) We fully implemented the %WAIT function and the WAIT subroutine on Windows and UNIX systems.
- (Windows, UNIX) The GLINE subroutine now supports the *prompt* argument on Windows and UNIX systems.
- (Windows, UNIX) We added the PARSE, BEGFL, and RSTATD subroutines.
- (UNIX) We added support for the SETLOG subroutine to reset the runtime internals when TERM or DBLCASE are reset.
- %DESCR is supported for declaring arguments.
- %ERROR is equivalent to the ERROR subroutine.
- Parentheses are not required for %REF and %VAL when declaring arguments.
- Synergy DBL supports ^REF (%REF) on subroutine argument declarations. If the type of the ^REF argument is **a** or **d**, you must specify a size. If the type is **i** and no size is specified, size 4 is assumed. Type **n** is not allowed.
- ^ARGA(*arg_position*) accesses the specified argument as alpha data type. ^ARGA is equivalent to ^ARG.
- You can specify all data reference operations (^ARG, ^SIZE, and so forth) with a caret (^) or with a percent sign (%) (%ARG, %SIZE, and so forth).
- You can specify unquoted arguments with ^X. For example:

  ```
  ^x(F4)
  ```

- Synergy DBL supports the third argument (length) for the GETFNM subroutine (or DBL$GETFNM). This subroutine is the same as FILNM.
- The following intrinsic functions are supported:

%CPUTIME	Returns the accumulated CPU time.
%DEFINED	Determines if an argument has been defined.
%ERLINE	Returns the line number of the statement that caused an error.
%INT	Returns the whole number in a numeric expression.
%JBNO	Returns the current job number.
%PACKED	Converts a numeric expression to packed decimal data.

%RFA	Returns a record's address.
%SYSERR	Returns the system error associated with the last trappable error.
%TNMBR	Returns the terminal number.
%VARIANT	Returns the value of an internal compiler variable.
%VERSN	Returns the Synergy DBL version number.
%VMS	Indicates if a program is being compiled on OpenVMS.
%WAIT	Suspends program execution.
%WKDAY	Returns the day of the week.
%XTRNL	Passes an argument by external reference.

- We added the following external subroutines. (The subroutines that begin with "DBL$" can be called without the "DBL$". For example, DBL$CPUTM can be called and is documented as CPUTM.)

DBL$CPUTM	Returns accumulated CPU time.
DBL$ERRTXT	Returns an error message.
DBL$GETDFN	Returns the default file specification.
DBL$GETFA	Returns file attributes.
DBL$JBNAM	Provides the image name of the calling program.
DBL$PARSE	Extracts components of a file specification.
DBL$SETKRF	Sets the key of reference for the next operation.
DBL$TTFLGS	(first three options) Sets terminal-oriented flags used by the runtime library.
DBL$XARGS	Determines the status of arguments passed to a subroutine.
EXITERROR	Exits the routine with a specific error.
JBNO	Provides the job number for the current process.
OPENELB	Makes an ELB's subroutines available to the executing program.
RSTATD	Returns numeric information about the last record read.
TTNAME	Returns the name of the terminal running the program.
XSUBR	Invokes an external subroutine with the specified arguments.

- Calling the FILNM subroutine on a channel opened to TT: returns TT:.

- We added the INITPORT subroutine on UNIX systems.
- The third argument for the ERROR subroutine returns the system error number. For OpenVMS, the optional fourth argument returns either the RMS STV value from an RMS error or additional system-specific information.

Terminal numbers

- Terminal numbers can be up to six decimal digits long.

C interface

- The "DESCR" typedef (DBL data descriptor) in the Synergy DBL C interface functions has been changed to "DESCRIP." (5.1 or 5.3)

Miscellaneous

- (OpenVMS) If you are updating to Synergy DBL 5.7 on an OpenVMS system, we recommend that you recompile and relink to take advantage of new features and optimizations. Also, if you do not recompile, you will not be able to debug your programs.
- (OpenVMS) **Sortparms.dbl** has changed. You should use the new version if you use a local modified copy of this file.
- Existing main routines will no longer generate a "Return without call" error (NOCALL) if you specify an extra return statement. Once you recompile the main routine, it will correctly generate this error.
- We added a VAXDBL subroutine directory below DBLDIR which contains unsupported DBL routines to help you in your conversion of Synergy DBL for OpenVMS code to UNIX and MS-DOS. These routines include GETFA, SORT, **Sortparms.dbl**, and GETE (to use in place of DBL$TTFLGS(1000) in conjunction with ACCEPT).
- (DOS) The **CHKLOCK** utility is now being included in the MS-DOS distribution. However, the **-p** and **-r** options are not available on MS-DOS. The only option available on MS-DOS is **-b**, which specifies the file block size.

Version 5.1/5.3

This section briefly describes new features in Synergy Language (Synergy DBL) version 5.1 (UNIX) and 5.3 (OpenVMS) and the changes in these versions that may break your code. *Test your system fully.* Some of the features new to version 5 will change logic and cannot be detected at compilation time.

Features that may break your existing code

New features that may affect your code in the compiler

The following features may make your code compile differently:

- If you use the rounding operator (#) and the value being rounded or the round value is implied-decimal, the compiler will report an "Only integer and decimal operands allowed ('#')" error (NOFXD).
- You can no longer use the percent sign (%) to specify a screen function in a DISPLAY statement. Screen functions are not true intrinsic functions; thus, they can only be specified with the dollar sign ($).
- If you're including the Synergy Developer's Toolkit file **tools.def** in your Synergy DBL program, you can no longer use the %TRUE and %FALSE intrinsic functions.
- Each global data section must have exactly one ", INIT" in the set of references to that global data section.
- We implemented full text replacement with .DEFINE. Defined symbols will now replace all identifiers, not just literals.
- You can no longer specify the size of an argument in its declaration. (For example, "arg ,a4" will now generate an error.) In previous versions, the size was ignored.
- If you omit the size in a field declaration, the size specification will default to *. (For example, "fld ,a" will now default to "fld ,a*".) In previous versions, if no size was declared, the size would default to 1. When you have a size specification of *, you must declare an initial value.
- You can no longer use a colon (:) on formatted stores from implied-decimal-to-alpha (for example, "avar = fvar:format_string"). Instead, you must use a comma (,). Also, version 4 allowed you to specify an alpha expression as the source in a formatted store statement. In version 5, the compiler generates a "Numeric expression required" error (NUMREQ).
- We made some changes to the compiler, including modifying some defaults and defining some new compiler options. See "Compiler" on page 1-96 for more information.
- You can no longer divide by 0 in a compile-time expression.
- You can no longer use a comma (,) to continue a FOR statement across physical source lines.
- In FOR statements with the increment form (FOR/UNTIL/DO), the FOR and the DO can no longer be on separate lines unless the DO line begins with a continuation character (&).

- We changed the default IF statement format to the ANS DIBOL form IF/THEN/ELSE: the THEN and the ELSE are required, and the ELSE matches the last THEN (not the last IF, as it did in DBL versions 1 through 4). A true value causes the THEN statement to be executed, while a false value causes the ELSE statement to be executed. You may need to specify the Alternate IF compiler option or set system option #31 to make your code compile properly.
- Synergy DBL now strips non-significant leading or trailing zeros in decimal and implied-decimal literals prior to processing.
- The maximum size of a logical line is now 1023 characters, excluding comments, continuation characters (&), and preceding and trailing spaces on each line.
- We now allow only the following abbreviations for the MERGE and SORT statement qualifiers:

  REC for RECORD

  OUT for OUTPUT

  OPT for OPTIONS

  TEMP for TEMPFILE

  IN for INPUT

  In version 5, the MERGE statement requires at least two files as input.
- An operator can no longer have spaces between the periods (.) and the letter characters.
- We no longer clear our symbol definitions after a routine is processed. If you redefine a symbol in a subsequent routine, the compiler will now generate a warning. Use the symbol definition clearing option in the compiler to clear the symbol definitions after each routine is processed.
- You can no longer place a trailing minus sign (-) on an initial value.
- In WHILE statements, you can no longer place the WHILE and the DO on separate lines unless the DO line begins with a continuation character (&).

New features that may affect your code in the runtime

The following features may make your code work differently in the runtime:

- ^ARG (%ARG) now returns an alpha value. This may affect your code if you are accessing numeric data with ^ARG. Use ^ARGN instead.
- An alpha expression will now evaluate to false only if it is null or if it only contains ASCII spaces. In all other cases, it will evaluate to true. Therefore, an alpha expression that is resolved to a value of 0 will now evaluate to true.
- DOS BRDCST, MODLST, NETREQ, and SETBC subroutines no longer exist.

- We changed our C interface runtime interface functions. We now require you to pass the argument block pointer to C interface runtime interface functions. (This argument block pointer gets passed to your C routine.) We also changed the names of the runtime interface functions to help eliminate the possibility of calling the new ones with the wrong arguments. The old routines no longer exist.

  Note that you must include the file **xcall.h** at the beginning of your C routine. **Xcall.h** defines the typedefs that are included in the function syntax below.

  The new functions are as follows:

  `dblerror(`*int error*, *char* `*`*string*`)`

  `dbl_exec(SEGENTRY *`*segment*, `DESCR **`*argblock*`)`

  `get_chn_sts(`*int channel*, `DBL_CSTAT *`*structure*, *char* `*i)`

  `get_ernum(`*char* `*`*error*, *int length*`)`

  `get_gbl_ptr(`*char* `*`*global*`)`

  `get_seg_ptr(`*char* `*`*routine*`)`

  `get_xarg(DESCR **`*argblock*, *int argument*, `DESCR *`*descriptor*`)`

  `get_xarg_cstr(DESCR **`*argblock*, *int argument*, *char* `*`*buffer*, *int length*, *int strip_flag*`)`

  `get_xarg_val(DESCR **`*argblock*, *int argument*`)`

  `put_descr(DESCR *`*destination*, `DESCR *`*source*`)`

  `put_xarg(DESCR **`*argblock*, *int argument*, `DESCR *`*source*`)`

  `put_xarg_str(DESCR **`*argblock*, *int argument*, *char* `*`*string*, *int length*`)`

  `put_xarg_val(DESCR **`*argblock*, *int argument*, *int value*`)`

In Synergy DBL 5.7, the DESCR typedef was changed to DESCRIP.

  When using our C interface, we recommend that you use our runtime interface functions (rather than directly accessing arguments).

- With the addition of exclusive file locking, the runtime now posts one conventional lock on each file opened (including files open for input). This may require you to increase the number of locks allowed on your system. You can use system option #33 to temporarily disable these locks to ease your conversion to Synergy DBL 5.

- We no longer allow file specifications of the form "$LOG/file". In version 5, you must instead use the following format:

  ```
  open(1, o, "LOG:file")
  ```

- Synergy DBL now stores data declared as global or common in a separate space from data that is local to a main routine.
- Intrinsic functions that return numeric values now return integer values (instead of decimal). One side effect of this change is that if you pass intrinsic return values to a subroutine (for example, "xcall sub(%size(fld))"), either you need to declare the corresponding argument as data type integer (**i**) or access type numeric (**n**), or you can use the numeric argument compiler option to map all decimal-type arguments to numeric.
- An ISAM file record must now be at least four characters long.
- (Windows, UNIX) The ISCLR subroutine no longer closes the channel if the channel is open. If you include the channel argument, it is ignored.
- When you open a file in output mode, the OPEN statement will create the file (and overwrite any existing file with the same name). In version 4, the OPEN statement created a temporary file then renamed the temporary file to the specified name on close; if the file wasn't closed, the temporary file was purged on exit. If you want the OPEN statement to act as it did in version 4, you must specify the TEMPFILE qualifier, OPEN(*ch, mode, file,* TEMPFILE), and explicitly purge the file using the PURGE statement if you don't want to keep it.
- We now have three separate subroutine libraries: DLIB, ILIB, and ULIB. DLIB and ILIB are system-supplied libraries that contain external subroutines and intrinsic functions, respectively. ULIB will only contain user-defined functions.
- Truth values are now evaluated from left to right.
- The UPCASE and LOCASE statements no longer convert special characters (unless you use the LOCALIZE subroutine, which was added in Synergy DBL 5.7).
- The return status on the GETLOG subroutine, which translates a logical, is now the length of the logical. If no logical is found, the return status is 0. (This feature was included in DBL versions 4.41 and 4.5.)
- The GETRFA subroutine no longer returns the RFA from a FIND statement. You must now use the GETRFA qualifier on the FIND statement.
- The DELET subroutine on an ISAM file now deletes both the data and the index file if you specify the extension. (In previous versions of DBL, you could also specify a channel as the first argument. This syntax is still valid, but the channel argument is ignored.)
- The RENAM subroutine on an ISAM file now renames both the data and the index file if you specify the extension. (In previous versions of DBL, you could also specify a channel as the first argument. This syntax is still valid, but the channel argument is ignored.)
- The RENAM subroutine now allows and processes a search list on the second argument.
- (OpenVMS) The TNMBR subroutine always returns negative numbers on OpenVMS for detached and batch jobs. The TNMBR DCL symbol is ignored in this case.

New features

Assignment statements

- Synergy DBL's data formatting routines now align with the ANS DIBOL definition for formatting data. We made the following changes:
  - If a format doesn't contain an explicit "X" format control and the value is 0, all format control characters except the "*" will be suppressed (blanked).
  - Except for a comma (,), all format control characters to the left of a "$" will be considered non-format characters (and displayed within the result).
  - A "Z" to the left of a "*" that would be blank according to a given value will now be a "*".
- If an alpha source contains a valid implied-decimal number (for example, 1.4), under Synergy DBL version 4, an assignment to a numeric field gave a bad digit error. Under version 5, no error occurs, and the implied-decimal source is rounded and stored as a whole number in the destination. If truncation is required, the TRUNCATE qualifier must be used in the .MAIN or .SUBROUTINE compiler directives.

C interface

- C routines can now call DBL routines.
- Your distribution contains the files **rulib.xll**, **rxcalls**, and **makedbr** and the library ULIB to help you build your Synergy DBL runtime with additional C routines.
- The order of subroutines in your **rulib.xll** file (which was the **rdlib.xll** file in version 4) is no longer important. Routine names are alphabetized for binary search at runtime.

Compiler

We made the following changes to the Synergy compiler:

- We eliminated the **-N** trace flag option when you use a command file to input file names to the compiler or the linker.
- If you omit an argument to one of the compiler options, the compiler will use the next undefined element on the line as that argument. Other compiler options and the separator symbol (--) will no longer be used as filenames.
- We changed the compiler syntax to include one or more list options.
- The compiler now includes the file **dbl.def** in each Synergy DBL program as if you specified it in a .INCLUDE statement as the first line in your program. If you end a routine with the END statement (instead of .END), the **dbl.def** identifiers will be cleared along with any other identifiers you've defined. If you end your routines with END and you want to use the symbols defined in **dbl.def**, you must either explicitly include **DBLDIR:dbl.def** in each routine (after the first routine) that will use the identifiers, or change all of your final END statements to .ENDs and remove duplicate .INCLUDE statements throughout the file.

- We added the DBLMAXERR environment variable. It enables you to specify the maximum number of errors the compiler will generate before aborting with a "Too many errors" error (ERRCNT).
- Only descriptors for referenced variables are now emitted to the object file. (This feature reduces the size of your object and runtime files, as well as runtime memory requirements.)
- Line numbers have been changed to use source line numbering.
- Synergy DBL version 4 listing anomalies (page breaks, for instance) have been eliminated.
- We automated parsing. Most Synergy DBL syntax is now defined using a "context-free" grammar, and that grammar is processed by a program that generates a parsing routine.
- We changed the name of the "Program section sizes" table in the compiler listing to "Memory Usage Summary."
- We added or changed the following compiler options:

/debug or **-d**	Debug. Create symbolic access table.
/gbldefs or **-g**	Global definitions. Don't include the **dbl.def** file in a Synergy DBL program.
/show=noheaders or **-h**	Header listing. Indicates that headers should not be printed to the listing.
/offsets or **-**i	Symbol table offsets. Prints a table that contains offsets into the symbol table.
/noobject or **-n**	No object file. The compiler won't generate an object file.
/decargs or **-N**	Numeric argument. Maps all decimal arguments to numeric type.
/warnings or **-W**	No warnings. Won't generate any warnings.
/warnings=2 or **-W2**	DEFINE warning. Suppresses warnings if you redefine a symbol or try to .UNDEFINE a symbol that isn't defined.

- If your code segment reaches 64K in size, the compiler will generate a "Code segment too big" error (SEGBIG).

Compiler directives

We made the following compiler directive changes:

- You can now .INCLUDE elements from the ICS Data Dictionary in your Synergy DBL program.
- If the filename that you specify in the .INCLUDE directive is not an alpha expression, you may need to use the DBLCASE environment variable to ensure that the compiler won't convert your file names to uppercase (since it automatically converts all non-quoted strings to uppercase for processing).

- The .LINE compiler directive is ignored because Synergy DBL now uses physical line numbers instead of the logical line numbers used in previous versions.
- The .PAGE compiler directive no longer appears in a listing of source code. You'll notice a skipped line number in the listing where the .PAGE directives are. The compiler will not generate a new page until a source line is listed. Two consecutive .PAGE directives will not generate two pages.
- The argument in the .START directive that specified an optional title to be printed at the top of all successive pages of the compiler listing no longer sets the title. .START without any options defaults to a page break. .START does not default to a page break if any options are present. A .START LIST line that turns the listing back on will not be listed.
- In previous versions of Synergy DBL, the compiler interpreted all of the text that followed .TITLE (excluding leading white space and regardless of form) as the listing page header. The ANS DIBOL standard specifies that a quoted string is the optional argument to .TITLE. So you won't have to modify your existing code, Synergy DBL 5 will use the "old" form of .TITLE if the first item after .TITLE is not a quoted string, or the ANS DIBOL form if the first item is a quoted string. This will only cause a problem if your code contains a .TITLE with a quoted string as the first item with additional items following the quoted string.
- We added the following compiler directives:

.ALIGN	Aligns the data location counter to a specified boundary.
.FUNCTION	Creates user-defined functions.
.LIST	Controls the listing of source code.
.NOLIST	
.MAIN	Indicates the beginning of a main routine.
.UNDEFINE	Removes the definition of a replacement identifier.

Data division

We made the following changes that affect the data division:

- We changed the COMMON statement so that it now has two formats: GLOBAL and EXTERNAL. The GLOBAL COMMON statement can define a shared data record and its component fields in either the main routine or a subroutine.
- A global data section name must not have the same name as a COMMON symbol.
- Be careful when changing the size of global data sections or COMMON symbols. 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, execution of these programs could cause memory access violations.
- The compiler now allows you to specify initial values on field declarations that have a field position indicator (@position). Initial values are only allowed when the position in the record doesn't overlay any previously declared areas.

- Synergy DBL will generate an error if a named record or field is larger than 65,535 bytes.
- We added the following new statements:

  EXTERNAL FUNCTION Declares a user-defined function.

  GROUP Defines a group.

  LITERAL Defines a local data structure that can't be modified during program execution.

- We added an optional STATIC qualifier to the RECORD statement. If you specify the STATIC qualifier in a subroutine or function, the contents of the static record upon entry will be the same as they were when the routine last exited. If you don't specify STATIC, the record's contents may be reinitialized during program execution. Because data defined in the main routine will never be reinitialized, it is not necessary for you to use the STATIC qualifier in the main routine.

Data reference operations

- We added several new data reference operations, which enable you to access data as any data type and to determine various data characteristics:

  ^A Accesses data as an alpha type.

  ^ADDR Obtains the address of a data element.

  ^ARG Returns the specified argument as an alpha data type.

  ^ARGN Returns the specified argument as a numeric data type.

  ^D Accesses data as a decimal or implied-decimal type.

  ^DESCR Passes an argument by descriptor.

  ^I Accesses data as an integer type.

  ^PASSED Determines if an argument has been passed.

  ^REF Passes an argument by reference.

  ^SIZE Returns the size of an expression.

  ^VAL Passes an argument by value.

- Note that the intrinsic forms of the above functions are still valid. To make the conversion process to version 5 a little easier, Synergy DBL will automatically map those intrinsics to the appropriate data reference operations.

Data types

We made the following changes for data types:

- Synergy DBL now recognizes an integer data type, which is a byte-oriented, binary representation of a signed whole number.
- Fixed-point decimal values are now called "implied-decimal" values. They are designated in your data division as follows:

  `D`*total_size.decimal_places*

  Synergy DBL will still recognize any data type designations in your existing code that begin with "f."
- Synergy DBL now recognizes a numeric data access type for subroutine argument and function declarations (for example, "arg ,n"). If the access type is **N**, the passed data will be accessed as numeric. If the passed data type is numeric, it will be accessed as it is typed in the calling routine. If the passed data type is alpha, it will be accessed as decimal type.

ELBs and OLBs

- The ELB filename that you specify while linking will be used at runtime to open the ELB.
- The syntax for ELBs is the same as in version 4. Some semantic differences are as follows:
  - You can use an OLB as input to an ELB being created with **dblink**:

    `dblink -l` *xxx yyy*`.olb`
  - ELBs now support any type of subroutine, including those with common records, global data sections, and any number of external routine references.
  - You can update subroutines in an ELB without relinking any programs that access the subroutines in the ELB.
- When you use the librarian to list the routines in an OLB, it now lists them in sorted order.
- We added the **listelb** utility, which lists information about ELB routines.
- An OLB can now contain a main routine.
- We enhanced the linker so it 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.

Errors and error trapping

- We no longer append error numbers to runtime error messages. You can now specify any runtime error with the $ERR_*mnemonic* specification (for example, $ERR_INVOPT). We recommend that you use this specification instead of error numbers in your programs.
- With the exception of catastrophic errors such as stack overflow and internal failures, you can now trap all runtime errors.

Expressions

We made the following changes for expressions:

- The equal sign is now a valid operator in an expression. The left side must be a variable specification, and the right side is treated as an expression and stored in the variable before it is used in the rest of the expression.
- We added bitwise operators, which perform Boolean operations on the bits within integer operands. (Decimal operands are also allowed, and Synergy DBL will convert them to integer operands before operating on them.) The following bitwise operators are now available:

.BOR.	Bitwise OR.
.BXOR.	Bitwise exclusive OR.
.BAND.	Bitwise AND.
.BNAND.	Bitwise NOT AND.
.BNOT.	Bitwise NOT.

- We added Boolean left-to-right evaluation. If an .AND. or .OR. Boolean operation can be determined by the evaluation of its left-hand operand, Synergy DBL won't process the right-hand operand.
- You can now specify any expression that can be completely evaluated at compilation time in your data division. No dimension, index, or range specifications are allowed in a compile-time expression, regardless of whether or not their components are literals.
- Where possible, expressions will now be evaluated at compile time.
- The compiler will optimize expressions as much as possible. Terms may be shifted, and literals may be "folded" at compilation time.
- We added string concatenation and reduction. The addition operator (+) appends the operand on the right to the operand on the left. The subtraction operator (-) removes the first occurrence of the operand on the right from the operand on the left.
- We added string relational operators to compare alpha operands. The following string relational operators are now available:

.EQS.	Equal to
.NES.	Not equal to
.GTS.	Greater than
.LTS.	Less than
.GES.	Greater than or equal to
.LES.	Less than or equal to

  These operators compare alpha operands according to the order of characters within the ASCII character set.

Intrinsic functions

We made the following changes to intrinsic functions:

- In version 5, %B, %O, and %X require parentheses around the argument.
- %BIN, %HEX, and %OCT convert from the numeric data type to a 32-bit longword value. If the number being converted is greater than the value that will fit in the 32-bit integer, you'll get an "Arithmetic operand exceeds maximum size" error ($ERR_BIGNUM).
- The %FLD intrinsic function is no longer provided.
- In version 4, %RND returned a decimal type variable. In Synergy DBL 5, it returns the same data type as the variable being rounded.
- The intrinsic %TRIM(" ") of a null string returns 0, while the intrinsic %TRIM(" ") of blanks returns 1. (The latter is for historical reasons.)
- We added several new intrinsic functions:

%ABS	Returns the absolute value of a specified numeric expression.
%ATRIM	Returns a specified alpha expression with its trailing blanks stripped.
%B	Returns the integer value of a specified alpha expression evaluated as a binary number.
%BIN	Returns a binary string representation of a specified numeric expression.
%BIT_IS_CLR	Evaluates a bit value in a binary representation of a value and returns a 1 if the bit is clear.
%BIT_IS_SET	Evaluates a bit value in a binary representation of a value and returns a 1 if the bit is set.
%BKSTR	Returns the position of the right-most occurrence of a target string within a string.
%CHAR	Returns the ASCII character that corresponds to a specified numeric expression.
%CHOPEN	Indicates whether the specified channel is open.
%CNV_IP	Converts a native-form integer value to portable form.
%CNV_PI	Converts a portable integer created by a previous %CNV_IP function back to a native integer value.
%DATE	Returns the current date in *DD-MMM-YYYY* or *DD-MMM-YY* format.
%DATETIME	Returns the current date and time in *YYYYMMDDhhmmssuuuuuu* format, where *uuuuuu* is microseconds.

%HEX	Returns a hexadecimal string representation of a specified numeric expression.
%IMPLIED	Returns the implied-decimal representation of a specified numeric expression.
%INTEGER	Returns the integer representation of a specified numeric expression.
%KEYVAL	Returns the value of an index file's key from a record.
%O	Returns the integer value of a specified alpha expression evaluated as an octal number.
%OCT	Returns an octal string representation of a specified numeric expression.
%RVSTR	Returns the position of the right-most occurrence of a target string within a string.
%STRING	Returns a string representation of a specified numeric expression.
%TTSTS	Determines whether any input characters are pending on the terminal referenced by the specified channel number.
%UNSIGNED	Returns the unsigned value of a specified integer expression.
%X	Returns the integer value of a specified alpha expression evaluated as a hexadecimal number.
%ZONED	Returns the decimal (zoned numeric) representation of a specified numeric expression.

Synergy DBL originally defined intrinsic function names as $*name*, but to be consistent with the ANS DIBOL standard, function references are now defined as *%name*. Although Synergy DBL will still recognize the $ form in your existing code, we strongly recommend that you switch to the % form.

ISAM

We added several new features to ISAM. You will need to use the new **irecovr** utility to convert your ISAM files before you can use the platform-independent file structure, data compression, and improved RFA access features. To use the multiple fixed-length records, variable-length records, and static RFA, you will need to create a new ISAM file.

- When prompted for the name of the key field while using the **bldism** utility, you can specify /null after the key name. You will then be prompted for the necessary null key information.
- You can now include comments or descriptions in the **bldism** utility by beginning the comment line with a semicolon. All lines that begin with a semicolon will be ignored.

- The **bldism** utility now offers online help. To find out what input is valid for any prompt, just type a question mark (?) after that prompt.
- The **bldism** utility and the ISAMC subroutine have a new static_rfa option. This option causes each record in a file to have a unique RFA throughout the life of that file. The feature is primarily intended for applications that rely heavily on RMS-compatible RFA access or for variable-length records with changing data length.
- Repeated strings of characters can now be compressed to 1 to 4 bytes. Records that contain text fields are ideal candidates for compression. You can save 10 to 50 percent of your disk space without changing your programs.
- You can also add file options after the name of the ISAM file. These options specify the type of file you want to create, whether the file will be compressed, and whether it will have a static RFA.
- When creating ISAM files, you can now specify one of the following file types:

  Fixed-length records (the file type used in DBL 4 ISAM files). All data in the ISAM data partition is stored in records of the same length, regardless of the actual size of the data in the record. If your ISAM file is used for only one data structure, use the fixed-length format.

  Multiple fixed-length records. If your ISAM file is used for a predefined group of data structures, you can use a multiple fixed-length record format. The size of the stored record is determined by the data passed by the STORE statement. This file type enables you to optimize disk storage and reduce the number of open files.

  Variable-length records. 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. The initial size of stored data is determined by the size of the data passed to the STORE statement. If you change the record size during a WRITE statement, the original record may be marked as deleted and the modified record will be written to a new location. This feature saves disk space.
- We added a new utility called **ipar** to generate parameter file descriptions of ISAM files. The descriptions can then be used as input to **bldism** to rebuild the same files, and contain current content information in their comment lines.
- We added a new high-speed ISAM file converter/rebuilder. The cached utility program **irecovr** will quickly convert all of your ISAM files to our new ISAM file structure, which means you no longer have to unload the files to sequential files and then reload them. You can also use **irecovr** to recover deleted file space.
- The **isload** utility now offers online help. To find out what input is valid for any prompt, just type a question mark (?) after that prompt.

- We added four new file specification qualifiers to the **isload** utility. After the prompt

  Enter name of ISAM file to be loaded:

  you can enter the following syntax:

  filename[, qualifier]

  where *qualifier* is one of the following qualifiers:

COUNTED	Specifies that a sequential file will contain counted-byte, variable-length records.
FIXED	Specifies that the sequential file will contain fixed-length records that are the maximum record size.
KEY=*key_no*	Specifies the key to use when unloading the ISAM file.
NOLOCK	Specifies that the file from which you are loading should be opened with no locks.

- We added options to the **ismvfy** utility to make it verify more aspects of the ISAM file's structure.
- We changed the key numbering and terms we use in the ISAM utilities. We now use key numbers 0-7, and we specify the keys as primary key, first alternate key, second alternate key, and so forth.
- The new ISAM format will be the same on all Synergy DBL systems except OpenVMS. As a result, you can now copy ISAM files to any Windows or UNIX operating system and access them without conversion. You can also access ISAM files across heterogeneous networks.
- You can now associate null values with secondary keys. When you specify these secondary keys with their associated null values on a STORE or WRITE, an entry for the record will not be included in that key's index. When you READ on one of these secondary keys, you will not find records with the null value.
- The **status** utility now offers online help. To find out what input is valid for any prompt, just type a question mark (?) after that prompt.
- We added two new file options to the ISAMC subroutine: multiple and static_rfa. ISAMC also now enables you to specify null keys with a new keyspec option.
- The third argument passed to the ISKEY subroutine (which receives status information) can now be any length, and Synergy DBL will move as much information as will fit into the buffer.
- The second argument passed to the ISSTS subroutine (which receives the ISAM file information) can now be any length, and Synergy DBL will move as much information as will fit into the buffer.

Linker

We made the following changes to the Synergy linker:

- We added a debug linker option (**-d**).
- The runtime now allocates arithmetic temporary variables (ATEMPs) dynamically and provides you with as many ATEMPs as your program needs. As a result, we eliminated the **-a** option to the linker, which enabled you to specify the number of ATEMPs to allocate.
- We enhanced the Synergy DBL linker to eliminate conflicts during builds. If you attempt to build a **.dbr** file that is being executed by another programmer, a temporary file is created. When the **.dbr** is completed, the old **.dbr** is removed and the new temporary **.dbr** is renamed.
- We redesigned our object module format, making linking faster.
- We added two linker utilities: **listdbo** and **listdbr**. The **listdbo** utility lists the contents of an object file. The **listdbr** utility lists the contents of an executable file.
- You can now specify a maximum of 256 input files. (Before, you could specify only 129 files to be linked.) The maximum number of input files to the librarian is also 256.

Procedure division statements

We made the following changes to procedure division statements:

- We now generate a "Bad mode specified" error ($ERR_IOMODE) instead of a fatal RUNERR if you specify an invalid mode on an OPEN statement.
- Numeric CASE labels can now be any signed, 32-bit value. (In version 4, they were limited to unsigned, 15-bit values.)
- We added or modified several I/O statement qualifiers. You can use either a colon (for example, LOCK:flag) or an equal sign (LOCK=flag) with these qualifiers:

DIRECTION	Sets the direction of the READS.
GETRFA	Returns the record's RFA after the operation has been performed.
KEYNUM	Specifies the number of the key to access.
LOCK	Sets file locking behavior.
MATCH	Sets matching behavior.
NUMREC	Specifies the number of records to pre-allocate for the file.
POSITION	Sets the position to first, last, EOF, or BOF.
RECSIZ	Specifies the record size to be associated with the file being opened.
RFA	Specifies the RFA of the target record.
SHARE	Sets exclusive or shared access.
TEMPFILE	Specifies the file specification for a temporary file to be used for output.
WAIT	Specifies the time to wait for completion of input.

- For FIND and READ statements, we created a qualifier order of precedence to determine how a record will be found:

Precedence	Qualifier	Explanation
1	POSITION	Takes the highest precedence. MATCH, RFA, and *keyspec* are ignored if they were specified.
2	MATCH	Takes precedence when POSITION is not specified or is ignored. RFA is ignored unless MATCH:Q_RFA is specified. *Keyspec* is ignored when MATCH:Q_RFA or MATCH:Q_SEQ is specified.
3	RFA	Takes precedence when POSITION is not specified or is ignored, MATCH has not been specified, or MATCH:Q_RFA is specified.
4	*keyspec*	Used by MATCH in some cases, but otherwise takes the lowest precedence.

- Synergy DBL now ignores the optional decimal expression (the blocking factor) on the PROC statement. The new .PROC compiler directive is equivalent to the PROC statement.
- If a qualifier on a READ, WRITE, GET, PUT, or STORE defines the key or record number, and the key or record number is not specified, its position no longer has to be held with a comma.
- You can now open to **stderr** (standard error) with the following syntax:

  `open(`*channel*`, `*mode*`, "err:")`

- You can now specify a program exit value on a STOP statement.
- You can now open a UNIX pipe to a subprocess using the following syntax:

  `open(1, o, "|`*filename*`")`
  `open(1, i, "|`*filename*`")`

  where *filename* begins with the vertical bar character (|) and specifies a UNIX command to be executed.

Subroutines

We made the following changes to Synergy subroutines:

- You can now specify any character as an activation character to subroutines ACCHR and DACHR, and you can specify any number of characters.
- We added several new external subroutines:

  EXITE Enables you to exit the current closed routine with a specified Synergy DBL error and specify whether that error can be trapped.

  FREE Releases all record locks on a channel.

  TTNAME Returns the name of the terminal with which the running program is associated.

  XSTAT Gets the return value of the last C function, Synergy DBL function, or subroutine that was called.

- The DATE subroutine has changed slightly. If its argument is at least 11 characters in size, it will now return a date in the form *DD-MMM-YYYY* (for example, 12-Jan-1995). If its argument is less than 11 characters, it will return a nine-character date in the form *DD-MMM-YY* (for example, 12-Jan-95).
- The DTOB subroutine converts the decimal data type to a 32-bit longword value. If the number being converted is greater than the value that will fit in the 32-bit integer, you'll get an "Arithmetic operand exceeds maximum size" error ($ERR_BIGNUM).
- We added a process group ID argument to the JBNO subroutine. It now has this syntax:

  `xcall jbno(`*[id], [parent_id], [group_id]*`)`

- The SHELL subroutine now allows a command line to be executed by a shell. You can also now specify shells other than Bourne and C.
- The SPAWN subroutine now handles simple shell meta characters (for example, input and output redirection).
- We added some functionality to the WAIT subroutine. The syntax is as follows:

  `xcall wait(`*seconds*, *parameter[, event]*`)`

- We now allow search list logicals in path specifications. A search list logical specifies one or more directory paths in which to search for a file.

Synergy daemon – UNIX only

- We replaced the message controller with the Synergy daemon. The Synergy daemon provides the same functionality as the old message controller, along with the new License Manager functionality.

Synergy DBL stack

- We now create dynamic control space on the Synergy DBL stack, which increases stack usage requirements significantly. We changed the default stack size from 1024 to 4096. The minimum stack size is now 512, and the maximum is 16384.

System options

Use the SETLOG subroutine to reset DBLOPT if you want to change any settings.

- We added the following new system options:

#10	Causes the interrupt character typed at the terminal to be ignored at program start-up and between chained programs.
#11	Changes the Synergy DBL default from round to truncate.
#16	Maps the quit character to interrupt.
#28	Causes the compiler to map decimal subroutine arguments (**d**) to numeric (**n**).
#29	Causes Synergy DBL to generate a "Dimension specifications required {var}" error (REQDIMREF) if a dimensioned variable is referenced without dimension specifications ([]).
#30	Forces the runtime to use the default, built-in VT*xxx* terminal definition, regardless of how the TERM environment variable is set.
#31	Specifies that the alternate form of the IF statement will be used (the Synergy DBL IF).
#32	Eliminates the uppercasing and lowercasing of all two-byte characters.
#33	Disables conventional file locks.

Note the following:

- System option #14 is now ignored. (System option #31 now designates that the alternative form of the IF statement will be used.)
- System option #25, which swapped memory on SPAWN, SHELL, and LPQUE, no longer exists.

Utilities

- We added the Synergy Control Panel utility to enable you to modify Synergy DBL message text. This utility replaces the **cherror** utility.

Variables and variable references

We made the following changes to variables and variable references:

- You can now reference the entire scope of an array using the following syntax:

  ```
  array[]
  ```

- You can now pass dimensioned arguments to subroutines and user-defined functions.
- You can now have the same variable name in more than one named record. To reference such a variable, use a variable path specification.
- A variable path uniquely references a named record, a group, or a field within a group. Variable path specifications have the following format:

  [group_name][.group_name....]field_name

  The path can contain as many group names as necessary to create a unique variable reference.

Windowing subroutines

- We added the function WA_INSERT to the W_AREA subroutine. This function inserts a line at the top or bottom of the scrolling area.
- We added two new functions to the W_PROC subroutine:

  WP_RESIZE Resizes the screen to a new specified size.

  WP_OPTION Controls hardware scrolling, which uses the capabilities of the terminal to do the scrolling.

- (UNIX) The **termcap** and **terminfo** databases are accessed for the sequences to set the scrolling region and scroll up or scroll down. If these sequences are not present, hardware scrolling will not be enabled.

Miscellaneous

- DBLLIBRARY is a new environment variable that specifies the directory or search path where the library file can be found.
- REGIS color is not available in Synergy DBL 5.
- We converted to full 32-bit segment sizes and addressing and 16-bit code size.
- LAT support works differently. With version 5, you need to use the LAT subroutine, an example routine in the DBLSTARLET directory that shows the code required to negotiate different connections and the types of timing required.

2

License Manager

Version 9

This section briefly describes new features in License Manager version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- (Windows) We added a new option to lmu, **-xb**, which is used to remove a Backup License Server configuration. It can be run from either the backup server or the primary server. (9.5)
- We added a **-o** option to **lmu**. This causes the output that normally goes to the screen when you run **lmu** without any options to go to a file instead. The syntax is **lmu -o** *filename*. (9.1.1b)

Version 8

This section briefly describes new features in License Manager version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- (Windows) When a backup license server is changed to reflect a new primary server machine, the Synergy Configuration program will now reflect the change. (8.3)
- (Windows) *xf*Server errors that are the result of licensing failures are now logged to the application event log. (8.3)
- (UNIX) License Manager now logs information regarding queues when they reach full state. In addition, License Manager will also attempt to re-send a failed message following such a state. (8.3)
- (Windows) We added some usage warnings that display when configuring a backup license server. (8.1.7e)
- (Windows) License Manager now displays a warning message when only one configuration remains out of the eight configurations allowed per BLSRV license. (8.1.7e)
- (Windows) We added a dependency on the Windows services LanmanWorkstation and EventLog to **synd**. This ensures that services are always started in the proper sequence. (8.1.7c)
- (Windows, UNIX) We enhanced the performance of License Manager when large numbers of licenses are in use. (8.1.5)
- (Windows, UNIX) The **lmu** utility can now display status selectively. To display status for one or more specific products, specify the product name(s) in the **lmu** command (for example, 'lmu RUN8'). There is also a new **-u** option that you can use with this syntax to display information about each license seat currently being consumed. (8.1.3)
- (Windows, UNIX) The usage information displayed by **synd -h** and **lmu -h** has been improved. (8.1.3)
- (Windows) We added a new **synd** option, **-rs**, which enables you to register and then start License Manager server in one step. See "The synd Program" in the "Configuring License Manager" chapter of the *Installation Configuration Guide* for details. (8.1)
- (Windows) The **lmu** program now registers and/or starts a License Manager that has not been registered or started. (8.1)

Version 7

This section briefly describes new features in License Manager version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- (Windows) You can now re-initialize licensing with a different licensee name prior to configuring product licenses. This addresses situations where the licensee name specified when Synergy/DE was installed is different than the one specified when the product was ordered. The syntax on Windows 98/Me (stand-alone) is "lmu -r*NewLicenseeName*". The syntax on Windows NT/2000/XP (server) is "lmu -r*NewLicenseeName* -ns". (This does not apply to license clients because the licensee name is defined on the server.) (7.5)
- (Windows) Version 7.5 includes a new product, the backup license server. The backup license server enables you to continue serving licenses to users in the event that the primary server becomes unavailable. The backup license server is licensed and sold separately. (7.5)
- (Windows) *xf*Server, *xf*ServerPlus, and SQL OpenNet can now be installed on a machine configured as a license client. To enable this feature, the SynLM service must be re-registered (via Synergy/DE uninstall-reinstall or by running **synd -x** followed by **synd -r** using the current version of **synd.exe**). A reboot may be necessary. (7.5)
- The License Manager utility now supports a Synergy Key File (**.skf** file) as input when configuring license keys. A key file contains license configuration keys for one or more products for one or more licensee names. These files enable you to configure all products on a workstation at once, so you no longer have to enter information for each product at the command line. You can obtain a key file from Synergex Online Services or when you request keys by email (Windows only), mail, or fax. (7.3)
- (Windows) The new Synergy Configuration Program has replaced the License Configuration program on the Start menu. To run it, select Synergy/DE > Utilities > Synergy Configuration Program. Use this new program to configure your Synergy licenses. (7.3)
- To help ensure that unique licensee names are specified, the term "Company name" has been replaced with "Licensee name" throughout all Synergy/DE products and documentation that relate to licensing. (7.3)
- The license information displayed by **lmu -b** differs for client and server/stand-alone configurations. (7.3)
- (Windows) We added a new option, **-s**, to the **lmu** program. This option displays the License Manager session and seat IDs, which can be useful when tracking license logins with the DEBUGLOGGING option. (7.3)
- We added several new options to **lmu** to support the new Synergy key file functionality: **-f**, **-t**, and **-m**. (7.3)

Version 6

This section briefly describes new features in License Manager version 6 and the changes in this version that may break your code.

New features

- We changed **lmu** to a console application from a Windows application. If you have been using START with the /w[ait] switch in your Windows 95 batch files for compiling and linking, you no longer need to do so. Now, if you use the START /w switch, an MS-DOS box will open. You must close this manually.
- (Windows, UNIX) We added a new **-k** option to **lmu** to allow reconfiguration of licensing without removing license files or touching the registry. A password is required and can be obtained from Synergy/DE Developer Support.
- (UNIX) We added additional logging options to **synd** (the license manager).
- (Windows) We now provide limited support for License Manager server on Citrix. (Citrix remains a limited support platform because it is based on Windows NT 3.51.)

Version 5

This section briefly describes new features in License Manager version 5 and the changes in this version that may break your code.

New features

- We added a developer-accessible License Manager, which is a set of utilities that controls the installation and use of Synergy products.
- (UNIX) We replaced the message controller with the Synergy daemon. The Synergy daemon provides the same functionality as the message controller, along with the new License Manager functionality.

3

UI Toolkit

Version 9

This section briefly describes new features in UI Toolkit version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to UI Toolkit version 9.

- (Windows) In version 9.5.3, we have improved bounds checking in Toolkit. The distributed UI Toolkit library, **tklib.elb**, is now built with the **-qstrict** compiler option, which means it now meets a higher quality requirement, performing checks that are similar to those done when DTK_BOUNDS=2. This modification may break your code. For example, you may see "Invalid subscript" or "Index out of bounds" errors if fields passed to Toolkit routines are too small. If this is the case, we highly recommend that you fix your code; however, if you have a critical situation where you cannot do that, we also distribute **tklib_nostrict.elb**, which can be used temporarily instead. (This file will *not* be distributed in version 10.)

  We also now distribute a second Toolkit library, **tklib_qcheck.elb**, which is built with the **-qcheck** compiler option. In addition to the checks done above, pseudo arrays in qchecked code are treated as real arrays so that bounds checking can be done on those as well. To take advantage of this enhanced library, you must compile *your* code with the **-qcheck** compiler option as well. (We recommend using SYNCMPOPT to simplify this.) Then link your "qchecked" application against **tklib_qcheck.elb** to highlight places in your code where memory is incorrectly accessed. Note the following:

  - You should always explicitly link against **tklib_qcheck.elb**, rather than renaming it to **tklib.elb**. Renaming it will cause problems with Script, Scridl, Proto, and ReportWriter, in addition to triggering an automatic re-install due to the now missing file (**tklib_qcheck.elb**).
  - You will experience problems if you build routines that use the ReportWriter External Subroutine Interface with **tklib_qcheck.elb**.

  (9.5.3)

- As of version 9.5.3, the **lstprv.def** and **gblctl.def** files are no longer distributed. (Changes to these files always required your application to be recompiled.) This may break your code. You must remove includes to these files and replace them with calls to functions that provide the same information. For instance, if your code references **glst_scrid** (which was previously available from **lstprv.def**), instead call L_STATUS with D_LCTRID to get that ID. (9.5.3)

- (.NET) If you are using UI Toolkit with Synergy .NET (i.e., if you use **Synergex.SynergyDE.tklib.dll**), you must recompile all of your Synergy .NET code. This is because Toolkit globals changed, and when this happens, all projects that reference the assembly containing them must be recompiled. To force this recompile, in 9.5.3 the version of **Synergex.SynergyDE.tklib.dll** has been changed. (9.5.3)
- (Windows) Windows 7 and all 64-bit Windows platforms now prevent exceptions from being thrown across the kernel (**user32.dll**) boundary. This means that a Windows callback—for example, a close method invoked by clicking the close box (x) for a container—cannot perform a STOP chain or throw an exception that could be caught in a routine on the other side of the callback without causing random hangs and crashes. We have updated the Synergy runtime to generate an $ERR_NOCHAIN error if a Windows callback attempts to perform a STOP chain. (We cannot, however, prevent the callback from throwing an exception that could be caught in a routine on the other side.) (9.3.1b)
- If you include the **lstprv.def** file or the **gblctl.def** file, you must recompile and relink. And if you use the **g_werr** record in **gblctl.def**, you must first replace **g_werr** with the documented method for obtaining this information. (See "Trapping errors" in the "Synergy Windowing API" chapter of the *Synergy Language Reference Manual* for information.) (9.3)
- We changed the type of **gs_dec** in **fldinf.def** from a **d1** to an **i1**. If you have code that includes **fldinf.def** and accesses that member, you must recompile that code. (9.3)
- (Windows) We changed the way renditions work for check boxes so that field-level rendition settings now apply to the prompt portion of a check box when the new D_INPPRCRND state for E_STATE is set to D_ON. Previously, Input Field Display renditions applied to prompts for check boxes in this case. This may change the appearance of existing applications. See related note in "Input processing" on page 3-11 for more information. (9.3)
- (Windows) We updated %U_PRINTQUERY to use a more up-to-date API for retrieving printer information. This modification may change behavior. %U_PRINTQUERY may return more printers than it did in previous versions, and the actual driver name registered on Windows (rather than "winspool") will be returned in the "driver" element of *info_record*. (9.3)
- (Windows) We changed the way E_SECT justifies centered and right-justified text in the information line and footer. This may change the appearance of existing applications. See related entry "Environment processing" on page 3-11. (9.3)

- (Windows) We updated UI Toolkit for 32-bit Synergy/DE to use the new implementation of the ActiveX list control, which was introduced in 64-bit Synergy/DE in version 9.1.5a. This new implementation is identical to the ActiveX list control previously used for 32-bit Synergy/DE, with a few exceptions:
  - Text assigned to headers and footers is trimmed, and if a header or footer is divided into columns, the text for each column is trimmed. If the header or footer is non-static, text is then centered in the column by default. (If they're static, text defaults to left-justified, as it did with the old ActiveX list control.)
  - There's an additional published property, Border, which is a true/false property that determines whether the border of the control is visible. By default, this is set to true (1), which makes the border visible.
  - When you click in the blank area to the right of list columns (but to the left of the vertical scroll bar), Toolkit reports the column number to the click method as 0, rather than the number of the right-most column (which is what Toolkit reports for the previous implementation of the 32-bit ActiveX list control).

  Also note that the DTK_NEWGRID environment variable has been retired. (9.3)
- (Windows) To solve a problem for lists in composite windows, we added OK and Cancel buttons to the default Find window for L_SELECT, and we set OK as the default button. If D_RETURNBTN is not enabled, an additional ENTER is now required to activate OK for the default Find window. You can get the previous behavior by creating your own Find window without buttons and passing it to L_SELECT as the parameter. (9.1.5b)
- (Windows) We enabled support for ActiveX lists on 64-bit Windows systems by implementing a new version of the FarPoint Spread control. This new implementation is identical to the ActiveX list control on 32-bit Windows with a few exceptions (which became standard behavior for both 32-bit and 64-bit ActiveX lists in 9.3.1). See the 9.3 entry on the new ActiveX list control above for information on these differences. (9.1.5a)
- We updated composite window processing so that unplaced child windows are no longer included in the tabbing order or the C_NEXT/C_PREV order, and buttons on an unplaced child window are no longer included in the determination of the default button or of ALT+*key* button activation. Previously, if you called C_PROCESS for a composite window that had no placed or enabled child windows or buttons, C_PROCESS would wait for input because it included the unplaced child in the tabbing order. Now C_PROCESS returns immediately. (9.1.5)
- In previous versions, the MB_ routines did not enforce the maximum number of entry lists, which is 9, or the maximum number of menu column rows, which is 20. Exceeding either of these causes unpredictable results for subsequent calls on the column, so we corrected Toolkit to enforce these as a column is constructed. Toolkit now returns a fatal error ("Too many rows" or "Too many lists") when a column exceeds one of these maximums. (9.1.5)

- As of version 9, we strongly prototype all Synergex-supplied functions and subroutines. This strict adherence to the documented syntax will highlight places in code where the previous compiler failed to detect
  - if the data type of a passed argument is different than the data type defined for the argument (for example, where a decimal variable is passed as an argument that is defined for the routine as alpha).
  - if too many arguments are passed to a routine. For example, the IB_END section in previous versions of the *UI Toolkit Reference Manual* documented six arguments for IB_END even though it recognized only four. (The *field_id* and *selection_id* arguments have long been ignored.) Previous versions of the compiler allowed you to pass six arguments to IB_END, but with version 9, the compiler generates an error indicating that you've passed too many arguments.

  See the *Synergy/DE Quick Migration Guide* for details. If you overload any Synergex-supplied function or subroutine, it must now exactly match the prototype of the Synergex-supplied routine in return type, number, direction, and type of parameters. (9.1)

- As of version 9, we undefine TOOLS_INIT in **tools.def**. If you undefine TOOLS_INIT in your code, you will need to remove this statement to compile with version 9. If you want to be able to compile with version 8.3 or earlier as well, enclose the .undefine in an .ifdef statement that tests for the TOOLS_INIT identifier:

  ```
  .ifdef TOOLS_INIT
  .undefine TOOLS_INIT
  .endc
  ```

  (9.1)

- You must recompile and relink if you include the **lstprv.def** file or the **gblctl.def** file. Note that we plan to remove these files from the distribution. If you have code that includes one of these files, we recommend that you remove such includes and replace them with calls to the functions that provide the same information. For instance, if your code references **glst_scrid**, which is defined in **lstprv.def**, instead call L_STATUS with D_LCTRID to get that ID. (9.1)

New features

ActiveX Processing

- We moved the documentation for AX_TIMEOUT from the *Synergy Language Reference Manual* to the "ActiveX Routines" chapter of the *UI Toolkit Reference Manual*. (9.5)
- (Windows) We added definitions for SS_ALLCOLS, SS_ALLROWS, and SS_HEADER to **grid.def**, a file that defines constants used by the FarPoint Spread control. (9.3)

Button processing

- (Windows) We updated Toolkit so that you can now add and modify ToolTips for buttons on windows and lists. To support this, we added a tenth argument, *tooltip*, to the DSB_ADD and DSB_MODIFY subfunctions of B_BUTTON and L_BUTTON. And to retrieve ToolTip text and length for buttons on windows and lists, we added a new subfunction, DSB_TOOLTIP, to %B_INFO. Note that a ToolTip can be added to a window or list button only if the window or list was created with Synergy/DE version 9.3. (9.3)
- (Windows, UNIX) We added a new function, %B_INFO, that returns information about buttons on a window. (9.1.3)
- (Windows) We added "*NONE*" as an option for the *button_name* argument for B_BUTTONSET. If you pass *button_name* as "*NONE*", Toolkit clears any explicitly set button default for the window. (9.1)

Composite window processing

- (Windows) We improved tabbing for composite windows. You can now instruct Toolkit to tab through buttons on any window or list, even if that window or list is on a tab set. (Previously, Toolkit supported tabbing through buttons on a child input window, a child input list, an input window in a tab set, and the composite window container, but nothing else.) As part of this enhancement,
  - we added a new utility routine, U_CTRCONTEXT, which is designed for use in methods for child windows that aren't input windows, container windows for lists, tab set windows, or composite container windows. U_CTRCONTEXT sets focus for a child window to either the window itself, a button on the window, or, if the window is part of a tab set, a button on the tab set. Like similar routines, such as I_CTRCONTEXT and L_CTRCONTEXT, U_CTRCONTEXT uses a reason code to determine where to establish focus.
  - we updated the default child processing methods C_METHAX, C_METHSEL, and C_METHTXT to tab through buttons on a child window by calling U_CTRCONTEXT. See **ctrmeths.dbl**.
  - we updated the DTS_CTRCONTEXT subfunction for %TS_TABSET to invoke U_CTRCONTEXT and L_CTRCONTEXT.

  (9.1.3)

- (Windows) We updated composite window processing so that it now establishes focus on a child window when the child window is right-clicked. (9.1.3)
- (Windows) In versions 9.1.1 and 9.1.1a, when tabbing between buttons on a container window, the TAB key became ineffective if one of the buttons was disabled. We corrected this by making TAB and SHIFT+TAB skip over disabled buttons on a container, just as they do with input windows. (9.1.1b)
- (Windows) In version 9.1.1, buttons for lists were not included in the tabbing order for a composite window. We corrected this and added a new routine, L_CTRCONTEXT, that enables you to establish focus on the list or a button for the list, depending on how the list was activated. Additionally, we updated C_METHLST to call L_CTRCONTEXT. (9.1.1a)
- UI Toolkit now enables you to create composite windows. A composite window consists of a parent window and child windows, and it functions and appears to the user as a single window. For example, a composite window could have input fields at the top for header information, a table in the middle for line item entry, a multi-line field that supports rich text editing through an ActiveX control, and buttons.

  To implement this feature, we added the %C_CONTAINER function, which has the following subfunctions:

  - DC_ADD, which enables you to add a child window or list to a composite window
  - DC_BTNFOCUS, which enables you to determine whether a button on a composite window has focus, to change focus to a specified button, and to retrieve the tabbing index of the button that had focus prior to the DC_BTNFOCUS call
  - DC_CHILD, which enables you to determine whether a child exists for a given tabbing index and, if it does, retrieve the type and ID of the child
  - DC_CHILDINDEX, which enables you to determine whether a window or list is a child and, if it is, retrieve the tabbing index for the child
  - DC_CONTAINER, which enables you to determine whether a window or list is a child and, if it is, retrieve the container window ID for the child
  - DC_CONTEXT, which enables you to use a tabbing index to change the active child for a composite window, and that indicates which child is active by returning the tabbing index for that child
  - DC_CREATE, which enables you to create a composite container window
  - DC_CTRCONTEXT, which enables you to set the active child or button based on a code that indicates the reason the container received focus
  - DC_ENABLE, which enables you to determine if a child window or list is enabled or disabled and/or enable or disable the child window or list
  - DC_EVENT, which enables you to register an extension to the standard event methods for a child window or list
  - DC_GETEVENT, which enables you to retrieve the address for an extension registered for a child window or list event method

- DC_NUMCHILD, which enables you to determine how many children (both windows and lists) are associated with a composite container window
- DC_REMOVE, which enables you to remove a child window or list from a composite window

We also added the C_PROCESS subroutine to process composite windows. This subroutine calls either the following new child processing methods or your own custom methods. Note that the UI Toolkit distribution includes **ctrmeths.dbl**, a file that contains the code for these methods. You can use this file as a reference when writing a custom method.

- C_METHAX for child ActiveX windows
- C_METHCTR for child composite windows
- C_METHINP for child input windows
- C_METHLST for child lists
- C_METHNOP, which prevents a child from gaining focus
- C_METHSEL for child selection lists
- C_METHTS for child tab sets
- C_METHTXT for child text windows

We added the following new reserved menu entries, which are recognized by C_PROCESS:

- C_FIRST, which makes the first child (window or list) active (or makes the first button active if there is no child)
- C_LAST, which makes the last button active (or makes the last child active if there is no enabled button)
- C_NEXT, which makes the next child (window or list) active (or if the last child is currently active, makes the first button on the composite container window active)
- C_PREV, which makes the previous child window, child list, or button on the container active
- C_RESET, which checks for any programmatic context change (for example one made with DC_CONTEXT or DC_BTNFOCUS) and then sets the context in a composite window to the active child window, child list, or button
- C_SET*nnnn*, which makes the child window or list with the tabbing index *nnnn* active

We also added the following:

- The I_CTRCONTEXT subroutine to establish context for input windows based on the reason the input window was activated (which is useful in child methods for input windows, where most often input context should initially reflect the reason the input window received focus)
- The L_NEXT subroutine to specify which list item will be selected when a list is next processed

- The %L_SELSTYLE function to retrieve and/or change the style for the selected item in a list (which is useful in child list processing methods, where it can be difficult for users to tell when the list has focus unless you change this style)
- The DTS_CTRCONTEXT subfunction for %TS_TABSET to set context for a window that has been activated on a tab (which is useful in child tab set processing methods, where it is often best for focus within the window to initially reflect the reason the window itself received focus)
- The D_LGETEVENTS subfunction for L_STATUS to determine if a method set has been registered for a list and, if one has, return the method set ID
- The D_GETEVENT subfunction for %U_WNDEVENTS to return the address of a method routine for a given event
- The D_GETEVENTS subfunction for %U_WNDEVENTS to determine whether a method set has been registered for a Toolkit window and, if it has, return the method set ID

And we made the following changes:

- We updated I_INPUT so that pressing TAB or calling I_NEXTCTL from the last button in a composite window (or last field, if there are no buttons) signals the new C_NEXT reserved menu entry to activate the next child. Additionally, pressing SHIFT+TAB or calling I_PREVCTL from the first field (or first button if there are no fields) signals the new C_PREV reserved menu entry to activate the previous child in a composite window.
- We added "*LASTCTL*", a new field specification keyword for I_NEXT. This keyword instructs I_NEXT to process the last enabled control (field or button) for the input set or window.
- We changed .SET and IB_SET so that they can now be used to create empty input sets, which enables you to create input windows that have no fields. This is useful if you want to create a panel of buttons that can appear anywhere on a composite window.

For Windows, we made the following additional changes:

- We added a new data member, **inp_navevent**, to the **inputinfo** structure to indicate which event triggered movement to or from a field.
- We changed the way tabbing works for lists that are part of a composite window. When the last field in an item is reached, TAB activates the next child. When on the first field for an item, SHIFT+TAB activates the previous child.
- We changed the way tabbing works for ActiveX controls in composite windows. As long as TAB is not mapped as an accelerator by the control, TAB signals C_NEXT. If SHIFT+TAB is not mapped, it signals C_PREV.
- We changed the way tabbing works for text processing when in a composite window. If TAB is not a shortcut on a placed menu column, TAB causes T_EDIT to return after signaling C_NEXT. And if SHIFT+TAB is not a shortcut on a placed menu entry, it causes T_EDIT to return after signaling C_PREV.

- We changed the way tabbing works for selection windows that are part of a composite window. TAB causes S_SELECT to return and signal C_NEXT, and SHIFT+TAB causes S_SELECT to return and signal C_PREV.
- We created new rules for establishing the default button in a composite window and for determining which button will be activated when an ALT+*key* key combination is pressed.
- We changed the rules for automatically enabling and disabling buttons. For composite windows, rather than enabling or disabling a button when its window or list is activated or deactivated, all buttons are automatically enabled when the composite window is processed and disabled when processing for the composite window is finished. The exceptions are
  - buttons that are explicitly disabled (with %B_DISABLE) and buttons on child windows or lists that are disabled (with DC_ENABLE), which remain disabled when the composite window is processed.
  - buttons that are explicitly enabled with %B_ENABLE without passing the *input_only* flag as true, which remain enabled when composite window processing is finished.
- All windows and lists added to a composite window now have their events method (%UWNDEVENTS_METHOD) set replaced with one that better handles mouse events in composite windows.

(9.1)

Debugger

- We improved Toolkit debugging when using the DTKDBG environment variable. When DTKDBG is set, the debugger now
  - checks for any %M_SIGNAL call that supersedes a pending menu entry. When the debugger encounters such a call, it displays a message box that indicates which menu entry the call is signaling and which entry is being superseded. It also asks if you want to continue. If you click Yes, the operation proceeds. If you choose not to continue, the debugger calls U_ABORT, which generates a fatal Toolkit error.
  - displays a message box if a USTART_METHOD cannot be invoked. The message box asks if you want to continue. If you click Yes, the program proceeds with non-DTKDBG behavior—i.e., it ignores the method. (In previous versions, this happened regardless of DTKDBG.) If you click No, the debugger calls U_ABORT, which generates a fatal Toolkit error.

(9.5.3)

Environment processing

- We updated %E_INFO so that it returns 0 whenever it is called with D_ENV_LEVEL while Toolkit is inactive. In previous versions, if you called %E_INFO with D_ENV_LEVEL, it would return 0 if Toolkit had never been initialized (with U_START) in the current session, but it would return 1 if it had been initialized and subsequently terminated (with U_FINISH). Now it returns 0 in both cases. (9.3.1a)
- We added a new window type, D_WTYPE_LISTCLASS, to the types that the D_WINDOW_TYPE subfunction for %E_INFO can return. D_WTYPE_LISTCLASS is returned if the window specified for D_WINDOW_TYPE is a list class. (9.3)
- (Windows) We changed the way E_SECT justifies centered and right-justified text in the information line and footer. If the D_VLINE option is enabled and justified text contains vertical bars, justification works as it has since Synergy/DE 9.1: text appears justified as specified, but the left section border is always next to the left side of the text. For example:

  ```
  |                    |centered |with |bars |                    |
  ```

  Otherwise, native Windows justification is used, and the information line or footer is generally evenly divided. For example:

  ```
  |left-justified text  |  centered text  |  right-justified text|
  ```

  Note that this may change the appearance of existing applications. (9.3)
- We updated **tools.def** with two new defined literals for the D_WINDOW_TYPE subfunction of %E_INFO. This subfunction now returns D_WTYPE_SELECT for a selection window and D_WTYPE_CONTAINER for a composite container window. (9.1.1b)
- We added a new subfunction, D_CHANNEL_MAX, to %E_INFO. This subfunction returns the highest channel number that Toolkit can use (which is controlled by U_START). If you have used **chninf.def** to find the highest channel number for past versions (we quit distributing this file as of 8.3.1), update your code to use D_CHANNEL_MAX instead. (9.1)

Input processing

- We updated I_TXTPOS with two new options that enable you to get either the last position occupied by the cursor or the next projected position for input in a multi-line text field. If the third argument is D_GETLAST, the next two arguments are returned with the row and column last occupied by the cursor (or 0,0 if no row or column had been established). If the third argument is D_GETNEXT, the next two arguments are returned with the next row and column that will be used for input (or 0,0 if none has been specified). (9.5.3)
- We added the ability to access the members of a Repository “Struct” type field. (9.5.1b)
- (Windows) We added the *position* and *tooltip* arguments (previously implemented for B_BUTTON) to IB_BUTTON. (9.3.1a)

- We updated Toolkit so that on Windows you can now apply the Input Field Processing or Read-Only Input Field processing renditions to the field that has focus in an input window. We also updated Toolkit on UNIX and OpenVMS so that you can choose to prevent these renditions from being applied to the field with focus. To do this, we made the following changes:
  - We added two new state codes to E_STATE: D_INPPRCRND and D_INPRNDOVER. When set to D_ON, D_INPPRCRND instructs Toolkit to apply the Input Field Processing or Read-Only Input Field Processing renditions to the field that has focus during input processing. (When set to D_OFF, it prevents them from being applied.) And D_INPRNDOVER instructs Toolkit to override field-level color and attribute settings with the Input Field Processing or Read-Only Input Field Processing renditions.
  - Field-level rendition settings now apply to prompts for check boxes on Windows when the new D_INPPRCRND state for E_STATE is set to D_ON. Previously, Toolkit used the renditions for the portion of the window that the first character of the prompt occupied. Now it uses these renditions only if there are no field-level settings. Note that this may change the appearance of existing applications on Windows.

  (9.1.1b)
- On Windows, we added support for the E_MODE reserved menu entry (which toggles insert/overstrike mode) to I_INPUT, I_INPFLD, and T_EDIT. Additionally, the fourth argument for T_EDIT is now supported on all platforms. (9.1)
- (Windows) In previous versions, when a field received focus because of a mouse click, the contents of the field were selected. To more closely adhere to Windows standards, we have suppressed this behavior so that the cursor is now placed where the click took place. You can still highlight the contents of a field by either double-clicking the field or using TAB or ENTER to move to the field. (9.1.1b)
- We updated Toolkit to generate a warning when DTK_BOUNDS is set and a multi-line text field will be truncated because it has too many rows to fit in its window. (See "Script processing" on page 3-15 for related entry.) (9.1)

List Processing

- We added two new subfunctions to L_STATUS: D_LHDRID returns the ID of the window used internally by Toolkit for a list's header, and D_LFTRID returns the ID of the window used internally for a list's footer. These subfunctions should be used in place of any references in your code to **glst_hdrid** or **glst_ftrid** (from **lstprv.def**). (9.5.3)
- (Windows) We now distribute version 8.0.0.9 of the FarPoint Spread ActiveX control, which is the control we use for ActiveX Toolkit lists. (9.3.1a)
- The prototype for L_SELECT has been updated to allow the second argument to both pass and receive a request flag. (9.3.1a)

- We added a new function, %L_CLASSINFO, to retrieve information about a list class. (9.3)
- (Windows) We updated UI Toolkit for 32-bit Synergy/DE to use the new implementation of the ActiveX list control, which was introduced in 64-bit Synergy/DE in version 9.1.5a. (It was also used for 32-bit Synergy/DE in 9.1.5a and 9.1.5b if you set the DTK_NEWGRID environment variable.) With this new implementation, you may see slightly improved performance and reduced flicker in some cases. Note the following:
  - The behavior and look of the new implementation is identical to that of the old implementation, with the exceptions listed in the related note in "Features and fixes that may break your existing code" on page 3-2. (Note that the change to behavior when you click in the blank area to the right of list columns may break your code.)
  - The DTK_NEWGRID environment variable has been retired. Toolkit no longer recognizes it.

  (9.3)
- (Windows) We now distribute version 8.0.0.7 of the FarPoint Spread ActiveX control, which is the control used for ActiveX Toolkit lists. (9.3)
- (Windows) We added ArrowKeyAction, a published property for the ActiveX list control. This property enables you to control how UP ARROW, DOWN ARROW, and other navigation keys work. For example, when set to 0 (the default), UP ARROW and DOWN ARROW move highlighting up and down one row at a time. When set to 1, UP ARROW and DOWN ARROW move highlighting up and down one item at a time. (An item can have more than one row.) (9.3)
- (Windows) We added NonGridBackColor, a published property for the ActiveX list control. This property enables you to set the color of the area outside of the bounds of data columns but inside the ActiveX control. (9.3)
- (Windows) We updated the ActiveX list control so that header and footer lines can now have as many as 32,767 characters. (Previously header and footer lines for ActiveX lists were limited to the width of the input window associated with the list.) (9.3)
- (Windows) For 9.1.5b, we upgraded to the following versions of the FarPoint Spread ActiveX control, which is the control we use for ActiveX Toolkit lists:
  - We upgraded to FarPoint Spread 8.0.0.6 for the version of the ActiveX Toolkit list control that runs on 64-bit Windows operating systems and that runs on 32-bit Windows operating systems when the DTK_NEWGRID environment variable is set.
  - We upgraded to FarPoint Spread 7.0.0.51 for the default 32-bit version of the ActiveX Toolkit list control (i.e., the version used when DTK_NEWGRID is not set).

  (9.1.5b)

- (Windows) We added OK and Cancel buttons to the default Find window for L_SELECT, and we set OK as the default button. This solves a problem with previous versions where it was impossible for the user to exit the default Find window normally when the D_RETURNBTN state was enabled and the list was in a composite window that had buttons. Now pressing ENTER activates the OK button in this situation. (9.1.5b)
- (Windows) For 9.1.5a, we enabled support for the ActiveX list control on 64-bit Windows systems by implementing a new version of the FarPoint Spread control. The ActiveX list control on 64-bit Windows is identical to the old ActiveX list control on 32-bit Windows with the exceptions listed in the related note in "Features and fixes that may break your existing code" on page 3-2. (Note that the change to behavior when you click in the blank area to the right of list columns may break your code.) With 9.1.5a, you can try the new implementation on 32-bit systems by setting the DTK_NEWGRID environment variable to any value, and you can return to the old implementation by setting DTK_NEWGRID to nothing. (9.1.5a)

.NET Processing

- (Windows) We added the ability to embed the following in Toolkit windows: .NET forms, .NET controls, and Windows Presentation Foundation (WPF) elements (i.e., System.Windows.Forms.Form objects, System.Windows.Forms.Control objects, and System.Windows.UIElement objects). To implement this feature, we added the following:
  - The %DOTNET_TKWIN function, which enables you to embed a .NET form, .NET control, or WPF element in a Toolkit window.
  - The DOTNET_TKADDCTL subroutine, which enables you to add a .NET form, .NET control, or WPF element to a previously embedded .NET form.
  - The DOTNET_TKINPUT subroutine, which enables you to process an embedded .NET form.
  - The DOTNET_TKFORM subroutine, which enables you to retrieve a .NET form object from a Toolkit container.
  - The C_METHNET composite window processing method, which calls the new DOTNET_TKINPUT subroutine.
  - The D_WTYPE_DOTNET value for the %E_INFO subfunction D_WINDOW_TYPE.

  Note that to run a Toolkit program with this functionality, you need .NET Framework (2.0 or later for .NET forms and controls, or 3.0 or later for WPF elements). You also need Windows XP or 2003 or later. (9.1.5)

Proto

We limited the number of times Proto attempts to save a window to a library when a locked window library record prevents the save. Proto now retries up to 10 times with two-second delays between retries. (Previously Proto would retry indefinitely.) (9.1.5)

Script processing

- We updated %SCR_PROCESS and the Script compiler to retry a save when locked window library records prevent them from saving. %SCR_PROCESS and Script now retry up to 10 times with two-second delays between retries, and they use %SCR_PROCESS's error reporting mechanism to report errors caused by failed attempts to save to window libraries. (Previously both returned an error via U_ABORT, causing the program to quit.) (9.3)
- To better support concurrently-running instances of Script, we updated Script to ignore any errors it encounters when attempting to delete the temporary file it creates for storing error information. (9.3)
- We updated Toolkit to support the repository specification of input length and display length for an arrayed field when accessing an individual member of the array. When using the array as a multiline text field, however, any repository specification of input or display length is ignored. (9.3)
- We updated Script to generate a warning if a multi-line text field will be truncated because it has too many rows to fit in its window. (See "Input processing" on page 3-11 for a related entry.) (9.1)

Tab set processing

- (Windows) We added the *position* and *tooltip* arguments (previously implemented for B_BUTTON) to the DTS_BUTTON subfunction of %TS_TABSET. (9.3.1a)
- (Windows) We added the following subfunctions to %U_WINMETRICS to enable you to retrieve information about the size of tabs and the position of tab sets.

D_TABHEIGHT	Returns the height (y-axis measurement) of a tab.
D_TABWIDTH	Returns the width (x-axis measurement) of a tab.
D_TABXOFFSET	Returns the distance from the left side of the client area of the tab set container window to the left side of windows on tabs.
D_TABYOFFSET	Returns the distance from the top of the client area of the tab set container window to the top of windows on tabs.

  (9.3)
- We added DTS_TABSET, a new subfunction for %TS_TABSET, which returns the ID of a tab set that contains a specified window or list. (If the window or list is not contained in a tab set it returns zero.) If the window or list is part of a composite window that's on the tab set, DTS_TABSET will also return the ID of the composite window if you pass an optional argument. (9.3)
- (Windows) We added the ability to change the text of a tab by using W_BRDR(WB_TITLE) for a window or L_SECT(D_TITLE) for a list. Note, however, that on UNIX and OpenVMS, you still need to remove and re-add the window in order to update the tab title. (9.1.1b)

Toolbar processing

- (Windows) We added a new routine, %TB_INFO, which enables you to retrieve information about toolbars and toolbar buttons. (9.3)

Utility routines

- We added U_EDIT, a new routine that, like T_EDIT, enables users to enter and edit text in a window. Unlike T_EDIT, however, U_EDIT automatically wraps text at the end of a line, preserves hard breaks, and enables you to specify the type of line break (hard or soft) that will be created when the user presses ENTER. With U_EDIT, you can also supply initial text to be displayed when U_EDIT starts, and U_EDIT returns the final state of text in a way that preserves hard returns. (9.5.3)
- We added a new, optional argument to U_VERSION to return the version number with space allocated in anticipation of version 10. We have also made the original first argument to U_VERSION optional so you don't have to obtain both formats to get the new one. (9.5.3)
- (UNIX, OpenVMS) We improved U_ABORT so that if possible it wraps (rather than truncates) lines of text that are longer than the 70-character limit for an error message. (9.5.3)
- We added a new function, %U_ENUMWNDS, that returns the count of and optionally the names of all windows in a window library. (9.3)
- We added a new function, %U_WNDTYPE, that determines the type of any loaded window, including a window loaded with U_LDWND. (The D_WINDOW_TYPE subfunction for %E_INFO also returns this information, but only for windows loaded with type-specific routines—e.g., I_LDINP for an input window.) (9.3)
- We updated %U_GETWNDERR to return text for returned Toolkit errors. (Previously, %U_GETWNDERR returned text only for the last returned window error.) So, for example, if you pass the *error* argument in a call to L_CLASS (causing Toolkit to return a Toolkit error, rather than sending it on to U_ABORT), %U_GETWNDERR will return Toolkit error text. (9.3)
- (Windows) We updated %U_PRINTQUERY to use a more up-to-date API for retrieving printer information. This means that %U_PRINTQUERY may return more printers than it did in previous versions. Additionally, the actual driver name registered on Windows (rather than "winspool") will be returned in the "driver" element of *info_record*. Note that this modification may change behavior. (9.3)
- (UNIX, OpenVMS) When using **dbs**, calls to U_MESSAGE, %U_MSGBOX, and U_ABORT now output messages to the standard output channel. (9.3)
- (Windows) We improved %U_ICON so that if an icon file (**.ico**) includes a small icon, Toolkit will use the small icon when it's needed. Previously, Toolkit scaled down the large version of the icon instead. (9.3)

- (Windows) We added a new routine, %U_HTMLHELP, that enables Toolkit programs to invoke Windows HTML Help (**.chm** files). (9.1)
- (Windows) We enhanced the %U_WINCOLOR subfunction D_CHOOSECOLOR and the %U_WNDFONT subfunction D_CHOOSEFONT to display help from HTML Help files (**.chm**) as well as WinHelp files (**.hlp**). If the extension of the file passed as the *help_file* argument is **.chm**, the context ID and help file information is now passed on to the HTML Help API command HH_HELP_CONTEXT. (9.1)

Miscellaneous

- (Windows) We added support for cell-based functions to the UI Toolkit library for .NET (**tklib.dll**). This gives you the ability to move a cell-based application to .NET without having to recode the UI elements. And if you already have a Windows application, you can move back-office programs to .NET. Note that your UI Toolkit applications will still function as cell-based applications—for example, there will be no mouse, no native menu controls, nor the ability to maximize the application window. But this new functionality does give you the ability to get your applications running quickly under .NET, and then you can migrate select screens to Windows Forms or WPF. (9.5.3)
- (Windows) We extended the meaning of DTK_BOUNDS on Windows platforms. If this is set to any value, Toolkit now also checks to make sure W_PROC(WP_RESIZE) is not used to resize one of Toolkit's reserved windows (header, footer, or information line) or an input window for a list. If W_PROC is used in this way, Toolkit calls U_ABORT to report this along with the window ID and the old and new sizes. (9.5.3)
- (Windows) **Tklib.elb** is now built with **-qstrict**, and there is a new distributed file, **tklib_qcheck.elb**, that is the UI Toolkit library built with **-qcheck**. (9.5.3)
- (.NET) We added support for scaling the font used in a .NET application that makes use of low-level windows or UI Toolkit. If SYN_RESIZE_SCALE is set to 1 in the environment, you may resize, maximize, or restore the form used for window/Toolkit display, and the font will be scaled accordingly. (9.5.3)
- We added **g_beep**, a configuration field in **tkctl.def** that enables you to turn off the terminal bell (on all platforms) and MessageBeep (on Windows). (Note, however, that **g_beep** cannot turn off Windows beeps or other sounds associated with message box icons.) We also added an environment variable, DTK_BEEP, that sets the initial value of **g_beep**. (9.3)
- We increased the maximum precision supported by Toolkit to 28, and we increased the maximum size of a decimal field to 28. To support this, we changed the **gs_dec** member of **fldinf.def** from a **d1** to an **i1**, so if you have any code that includes **fldinf.def** and accesses **gs_dec**, you will need to recompile that code. (9.3)
- (Windows) We updated the **SynergyQuoteWizard.ocx** example (in toolkit\examples\spc98\controls) so that it now works with Windows Vista. (9.1.3)

- (Windows) The ActiveX controls in the toolkit\examples\controls folder are now compatible with Windows Vista. (We rebuilt them using VB6.) There is one exception, however: we were unable to rebuild **SplashScreen.ocx**, so it is not available on Vista. (9.1.1b)
- You can now pass any argument type (**a**, **n**, or **object**) as an *a_methoddata* argument for any routine that includes these arguments (I_INPUT, I_INPFLD, L_PROCESS, etc.). (9.1)
- (Windows) We changed the status bar displayed for the footer and information line to use a Windows common control, so these now reflect Windows themes selected for users' systems. (9.1)
- (Windows) We enhanced the Toolkit debugger (DTKDBG) so that you can now access more than 25 lines of text by scrolling. By default, 300 lines are available, but you can change this by changing the DBG_BUFFER environment variable setting in **synergy.ini** or **synuser.ini**. (9.1)

Version 8

This section briefly describes new features in UI Toolkit version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to UI Toolkit version 8.

- (Windows) In previous versions, if a menu entry was signaled from a display, change, or leave method during I_INPUT processing, and a user action (mouse click, ENTER, TAB, SHIFT+TAB, etc.) moved input context to a subsequent field or button as the method was being processed, Toolkit incorrectly applied the menu entry processing to the subsequent field or button rather than to the field whose method called %M_SIGNAL. We corrected this so that context now remains on the field whose method signaled the menu entry. This matches Toolkit behavior on UNIX and OpenVMS, but may break your code. In case you have code that depends on the previous behavior, we added a global variable (**g_retaincontext**, which is defined in **tkctl.def**) and an environment variable (RETAIN_CONTEXT_CHANGE_ON_SIGNAL) that enable you to restore the previous behavior. (8.3)
- (Windows) In previous versions, if a menu entry was signaled from a method that moved input context to a subsequent field, UI Toolkit incorrectly applied the menu entry processing to the subsequent field rather than the field whose method called %M_SIGNAL. We corrected this so that context now remains on the field whose method signaled the menu entry, which may break your code. See "Input subroutines" on page 3-22 for more information. (8.3)
- (Windows) We improved Synergy/DE's color support to accommodate Windows system colors. This includes changes to the return values of U_REND and the WI_PALET subfunction to W_INFO that may break your code, and it includes changes to the default palette that may change the appearance of Toolkit applications and applications that use the Synergy Language windowing subroutines. See "Miscellaneous" on page 3-30 for more information. (8.3)
- We no longer distribute **tklib.olb** (or the related files **chninf.dbl**, **chninf.def**, and **fkey.dbl**). All of the user-replaceable routines in this **.olb** have been superseded with methods that can be registered by U_START or E_METHOD. This may break your code if your application relies on replacing routines in **tklib.olb** (for example, USR_UTILS or USR_START). On Windows and UNIX, use **tklib.elb** instead; on OpenVMS, use **tklib_sh.exe**. (8.3)

- Toolkit now enables you to create fields that reference specific elements in repository arrayed fields. To support this we updated Toolkit routines that accept field specifications so that they now recognize dimension indices (which are enclosed in square brackets) as part of a field specification. This modification may break your code because Toolkit no longer recognizes square brackets as field delimiters in field lists passed to I_DISABLE, I_DISPLAY, I_ENABLE, I_INIT, or I_SETDEL. See "Input subroutines" on page 3-22 for more information. (8.3)
- We removed the vestigial routine U_SELDFLT, which was renamed S_SELDFLT in Toolkit version 3.0. If you have code that calls U_SELDFLT, you must update it to call S_SELDFLT instead. (8.3)
- We removed the user-replaceable list subroutines (the subroutines that begin with USR_L). If your Toolkit program includes a customized version of one of these routines, Toolkit will now ignore it. For example, if your code includes a customized version of USR_LDATA, Toolkit will ignore that version and will instead use default L_DATA processing. Additionally, we no longer distribute **lstusr.dbl**, **lstprv.dbl**, or **lstbuf.def**. (8.3)
- We added a new option (D_NOLOAD) to MB_COLUMN that enables you to create a menu column at runtime without loading it. This modification may break your code. See "Menu subroutines" on page 3-28 for more information. (8.3)
- We added the ability to cancel a pending I_FORCE by passing a null string ('') for the *new_data* argument to I_FORCE. This modification may break your code. See "Input subroutines" on page 3-22 for more information. (8.3)
- (Windows) In previous versions of the ActiveX list control, the *a_row* argument passed to list item arrive and leave methods (LLARIVE_METHOD/LLEAVE_METHOD) was incorrect. We have corrected this value. Note, however, that this modification may break your code if your code compensates for an incorrect value. (8.1.7e)
- (Windows) In previous versions of the ActiveX list control, L_RESTART did not reset individual cell overrides for color and font (set with SetCellColor or SetCellFont). We have corrected this to reset these to the default color and font. (8.1.7)
- If you are including the **gblctl.def** file (which we don't recommend), you must recompile and relink. (8.1, 8.3)
- (Windows) You can now control how a set of radio buttons will be arranged. You can control this by specifying a height for the associated selection window. Note that this may change the appearance of existing windows that have radio buttons because selection window height, which was previously ignored, is now used to control the orientation of the buttons. (8.1)

- (Windows) Synergy/DE now supports the concept of a system-specific initialization file (**synergy.ini**) versus a user-specific initialization file (**synuser.ini**). Settings in the user-specific file override settings in the system-specific file. (For more information, see the "Environment Variables" chapter of *Environment Variables and System Options*.) The **synuser.ini** file is created (and searched for) in the Synergex subdirectory of your local application data directory. To determine where the **synuser.ini** file is located on your particular system, you can run the **synckini** utility. (For information on this utility, see "The synckini Utility" section in the "General Utilities" chapter of *Synergy Language Tools*.)

  To support this change, we changed the following UI Toolkit routines to write to **synuser.ini**. (For more information, see the related entry in "Utility subroutines" on page 3-40.)

  - U_EDITREND
  - U_SAVELOG
  - U_SAVESETTINGS
  - The D_SAVEFONT subfunction to %U_WNDFONT

  This may break your code because we changed the default location of **synergy.ini** from the Windows (or Winnt) directory to the directory specified by DBLDIR. If you depend on settings in **synergy.ini**, we recommend that you move **synergy.ini** (either the distributed version or the one in your Windows directory) to another location and set SFWINIPATH to that directory. (8.1)

- In previous versions, if the number of decimal places specified for a field was greater than nine (the maximum supported), the high-order digits of the value were truncated and the remainder became the precision. (For example, if you specified 12, 2 would be the precision.) We have corrected this. When a number greater than nine is specified in a window script or a call to IB_FIELD, Toolkit now generates an error message: "Decimal places specified (*nn*) not in range 0-9", where *nn* is the number of decimal places specified for the field. (8.1)
- We modified the behavior of IB_FIELD so that D_OFF followed by either D_FLD_PROMPT or D_FLD_PROMPTLEN will cancel either a text prompt or a runtime prompt. (In previous versions, using D_OFF with D_FLD_PROMPT would cancel only a text prompt, and using D_OFF with D_FLD_PROMPTLEN would cancel only a runtime prompt.) This makes the behavior of IB_FIELD consistent with window scripts, where "noprompt" is used to cancel either type of prompt. (8.1)
- In previous versions, if a field had a display length that extended beyond the right side of the window, the display was merely truncated. No error was generated. As a result of the change to support input field view length, we updated Toolkit to generate the error "Field *name* does not fit in the window" and report both the maximum width for the field's position and the specified width. This error is generated in Script processing (on the ".END" statement for the window) or in IB_END when building an input window at runtime. It also applies to any combination of size, input length, display length, view length, and enumerated length that results in an effective view length that is too wide to fit in the window. (8.1)

New features

Button subroutines

- (Windows) Toolkit now supports the following Windows option for buttons with select characters: “Hide underlined letters for keyboard navigation until I press the Alt key” (Windows Control Panel > Display > Appearance > Effects). (8.1.7a)
- (Windows) We reduced the amount of repainting that takes place when buttons change. (8.1.7a)
- We added support for additional image formats and for “transparent” images on toolbar buttons and command buttons. The new image formats are GIF, JPEG, Exif, PNG, and TIFF. To support transparent images, we added two environment variables, SYN_TRANSPARENT_COLOR and SYN_TRNSPARENCY_THRESHOLD. (8.1.7)
- We added subfunctions to the B_BUTTON and L_BUTTON subroutines. These new subfunctions enable you to remove, modify, and reorder buttons after they are added to a window or list. (8.1)

Input subroutines

- (Windows) UI Toolkit now uses the hand style cursor for hyperlinked prompts. (8.3.1c)
- We added a new function, %I_INFO, to enable you to retrieve information about an input window. %I_INFO returns either the number of input fields or the number of input sets for the input window. (8.3.1b)
- In previous 8.3.1 versions, I_DSPFLD would report an error when bounds checking was on if the buffer passed was smaller than the size of the field, even if I_DSPFLD could handle this situation correctly. We changed bounds checking so that you can avoid this. We added a new option, the value 2, which instructs bounds checking to enforce exact argument size checks in all cases, and we changed the behavior of DTK_BOUNDS/**g_dtkbounds** when set to 1. If you set DTK_BOUNDS/**g_dtkbounds** to 1, bounds checking now ignores cases where the referenced data is not the exact correct size, but causes no overrun. We also defined literals in **tools.def** for the two levels of bounds checking: D_BOUNDS_OVERRUN, to test only for overrun conditions, and D_BOUNDS_EXACT for exact argument size checking. (8.3.1b)
- We added a new routine, I_READONLY, that enables you to set or clear the read-only state of multiple fields. In one call you can set the read-only state for all of the fields in an input window, all of the fields in an input set, or up to 81 individually specified fields. (8.3)

- Toolkit now enables you to create fields that reference specific elements in repository arrayed fields. For example, to create a field that references the third field (in the first dimension) of the seventh element (of the second dimension) of an array named "total," you can use

  ```
  .FIELD total[3,7]
  ```

  or

  ```
  xcall ib_field(build_id, "total[3,7]")
  ```

  To support this, we updated

  - the .FIELD and .SET script commands to accept dimension specifications for fields that reference repository array fields.
  - the following routines to accept repository arrayed fields that have dimension specifications: I_DISABLE, I_DISPLAY, I_DSPFLD, I_ENABLE, %I_FLDDIM, I_FLDINF, I_FLDMOD, %I_FLDPREC, I_FLDSEL, %I_FLDSIZE, I_FLDSTRPOS, I_GETFLD, I_INIT, I_INPFLD, I_NEXT, I_PROMPT, I_PUTFLD, I_USER, IB_FIELD, and IB_SET.
  - Composer to parse and create scripts with dimension information for repository arrayed fields.

  Note that this modification may break your code because the following routines (which accept delimited lists of fields) no longer recognize square brackets as field delimiters in field lists: I_DISABLE, I_DISPLAY, I_ENABLE, I_INIT, and I_SETDEL. If you use the left or right square bracket as a field delimiter, replace it with a valid delimiter character (any other non-identifier character). (8.3)

- (Windows) We added a reserved field name (*BUTTON*) to I_NEXT that enables you to move focus to a specific button on an input window. The *BUTTON* reserved field name has one argument, which you can use to pass either a button name or the index for a button. (8.3)
- We improved I_FLDMOD so that you can now use it to modify an enumerated input field's display length, base, and step values. (8.3)
- We added the ability to clear the force buffer and cancel a pending I_FORCE call by passing a null string ('') for the *new_data* argument to I_FORCE. Note that this may break your code because in previous versions passing a null string for *new_data* forced a blank space into the field (the equivalent of ' '). If you depend on the previous behavior, update your code to pass a space instead of a null string. (8.3)

- (Windows) In previous versions, if a menu entry was signaled from a display, change, or leave method during I_INPUT processing, and a user action (mouse click, ENTER, TAB, SHIFT+TAB, etc.) moved input context to a subsequent field or button as the method was being processed, Toolkit incorrectly applied the menu entry processing to the subsequent field or button rather than to the field whose method called %M_SIGNAL. We corrected this so that context now remains on the field whose method signaled the menu entry. This matches Toolkit behavior on UNIX and OpenVMS. This modification may break your code. In case you have code that depends on the previous behavior, however, we added a global variable (**g_retaincontext**, which is defined in **tkctl.def**) and an environment variable (RETAIN_CONTEXT_CHANGE_ON_SIGNAL) that enable you to restore the previous behavior. (8.3)
- (Windows) We reduced CPU usage and Windows message overhead incurred when making calls to I_FLDMOD, I_DSPFLD, I_DISPLAY, or I_INIT. (8.1.5d)
- Toolkit now supports scrolling text fields on Windows. To do this, we added the following:
  - DSP_LENGTH/NODSP_LENGTH, a display length setting for the .FIELD script qualifier.
  - INPUT_LENGTH/NOINPUT_LENGTH, an input length setting for the .FIELD script qualifier.
  - VIEW_LENGTH/NOVIEW_LENGTH, a view length setting for the .FIELD script qualifier.
  - D_FLD_VIEWLEN, a view length setting for I_FLDMOD and IB_FIELD.
  - Display length, Input length, and View length, field properties in Window Designer.

  Now, if you set a field's view size smaller than its display size on Windows, the contents of the field will be scrollable up to the display length. On OpenVMS or UNIX, the field will be truncated to the new length. (8.1)
- We increased the maximum size for an input field from 99 characters to 65,535 characters. (Note, however, that because of limitations on Windows 98/Me the practical limit 32,766 on those platforms. You can still define a field with as many as 65,535 characters, but when you enter data into the field, you'll be limited to 32,766 characters, and when you display data into the field, it will be truncated to 32,766 characters.) (8.1)

- We made the following changes to I_INPFLD:
  - I_INPFLD now correctly supports all field view types (combo box, radio button, checkbox, and multi-line field).
  - (Windows) Arrive and leave methods are now invoked. Unlike I_INPUT, where the input record is passed to these methods, I_INPFLD passes only the data for the field.
  - If a field has a drill method, the drilldown button is now displayed. If a field has a hyperlink method, the prompt is now green and clickable. You can invoke these methods by either clicking the drilldown button or green prompt or by signaling their menu entries. I_INPFLD passes only the data for the field.

  (8.1)
- (Windows) You can now control how a set of radio buttons will be arranged. You can control this by specifying a height for the associated selection window. This setting limits the number of radio buttons that Toolkit can place in a single column of buttons. (For example, if you set the height of the selection window to three, and the field has seven radio buttons, the radio buttons will be arranged in three columns. The first and second columns will have three radio buttons each, and the third column will have one.) If you set the height to one, Toolkit will include only one radio button in a column, which will cause the radio buttons to be arranged horizontally. Note that this may change the appearance of existing windows that have radio buttons because the height of the selection window, which was previously ignored, is now used. (8.1)
- We modified the behavior of IB_FIELD so that D_OFF followed by either D_FLD_PROMPT or D_FLD_PROMPTLEN will cancel either a text prompt or a runtime prompt. (Previously, using D_OFF with D_FLD_PROMPT would cancel only a text prompt, and using D_OFF with D_FLD_PROMPTLEN would cancel only a runtime prompt.) This makes the behavior of IB_FIELD consistent with window scripts, where “noprompt” is used to cancel either type of prompt. (8.1)

Linked-list subroutines

- We changed LL_PROCESS to a ^VAL function whose return value is the internal record number for the current record, and we added a fifth, optional argument that accepts an internal record number for the D_LL_JUMP operation. This enables you to maintain more than one destination at a time for D_LL_JUMP operations. (Note that because %LL_PROCESS is a ^VAL function, you can still use XCALL to call LL_PROCESS.) (8.3)

List subroutines

- (Windows) We now distribute version 7.0.0.33 of the FarPoint Spread ActiveX control, which is the control we use for ActiveX Toolkit lists. (8.3.1c)
- We added the following L_STATUS options to enable you to get more information about a list.
  - D_LHDRSTATIC, which indicates whether headers are static.
  - D_LFTRSTATIC, which indicates whether footers are static.
  - D_LDATALEN, which indicates whether a data area was passed to L_CREATE and, if so, its length.
  - D_LMETHOD, which returns the name or address of a list method (load, arrive, leave, or double-click) or the ID of the UWNDEVENTS_METHOD set for a list.
  - D_LLOADSTATE, which enables you to retrieve the load state of the top or bottom of a list. This option returns values that indicate if loading is in progress, if loading hasn't begun, if there are no more items needed, or if D_LEOF has been passed.
  - D_LQAVAIL, which returns the number of requests that can be added to the request queue before exceeding the 20 item limit.

  (8.3.1b)
- (Windows) We now distribute version 7.0.0.31 of the FarPoint Spread ActiveX control, which is the control we use for ActiveX Toolkit lists. (8.3.1a)
- (Windows) We implemented mouse wheel support for Toolkit lists. (ActiveX lists already support mouse wheel scrolling.) (8.3)
- We added two new options to L_STATUS: D_LCTRID, which returns the ID of the container window for a list, and D_LCOLSIZED, which returns a Boolean value that indicates whether the user has resized one or more columns in an ActiveX Toolkit list. (8.3)
- (Windows) We now distribute version 7.0.0.19 of the FarPoint Spread ActiveX control. (8.1.7c)
- (Windows) We improved the performance of the ActiveX list control, particularly when loading new items. (8.1.7c)
- (Windows) We reduced the amount of flicker (Toolkit repainting) that takes place when you resize a list while the Windows Control Panel option “Show window content while dragging” is selected. (8.1.7c)
- (Windows) Because we changed Toolkit’s default behavior for the ActiveX list control to reduce flicker (Toolkit now leaves the focus rectangle off until the list receives focus), we added a new published property, ActiveCellHighlightStyle, which enables you to restore the previous behavior. If you set this published property to 1, the cell focus rectangle will be displayed even when the list doesn’t have focus. (8.1.7)

- (Windows) We now distribute version 7.0.0.7 of the FarPoint™ Spread™ control. (8.1.7)
- (Windows) We added two new published properties to the ActiveX list control: SelBackColor and SelForeColor. These specify RGB values for the background and text colors for cells in a selected row and are used only when the RowMode property is set to a non-zero value. (8.1.7)
- (Windows) We changed the ActiveX list control to further reduce repainting. (8.1.5d)
- (Windows) We improved the performance of the ActiveX list control by reducing the amount of repainting required and by reducing the amount of time it takes to load the control in some cases. (8.1.5b)
- (Windows) We now distribute version 6.0.0.27 of the FarPoint Spread control. (8.1.5b)
- (Windows) We reduced the amount of repainting that takes place for the ActiveX list control. (8.1.5)
- (Windows) We now distribute version 6.0.0.18 of the FarPoint Spread control. (8.1.5)
- We added a new reason code, D_LABT_NOENABLED, which is returned by a call to L_STATUS with D_LABTRSN after L_SELECT has returned the *request* argument as D_LABORT because the *skip_disabled* argument for L_SELECT was passed as TRUE, but the list didn't contain any enabled items. Previously, L_SELECT did return the *request* argument as D_LABORT correctly, but a subsequent call to L_STATUS with D_LABTRSN returned zero. (8.1.5)
- (OpenVMS) We added a map for L_BUTTONSET to DTK_NULL in **tools.def**. This allows cross-platform code to reference L_BUTTONSET and still link on OpenVMS. (8.1.3)
- (Windows) We added a new extended property for the ActiveX list control: RowMode. When this is set to a non-zero value, selecting a cell highlights the entire row that contains the cell. When this property is set to zero, selecting a cell selects only the cell (not the row). Whenever L_INPUT or L_INPFLD moves focus to a modifiable field, RowMode is automatically reset to zero. This property is designed to make a list appear more like a selection window when used with L_SELECT. (8.1)
- (Windows) We upgraded the ActiveX list control to use FarPoint Spread version 6.0. In addition, %GRID_CREATE now uses version 6.0 unless you pass the control ID for a different version. (8.1)
- (Windows) If the FarPoint Spread control or the ActiveX list control are not registered, Toolkit now automatically registers them on the first attempt to load an ActiveX list control. Previously, Toolkit generated a fatal error if you attempted to load an ActiveX list control when these weren't registered. (8.1)

Menu subroutines

- We added a new argument to MB_COLUMN that enables you to specify a quick-select character for a column that is built at runtime. (8.3)
- We added a new option (D_NOLOAD) to MB_COLUMN that enables you to create a menu column at runtime without loading it. The *no_place* argument has been renamed to *options* and is now used to determine both column loading and placement options. A value of 1 (D_NOPLC) loads the column without placing it, and a value of 2 (D_NOLOAD) creates the column without loading or placing it. If you are creating pop-up columns or submenu columns, or if you are creating a column just to save it to a window library for later use, the D_NOLOAD option can save space in the memory control structure (which is limited to 65,535 bytes on Windows and UNIX and 8,192 bytes on OpenVMS). This modification may break your code. Previously, if you passed any value greater than 0 for the *no_place* argument, Toolkit would load the column without placing it. Now a value of 2 invokes the new D_NOLOAD behavior, and a value that is higher than 2 may cause Toolkit to place or load the column, depending on the first and second bits of the value's binary representation. (8.3)
- We increased the menu control space (the memory available for all loaded columns) from 8,192 bytes to 65,535 bytes. (8.1.7)
- We added the DTKKEYCTLFIL environment variable, which enables you to specify a path and filename for the key mapping script file. If you use DTKKEYCTLFIL, the file you specify replaces the default key mapping script file, **keymap.ctl**. (8.1.7)
- We added a new optional argument (D_REMOVE) to %M_SIGNAL. This argument instructs Toolkit to cancel a signaled menu entry. By default, however, %M_SIGNAL is set to D_SIGNAL, which signals a menu entry. (8.1)
- We added a new routine, %M_INFO, that returns information about the menu system. This includes information about menu columns, entries, lists, headers, and shortcuts. (8.1)
- (Windows) We added a new file, **winkeys.wsc**, to the Toolkit distribution on Windows. This file defines menu columns for **keys.dbr** that are based on the Windows shortcuts. (**Keys.dbr** is the file you create from **keys.dbl**. It enables you to test key mapping.) (8.1)

Selection processing

- We added an optional third argument to S_SELDFLT. This argument enables you to retrieve the default entry for a selection window or a set of radio buttons. Additionally, we made the second argument optional, so you can retrieve the default (set using S_SELDFLT) without also setting it. (8.3)

Toolbar subroutines

- (Windows XP) We added a new environment variable, SYN_3D_TOOLBAR, which applies the “hot” style to toolbar buttons when set to 1, giving them a 3-D edge with Windows XP themes. (8.3.1c)
- We improved U_ABORT so that it can now throw trappable errors. Previously, whenever U_ABORT was invoked, it would perform a Synergy Language STOP and display a fatal error message to the screen. This behavior may be adequate for an interactive Toolkit application, where the error message can be displayed to the screen. But error trapping gives you greater control over errors generated by U_ABORT and enables server-side applications and applications running as services to pass error information on to client programs. To enable you to activate the new behavior (or revert to the original behavior, which is the default), we added a global variable, **g_throwabort**, and an environment variable, DTK_THROW_ABORT. (8.3.1d)

Utility subroutines

- (Windows) We added the following options which enable you to center the %U_GETFILENAME and %U_MSGBOX dialog boxes over the application container window:
  - D_MCENTER, an option for the *styles* argument for %U_MSGBOX
  - D_OFN_CENTER, an option for %U_GETFILENAME

  (8.3)
- We improved U_ABORT so that it can now throw trappable errors. Previously, whenever U_ABORT was invoked, it would perform a Synergy Language STOP and display a fatal error message to the screen. This behavior may be adequate for an interactive Toolkit application, where the error message can be displayed to the screen. But error trapping gives you greater control over errors generated by U_ABORT and enables server-side applications and applications running as services to pass error information on to client programs. To enable you to activate the new behavior (or revert to the original behavior, which is the default), we added a global variable, **g_throwabort**, and an environment variable, DTK_THROW_ABORT. (8.3)
- The U_OPEN subroutine now supports filenames with up to 256 characters. Previously it supported no more than 94 characters. (8.1.5d)
- (OpenVMS) We added a map for U_SAVESETTINGS to DTK_NULL in **tools.def**. This allows cross-platform code to reference U_SAVESETTINGS and still link on OpenVMS. (8.1.3)

- (Windows) We changed the following utility routines to save information to the user-specific initialization file (**synuser.ini**) by default. Note that we also added a mechanism (argument, bit flag, etc.) to each routine to enable you to write to **synergy.ini** instead.
  - U_EDITREND. To enable you to save to **synergy.ini**, we added a checkbox, "For all users", to the Renditions dialog box.
  - U_SAVELOG. To enable you to save to **synergy.ini**, we added an optional argument, *all_users*.
  - U_SAVESETTINGS. To enable you to save to **synergy.ini**, we added an optional bit flag, D_APP_ALLUSERS.
  - The D_SAVEFONT subfunction to %U_WNDFONT. To enable you to save to **synergy.ini**, we added an optional argument, *all_users*.

  Note that this may break your code because we have changed the default location of **synergy.ini** from the Windows (or Winnt) directory to the directory specified by DBLDIR. For information, see "Features and fixes that may break your existing code" on page 3-19. (8.1)
- %U_CMDLINOPT is now declared as an external function (^val) in **tools.def**. (8.1)

Window Designer

- As part of the support for scrolling input fields (see "Input subroutines" on page 3-22), we added the following properties for input fields: Display length, Input length, and View length. For information on these properties, see the online help for Window Designer. (8.1)

Miscellaneous

- (Windows) We added a new variable, **g_txt_softreturn**, to **tools.def**. This variable affects the way Toolkit handles line ends on Windows when using T_EDIT or when using I_INPUT with multiline text fields. When set to zero (which is the default), **g_txt_softreturn** instructs Toolkit to retain line ends in incoming data and to use those line ends when displaying the data. When set to a non-zero value, Toolkit assumes that line ends are the result of wrapping and discards them, rewrapping lines as needed when displaying the data. (8.3.1d)
- We expanded Toolkit's bounds checking (DTK_BOUNDS=1) to include checking the size of the record passed to I_DISPLAY, I_GETFLD, I_PUTFLD, and I_INIT, as well as the size of the data area passed to I_DSPFLD. (8.3.1d)
- We moved the definition of D_LMAXFIND (the maximum number of characters in a find) from **lstprv.def** to **tools.def** to make it a public definition. (8.3.1b)

- (Windows) We improved Synergy/DE's color support to accommodate Windows system colors. You can now assign Windows system colors to the various UI elements in Toolkit applications (and applications that use the Synergy Language windowing subroutines) by assigning the system colors to Synergy color palette entries. To implement this, we made the following changes:
  - We increased the number of colors that Synergy/DE supports to 512 (color 0 through color 511), and we assigned Windows system colors to color 256 through color 285. Colors 0 through 255 are user-defined colors, as they have been in previous versions, and colors 286 through 511 are reserved for future Windows system colors.
  - We improved the dialog box (formerly named Define Colors) that opens when you click the Change Colors button in the Renditions window (the window that opens when you call U_EDITREND or U_MODREND) or when you change a color palette entry in Composer. This dialog box now enables you to choose colors that correspond to Windows system colors. Additionally, we made the dialog box easier to use by making it apply only to the color palette entry that is selected in the Rendition window when you click the Change Colors button. The name for the dialog box is now "Palette Entry #", where # is the number of the palette entry that will be affected by changes.
  - We removed Synergy color settings and the palette setting from the distributed **synergy.ini** file. In previous versions, the Synergy runtime defaults were used only if you removed these overrides or used a different **synergy.ini** file.
  - We changed the foreground and background colors for the first, third, and ninth color palette entries in the default palette to bring default palette and renditions settings closer to standard Window settings. The foreground color for these palette entries is now 264, the background color for the first and third entries is now 271, and the background for the ninth palette entry is now 276.
  - We added the **g_tabfacecolor** global variable to **tkctl.def**. This variable determines which color palette entry will be applied to a window when it is added to a tab set. On Windows, if themes are enabled, **g_tabfacecolor** is set to palette entry 9 by default to match the face of the tab control. (On UNIX and OpenVMS, **g_tabfacecolor** is set to 0 by default, so Toolkit does not change a window's color palette entry when it is added to a tab set.)

  This modification may break your code. To accommodate the increased number of Synergy colors, U_REND and the WI_PALET subfunction to W_INFO may now return values that are too large for **i1** variables (which were sufficient for all color values in previous versions). Check your code to make sure variables used to store these values are no smaller than **i2** or **d3**. (We recommend using **i4** variables.)

  Also note that the changes to the first, third, and ninth entries of the default color palette and the absence of color and palette definitions in the distributed **synergy.ini** file may change the appearance of an application if you install Synergy/DE on a new system. (8.3)

- We expanded Toolkit's bounds checking (DTK_BOUNDS=1) to include checking the size of the record passed to I_DISPLAY, I_GETFLD, I_PUTFLD, and I_INIT, as well as the size of the data area passed to I_DSPFLD. (8.3)
- (OpenVMS) The protections for the **dtkmap.ism** file have been modified to allow world write permissions. (8.3)
- (Windows XP, Server 2003) We added support for themes. Note that you will see a change in behavior with regards to dropped-down combo boxes. When a combo box is dropped down, a single mouse click outside the combo box will pull it back up, but will not select whatever was under the mouse. A second mouse click is required to select the field, button, Close box, etc. This is standard behavior. (8.1.7a)
- We increased the amount of memory available for Toolkit channel information from 5,000 bytes to 65,535 bytes to avoid "Data space overflow" errors when opening a large number of files with long names. (8.1.7)
- We improved performance, regaining losses incurred when we increased the maximum size for input fields to 65,535 bytes. (8.1.7)
- The **keymap.def**, **chninf.def**, and **mnuctl.def** files now include **.def** files from WND: rather than UTL:. (8.1.5)

Version 7

This section briefly describes new features in UI Toolkit version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to UI Toolkit version 7.

- If you are including the **gblctl.def** file (which we don't recommend), you must recompile and relink. We modified this file to support environment level toolbars. (7.5)
- In versions prior to 7.5, if a group definition in the repository specified a prefix, Toolkit added the prefix to the names of only the first-level members of that group. We have corrected this. Toolkit now adds the prefix to all members, regardless of level. If multiple prefixes are specified at different levels, the prefixes are appended to each other, outermost first. (7.5)
- (Windows) In previous versions of Toolkit, an hourglass cursor would be changed back to a normal cursor when the Toolkit deemed it appropriate to do so. This resulted in situations where an unwanted change could occur. This behavior has been removed. Setting the cursor to an hourglass will now cause the cursor to remain an hourglass until reset. If your program relies on Toolkit to reset the cursor, you will now need to add code for this. (7.3.3)
- (Windows) In %U_WNDEVENTS, we now set the D_LSTCLK_ONLINE flag in the **a_flags** parameter of a mouse click event when the user selects the border between two column headers in an ActiveX list. If you were unconditionally signaling a menu entry, this may break your code because it will cause the drag resize of the column to be ignored. (7.3.3)
- (Windows) If a user selects the border between two column headers in an ActiveX list, the D_LSTCLK_ONLINE flag is now set. (D_LSTCLK_ONLINE is a %UWNDEVENTS_METHOD flag for mouse click events.) If your code unconditionally signals a menu entry, this may break your code. It will cause the drag resize of the column to be ignored. (7.3)
- (Windows) In previous releases of UI Toolkit, updates to the information line or footer were not always deferred to the next W_UPDT (or implied update). They were deferred unless text for that section (left/center/right) had previously been set. This has been corrected, so these changes are now always deferred to the next update. If you were depending on UI Toolkit to update automatically for you, you must now perform a U_UPDATE. (7.1)

- We defined the size argument for the D_FLD_SIZE field qualifier for IB_FIELD and I_FLDMOD to be the input field's storage size. Previously, this argument was not strictly defined and could have been used to signify the maximum input length or the display length. We added two new field qualifiers for specifying these lengths: D_FLD_INPLEN and D_FLD_DSPLEN, respectively. Note that the length argument has not been affected and is still used to specify the maximum input length and, potentially, the display length (if no format exists on the field). (7.1)
- (Windows) We added support for extended standard buttons. In doing so, we removed support for the STD_BUTTONS environment variable, and for the modification of the **g_entnamok**, **g_entnamcancel**, and **g_entnamhelp** globals. (7.1)

New features

ActiveX subroutines

- (Windows) We added several routines that enable you to directly access the properties and methods for a single ActiveX control in a Toolkit container (one created and loaded by using %GRID_CREATE or %AX_TKSINGLE):
  - %AX_TKCTLID, which enables you to retrieve an ActiveX control ID.
  - the following routines, which call extension routines for their corresponding %UWNDEVENTS_METHOD events. These routines do nothing unless you pass the prefix argument to %AX_TKSINGLE or %GRID_CREATE.

    AX_TKEVENT_CLOSE
    AX_TKEVENT_LEFT_CLICK
    AX_TKEVENT_LEFT_DBLCLK
    AX_TKEVENT_MAXIMIZE
    AX_TKEVENT_MIDDLE_CLICK
    AX_TKEVENT_MIDDLE_DBLCLK
    AX_TKEVENT_MINIMIZE
    AX_TKEVENT_MOVE
    AX_TKEVENT_RESTORE
    AX_TKEVENT_RIGHT_CLICK
    AX_TKEVENT_RIGHT_DBLCLK
    AX_TKEVENT_SIZE
    AX_TKEVENT_SCROLL
  - AX_TKSET, which enables you to set a property for an ActiveX control.
  - AX_TKGET, which enables you to retrieve a property for an ActiveX control.
  - AX_TKGETINT, which enables you to retrieve an ActiveX control's property as an integer.
  - %AX_TKCALL, which enables you to call a method for an ActiveX control.

  (7.5)

- (Windows) We added a new routine, %GRID_CREATE, that enables you to create and use the FarPoint Spread control directly. Constant values required for certain properties and method arguments are defined in a new include file, **grid.def**. Note that you must supply a developer license string to use the control. (7.5)
- (Windows) We added a new routine, %AX_TKSINGLE that enables you to create a Toolkit container and load a single ActiveX control. (7.5)
- (Windows) We added D_SIGNAL, a subfunction for %U_WNDEVENTS, which enables you to trigger a routine that's registered as part of a method set for a Toolkit window. (7.5)

Button subroutines

- We added the subroutines B_BUTTON and B_BUTTONSET for specifying a button and button set on a window or tab set. (7.1)
- (Windows) We added stub routines for %B_ENABLE and %B_DISABLE to UI Toolkit, so you no longer need to .IFDEF the references to these functions for the Windows environment. (7.1)

Debugger

- UI Toolkit now uses the DTK_BOUNDS environment variable to set bounds checking. To turn on bounds checking set DTK_BOUNDS=1. (7.1)

Environment subroutines

- We added optional arguments to the EUTILS_METHOD syntax. When called from within I_INPUT, I_INPFLD, L_INPUT, or L_INPFLD, a second argument containing the inputinfo structure (defined in **inpinf.def**) is passed. In the third through twenty-second argument positions, the optional method data arguments passed to I_INPUT, I_INPFLD, L_INPUT, L_INPFLD, L_CHR, and L_SELECT are passed in turn to the EUTILS_METHOD routine. (7.5)
- We added the %E_INFO function which enables you to return information about event processing. (7.3)
- (Windows) The D_VALSTCHG option to E_METHOD can now be used for immediate validation of combo boxes on Windows. (7.1)
- We modified the E_METHOD subroutine to accept numeric routine addresses in addition to routine names. (7.1)
- We added a new state flag, D_CBADVANCE, to the E_STATE subroutine. On Windows, if D_CBADVANCE is turned off, context will *not* automatically advance when an entry is selected in a combo box. (7.1)
- We added the D_METH_APPMOVE, D_METH_APPSIZE, and D_METH_APPSTATE events to the E_METHOD subroutine. These events are initiated by moving the application window, resizing the application window, or changing the application window's state, respectively. (7.1)

- We added the D_METH_ENTRST option to the E_METHOD subroutine to override input cancellation when a menu entry is selected. This option allows you to specify the routine to call instead of the user-replaceable subroutine USR_ENTRST. (7.1)
- We added the D_METH_SCRIPTERR option to the E_METHOD subroutine that allows you to specify the method to invoke when script compilation errors occur when using the external routine interface to the Script compiler. (7.1)
- We added a new function, %E_FONT, that allows you to modify the font for the header, footer, or information line section of the application window. (7.1)
- We added a new argument to ECHKFLD_METHOD, EDSPFLD_METHOD, and EEDTDSP_METHOD. The *a_inpinfo* argument contains input information for user-defined fields. (7.1)

Input subroutines

- We added two new input field methods for display and edit formatting: %IDISPLAY_METHOD and %IEDITFMT_METHOD. (7.3)
- We added the following qualifiers to I_FLDMOD: D_FLD_POS (for the prompt position) and D_FLD_FPOS (for the field position). (7.3)
- (Windows) We added support for the **g_txt_rtrn** flag (in **tools.def**). If this flag is on, ENTER is used by multi-line edit controls as a hard line break instead of field advance or default button click. Note that because **g_txt_rtrn** is merely a flag, changing its value within an input field method will have no effect until I_INPUT is exited and re-entered. (7.1)
- We added the following qualifiers to I_FLDMOD: D_FLD_INFO, D_FLD_HELP, D_FLD_USER, D_FLD_DFLT, D_FLD_INCR, D_FLD_DECR, and D_FLD_COPY. These enable the information line, help identifier, user text, and default value to be modified at runtime. In addition, we added D_USERTYPE for specifying a user-defined field. (7.1)
- We added three new qualifiers to the I_ENABLE and I_DISABLE subroutines for enabling and disabling input fields. D_FLDS affects specified fields in a window, D_SET affects all fields in a set, and D_ALL affects all fields in a window. (7.1)
- We added the field qualifiers D_FLD_COLOR and D_FLD_ATTR to the I_FLDMOD and IB_FIELD subroutines for specifying color and attributes for an input field. (7.1)
- We added the field qualifiers D_FLD_DSPLEN and D_FLD_INPLEN to the I_FLDMOD and IB_FIELD subroutines for specifying the display length and maximum input length for an input field, respectively. (7.1)
- D_FLD_DISABLED and D_FLD_READONLY have been added to IB_FIELD and I_FLDMOD to enable you to specify a field as disabled or read-only at runtime. (7.1)
- We added a new subfunction (D_FLD_RANGE) to I_FLDMOD that allows the range for an input field to be modified at runtime. (7.1)

- I_FLDMOD and I_FLDSEL can now be called directly from within an input method. Note that this is not exactly equivalent to calling %M_SIGNAL because the field modification occurs within I_INPUT. (7.1)
- We added a new optional argument to the IB_BUTTONSET subroutine. The *button_name* argument allows a default button to be specified for a window. Specifying a default button, when used in conjunction with E_STATE(D_RETURNBTN), allows the RETURN (or ENTER) key to be used to select the button. (7.1)

List subroutines

- The Toolkit list processor now allows up to 16,290 bytes for the non-window data area. (7.5)
- You can now instruct Toolkit to search the display area (the input window associated with a list) or the non-window data for a list. (Formerly, UI Toolkit could search only the non-window data.) To support this, we added an optional argument to the L_FINDSPEC subroutine. For backward compatibility, non-window data is searched by default. In addition, L_STATUS can return the current setting using the new D_LFINDSOURCE argument flag. (7.3.3)
- We added new advanced ActiveX list features for the ActiveX list, such as the ability to set color, font, and column widths. We also added new flags to the click events for lists. (7.3)
- We added a new routine, L_BORDER, which enables you to turn the border and/or dragbar (caption bar) on or off for a list. (7.3)
- (Windows) We extended AX_WANTSKEY to allow you to more accurately specify the kinds of key mapping you want to occur when an ActiveX control is active. (7.1)
- We improved AX_INPUT to handle type-ahead better. (7.1)
- We added support for the wait value for L_SELECT and for a field in L_INPUT. We also added a new optional third argument (*wait_time*) to AX_INPUT. Additionally, you can call the new AX_TIMEOUT routine from an ActiveX method to reset the timer. (To do this, pass the wait value as the only argument.) (7.1)
- We significantly reduced the amount of repaint that occurs on the ActiveX list control. (7.1)
- We added a new optional fourth argument to AX_INPUT that if passed and TRUE causes AX_INPUT to return utility menu entries (U_ and O_HELP) instead of processing them. (7.1)
- We added the subroutines L_BUTTON and L_BUTTONSET for specifying a button and button set on a list. (7.1)
- We added the function %L_BUTTONSTATE for enabling and disabling a button on a list. (7.1)
- We added 20 optional data arguments to the L_CHR, L_INPFLD, L_INPUT, L_PROCESS, L_SELECT, LARRIVE_METHOD, LLEAVE_METHOD, LDBLCLICK_METHOD, and LLOAD_METHOD subroutines. (7.1)

- We added the subroutine L_RESIZE for resizing a list. (7.1)
- We added the subroutine L_STATE for specifying the state of a list. (7.1)
- We added the flag D_LSTATE to the L_STATUS subroutine for retrieving the state of a list. (7.1)

Menu subroutines

- We added a new function, %M_POPUP, that allows you to pop up a menu anywhere on the screen. (7.3)
- We changed the USR_ENTRST routine to be a method registered by E_METHOD. If you include **gblctl.def** (which is not recommended), you *must* recompile the including modules. (7.1)
- We added the D_SUBMENU option to the MB_ENTRY subroutine to allow creation of a menu entry that refers to a submenu. (7.1)

Script compiler

- A shortcut to the Script utility has been added to the Start menu. (7.3.3)
- We added a new .LISTCLASS option, "NOAUTOSIZE", which prevents the size of a list from being affected by the size of its container. When in effect, the list can only be resized by L_RESIZE. (7.3)
- The **gs_class** field in the **gs_inpfld** structure (defined in **fldinf.def**) now contains the ODBC subtype for user-defined fields. The ODBC subtype is defined by the Class field in repository. (7.1)
- We added the ability to access group members if their names are unique among all members. In Repository, we added the ability to define a group in the Repository as referencing a structure, and we added the ability to assign a prefix for each group. (7.1)
- You can now combine multiple repository structures, enabling their access as one structure by using the implicit group feature of S/DE Repository. (7.1)
- We added an external routine interface for the Script compiler. The %SCR_OPENLIBRARY, %SCR_CLOSELIBRARY, %SCR_PROCESS, and %SCR_ERRORCOUNT functions allow you to compile window scripts and retrieve compilation error information. (7.1)
- We now support the .BUTTON and .BUTTON_SET commands for a general window. (7.1)
- We added the input field qualifiers COLOR, NOCOLOR, HIGHLIGHT, NOHIGHLIGHT, BOLD, NOBOLD, REVERSE, NOREVERSE, BLINK, NOBLINK, ITALIC, NOITALIC, UNDERLINE, and NOUNDERLINE for specifying renditions for an input field. (7.1)
- We added the input field qualifiers DISABLE, NODISABLE, ENABLE (same as NODISABLE), READONLY, and NOREADONLY to enable you to specify a field as disabled, enabled, read-only, or writable. (7.1)
- We added support for specifying range values for date and time fields by using the existing RANGE qualifier for the .FIELD command. (7.1)

Script conversion utility

- The script conversion utility now checks to see if a script has already been converted. If it has, the script conversion utility does not convert it. (7.1)

Selection processing

- We increased the maximum number of entries in a selection window from 100 to 32,767. You can take advantage of this new maximum with S_SELBLD, S_SELECT, and the .SELECT script command. However, this doesn't apply to the ",select" input field qualifier. (7.5)

Toolbar processing

- You can now make a toolbar part of an environment. To support this, we increased the maximum number of toolbars that can be created from 20 to 256, and we added
  - TB_PUSHSTATE and TB_POPSTATE, subfunctions to the %TB_TOOLBAR function. These subfunctions enable you to save and restore a toolbar's state.
  - %TB_TKCREATE, a new function that enables you to create a toolbar that's a part of an environment.
  - TB_TKLOG, a new routine that enables you to log a toolbar with an environment level.
  - TB_TKGLOBAL, a new routine that enables you to promote a toolbar to the global environment level.
  - TB_TKDEL, a new routine that enables you to delete a toolbar that's assigned to an environment.
  - D_TOOLBARS and D_TOOLBAR_LEVEL, subfunctions to the %E_INFO function. These subfunctions enable you to retrieve a list of toolbars for a given level (D_TOOLBARS) and retrieve the environment level for a toolbar (D_TOOLBAR_LEVEL).

  Note: If you include **gblctl.def**, you must recompile and relink. We modified this file to support this feature. (7.5)

Tab set subroutines

- We added an optional returned argument to the DTS_TABINFO subfunction of %TS_TABSET. The new argument returns TRUE if the tab is disabled or false if it isn't. (7.1)
- We added an optional argument to the DTS_BUTTONSET subfunction of the %TS_TABSET function. The *button_name* argument allows a default button to be specified for a tab set. Specifying a default button, when used in conjunction with E_STATE(D_RETURNBTN), allows the RETURN (or ENTER) key to be used to select the button. (7.1)

- We added six new subfunctions to the %TS_TABSET function. The DTS_SELECT_ATTR and DTS_NONSELECT_ATTR subfunctions modify the attributes for the selected tab and for non-selected tabs, respectively. The DTS_DEFAULT and DTS_GETDEFAULT subfunctions modify and retrieve default settings for tab sets, respectively. The DTS_TABINDEX and DTS_TABINFO subfunctions retrieve the tab index for a contained object and information for a specific tab, respectively. (7.1)
- We added three new arguments to the DTS_CREATE subfunction of the %TS_TABSET function. The *pre_tab_string* and *post_tab_string* arguments specify the delimiters for a tab set on a UNIX or OpenVMS system. The *style* argument specifies the tabbed dialog style on Windows, which allows multi-line tabs and tabs on any side of a tab set. (7.1)

Utility subroutines

- (Windows) We added a new field to the *info_record* record for the %U_PRINTQUERY function. The extprinterid field supports 256 character printer names, which are required for Windows 2000 and Windows XP. (7.5)
- (Windows) We added several new functions and subfunctions that enable you to get Windows-specific information:
  - %U_WINMETRICS, a function that enables you to get the height and width of characters, the space between rows, the height of a row, the width and thickness of window frames, the width and thickness of borders, and many other types of information.
  - %U_WININFO, a function that returns the current pixel coordinates for the mouse pointer, the instance handle for the Synergy runtime, or the ASCII character that corresponds to a virtual key code.
  - %U_WINCELLS, a function that converts pixels to fractional rows or columns.
  - %U_WINCOLOR, a function that opens the Choose Color dialog box with an initial selection, returns the 256 RGB values associated with Synergy colors, returns the RGB value for Toolkit hyperlink prompts, returns the foreground and background colors for a Synergy palette entry, or uses an RGB value to set a Synergy color.
  - D_FONTSIZE, a subfunction to %U_WNDFONT that returns font size information (height and width) for a given Synergy font palette entry.

  (7.5)
- (Windows) U_REND now allows integer as well as decimal arrays. (7.5)
- We added the %U_GETWNDERR function to retrieve a window system error. (7.1)
- We added five new events to the %U_WNDEVENTS function. The D_EVENT_SIZE, D_EVENT_MINIMIZE, D_EVENT_MAXIMIZE, D_EVENT_RESTORE, and D_EVENT_SCROLL events allow a window to be resized, minimized, maximized, restored, and scrolled, respectively. (7.1)

Window Designer

- We added a percentage complete display that opens while resources are being released on shutdown. For especially large script files, this lets the user know that Window Designer has not hung. (7.1)
- (UNIX, OpenVMS) Buttons are now listed in the selection list of objects that can be created within an input window. (7.1)

Miscellaneous

- The Toolkit linked list API now allows up to 16,304 byte records. (7.5)
- We have significantly reduced the amount of system resources required for input fields. (7.5)

Version 6

This section briefly describes new features in UI Toolkit version 6 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features that may break your existing code

The following changes may break code when you update to UI Toolkit version 6.

- You must convert existing Toolkit window scripts (pre-V6) in order to use them with the version 6 Script compiler, Composer, or Window Designer. The **scrcnvrt.dbr** conversion program will convert a window script to the new version 6 format. See "Script compiler" on page 3-46 for more information. (6.1)
- We increased the maximum number of windows that can be loaded at one time from 98 to 1024. The new maximum also applies to lists. The maximum number of windows that can be placed in an environment level has been increased to 1024 as well. This modification may break your existing code if you use only **d2**s for window IDs. We recommend you use **i4**s. If you are including from any of the following Toolkit definition files, you must recompile and relink: **chninf.def**, **fldinf.def**, **gblctl.def**, **lstprv.def**, **mnuctl.def**, **system.def**, or **tkctl.def**. (6.1)
- The Script compiler, the Script-to-Repository Conversion program, **scridl**, and the IB_*xxx* subroutines now look for the new Repository environment variables, RPSMFIL, RPSTFIL, and RPSDAT, instead of ICSMFIL, ICSTFIL, and ICSDAT. (6.1)
- The Repository Subroutine Library, **ddlib.olb/ddlib.elb**, is no longer distributed with UI Toolkit; **ddlib.elb** is only distributed with the Repository. This modification will break your code if you are linking against it (**WND:ddlib.olb** or **WND:ddlib.elb**). You must now link against **RPSLIB:ddlib.elb**. (6.1)
- We changed the way UI Toolkit looks for the text message file, **syntxt.ism**. Previously, Toolkit attempted to open it as **TXT:syntxt**, then **DBLDIR:syntxt**. This meant that if TXT was not defined, Toolkit would find a copy of **syntxt** in the default directory on systems other than OpenVMS. We changed the symbol from TXT to SYNTXT so that a symbol collision is less likely, and we use it only if defined. This change may break your existing code if you are currently using the TXT logical. (6.1)
- We removed the "Font…" menu entry from the application window's System menu. This change may affect your application if you depend on the user's ability to change the font. To provide the same functionality from within your Toolkit application, see "Utility subroutines" on page 3-48. (6.1)

- We removed the following obsolete routines that appear in the left column. (In version 3, they were renamed to the names in the right column and removed from the documentation, but they still existed in **tklib**.) This change may break your code if you are still using the routine names in the left column. (6.1)

Removed routines	New routines
u_selbld	s_selbld
i_select	s_select
u_select_p	s_select_p
u_selld	s_selld
u_selinf	s_selinf

New features

General

- We added support for a runtime font palette that may include both fixed and proportional fonts. This font palette can be initialized using font definitions from your **synergy.ini** file and further modified at runtime using %U_WNDFONT. (6.3)

  Fonts in the font palette may be used as the global font, alternate font, or the font to apply to a specific general window, input window, selection window, input field, or prompt. The global and alternate fonts can be defined in the **synergy.ini** file or at runtime. Fonts for specific objects can be defined in a window script or at runtime.

- To help you develop your window scripts more easily and efficiently, the UI Toolkit distribution on Windows includes the new Composer application. Composer is an interactive design tool that allows for the visual creation and modification of window definitions. (6.1)

  UNIX and OpenVMS distributions include Window Designer and Proto for assistance in window script creation and modification.

- The Script Compiler, the Script-to-Repository Conversion program, **scridl**, and the IB_*xxx* subroutines have been updated to handle the new repository file layout changes for group support. UI Toolkit does not support the direct access of group members. Groups defined at the structure level may be included in a window definition as a single field. To access group members, define overlay fields at the structure level that offset into the desired group. (6.1)

- UI Toolkit honors the "Available to Toolkit?" flag, which is a repository field attribute. This flag defaults to YES, but when set to NO, prevents the field from being accessed by Script, Composer, Window Designer, and the IB_FIELD subroutine. (6.1)
- The following Toolkit routines have been made re-entrant and can therefore be called from within a hot-entry routine or a method: I_INPUT_P, I_INPFLD_P, L_CHR, L_CHR_P, L_INPFLD, L_INPFLD_P, L_INPUT, L_INPUT_P, L_SELECT, L_SELECT_P, S_SELECT, S_SELECT_P, T_EDIT, T_EDIT_P, U_FLD, U_FLD_P, U_CHR, U_CHR_P, L_PROCESS, and M_PROCESS.

Button subroutines

- We added the functions %B_ENABLE and %B_DISABLE for enabling and disabling buttons in input windows and tab sets. (6.3)

Environment subroutines

- We added a new state flag, D_RETURNBTN, to the E_STATE subroutine. On Windows systems, if D_RETURNBTN is set, the ENTER key has the effect of pressing the default button in an input window or tab set. (6.3)
- To enable immediate validation of check boxes and radio buttons, we added the D_VALSTCHG option to E_METHOD. Setting this option enables fields to be immediately validated before focus is lost. (6.3)
- To extend support for the year 2000, we added the D_METH_CENTURY option to E_METHOD. By registering a century method, the default century can be overloaded. (6.3)
- We added the ability to apply European formatting within the E_STATE subroutine, using the new D_EURO code defined in **tools.def**. (6.3)

Input subroutines

- We added a new optional argument to the I_SETINF subroutine. The *processing_state* argument returns the processing state for an input set. The processing state can be normal, done, default, or OK. (6.3)
- We added a new optional argument to the IB_BUTTON subroutine. The *select_char* argument allows an activation character to be specified for a button. Specifying an activation character underlines the first occurrence of the character on the button if the button has text. Pressing ALT + the activation character simulates a button press. (6.3)
- D_FLD_FONT and D_FLD_PROMPTFONT have been added to IB_FIELD and I_FLDMOD to enable you to specify fonts for fields and prompts at runtime. (6.3)

- D_FLD_CHANGE has been added to IB_FIELD and I_FLDMOD to enable you to specify or clear the change method for a field. (6.3)
- We added two fields to the **gs_inpfld** structure returned from I_FLDINF. These fields, **gs_font** and **gs_promptfont**, contain the font names for a field and a prompt, respectively, or are blank if no font is specified. (6.3)
- IB_FIELD now reports an error if the field type passed is invalid. (6.1)
- We changed the behavior of radio buttons on Windows: the UP/DOWN ARROW keys will move the focus to the previous/next radio button in the group if a radio button is focused. This overrides the use of the ARROW keys for menu entries while the radio button is focused. If there is no next or previous entry, nothing happens. (6.1)
- Support has been added for extended standard buttons. This feature enables you to define an input window "button set," specifying its location in the window, its justification and the number of buttons per row. Each button within the set can have either a text or bitmap face, and can invoke a subroutine, or signal a menu entry. (6.1)
- We added two new subroutines, IB_BUTTON and IB_BUTTONSET, that replace the subroutine IB_STDBUTTONS. IB_STDBUTTONS still exists (it now calls IB_BUTTON three times), but we recommend that you use the new routines to avoid the unnecessary overhead. (6.1)
- We added a new subroutine IB_RPS_STRUCTURE, that replaces the subroutine IB_DICTIONARY. IB_DICTIONARY is still supported (it calls IB_RPS_STRUCTURE), but we recommend that you use the new subroutine to avoid the unnecessary overhead. (6.1)
- We added a new subroutine, I_FRAMES, which allows you to show the frames on input fields without processing I_INPUT. This subroutine does nothing in UNIX or OpenVMS environments. (3.7.9)

List subroutines

- We added a new function, %L_ICON, which allows you to set an icon for a list. (6.3.1)
- We added a new function, %L_USER, which allows you to create and access user data associated with a list. (6.1)
- The *no_termination* argument to L_INPUT is now ignored on Windows. It is always handled as D_NOTERM. This is necessary to support item-to-item movement within a list during L_INPUT. (6.1)

Menu subroutines

- We added the new reserved menu entries E_MARK, E_CUT, E_COPY, E_PASTE, and E_CLEAR to mark, cut, copy, paste, and clear text on Windows. These menu entries are handled automatically by input processing. (6.3)
- We added the EDIT_SYSMENU environment variable to allow the "Edit" menu to be removed from the application window's system menu. (6.3)
- We added a new function, %M_TEXT, which enables you to modify the text and/or quick-select character of a menu or submenu entry. It also enables you to retrieve the previous text of the entry. (6.3)
- We changed the menu processing on UNIX and OpenVMS environments to match that on Windows: when a unique menu quick-select character is pressed, it not only highlights but also selects the associated menu entry. If the quick-select character is not unique in the column, it simply highlights it as before. (6.1)

Script compiler

- We added the list class option string qualifiers ACTIVEX and NOACTIVEX to specify which list (ActiveX Toolkit list or non-ActiveX Toolkit list) to use by default. (6.3)
- We added a SELECT qualifier to the .BUTTON script command to enable an activation character to be specified for a button. Specifying an activation character underlines the first occurrence of the character on the button if the button has text. Pressing ALT + the activation character simulates a button press. (6.3)
- We added a .FONT script command for specifying a font for a general window, input window, or selection window. (6.3)
- We added the input field qualifiers CHANGE_METHOD and NOCHANGE_METHOD to enable you to override or add to the Toolkit validations for a field. Use these qualifiers rather than ECHKFLD_METHOD. (6.3)
- We added the input field qualifiers FONT, NOFONT, PROMPTFONT, and NOPROMPTFONT to enable you to specify and clear fonts for input fields and their prompts. (6.3)
- You must now have a development license (PSDE or PSW) to compile a script file from Proto or Window Designer. Previously, a development license was required only for the stand-alone Script compiler. (6.1)
- Script no longer supports script commands of the syntax eliminated by the conversion program described above. (6.1)
- We created a command line interface for Script. (6.1)

- We added a conversion program, **scrcnvrt.dbr**, which converts existing window scripts to the new version 6 format. The table below identifies the changes made by the conversion program.

  If your script file references a dictionary, you must convert it to the new repository format before running the **scrcnvrt** conversion program. See the Repository release notes file, **REL_RPS.TXT**, for more information on converting version 3 dictionaries.

Script Syntax Changes	
This...	**Becomes...**
.dictionary	.repository_structure
.std_button	.button_set and .button
.sub	.column
.draw	.box or .line
\f	.wfield
bold	highlight

  Additional changes:

  - The listclass command requires .end and may have an optional .placement
  - All comments must be preceded by a semicolon
  - Semicolons within text must be preceded by a backslash (\)
  - With the exception of comments, the first line of a script must be .script
  - All text objects (a new text object is created whenever there is a position or margin change within text) are preceded by ".text", with optional position and margin qualifiers

  (6.1)

- We removed the obsolete .FLAG script command for menu columns. (6.1)
- We changed the script processing of the .FIELD statement to allow continuation between the arguments to the POSITION and FIELD_POSITION qualifiers. (6.1)
- With previous versions of the Repository, there was no way to indicate that a field contained "no" paint character, as opposed to containing a paint character of blank. This meant that Toolkit always had to assume it was a paint character of blank, and, hence, there was no way to have the input window's paint character be in effect. The version 6.1 Repository allows you to specify a paint value of "None" or "Character." (6.1)

Selection subroutines

- We added an optional, fifth argument to S_SELINF, which, if passed and true, will search for the default entry as a numeric value. (6.1)

Tab set subroutines

- We added a new optional argument to the DTS_BUTTON subfunction of the %TS_TABSET function. The *select_char* argument allows an activation character to be specified for a button. Specifying an activation character underlines the first occurrence of the character on the button if the button has text. Pressing ALT + the activation character simulates a button press. (6.3)
- We added two new routines, TS_TABSET and TS_PROCESS, which enable you to create and process tabbed dialogs. (6.1)

Utility subroutines

- We added a new option to the %U_GETFILENAME function. If D_OFN_NOTRANS is specified for the *option* argument, logicals that are specified by the user will not be translated in the filename that is returned. (6.3)
- We added the U_SAVESETTINGS subroutine to write the position and state of the application window to the **synergy.ini** file. (6.3)

  To set the pixel position for the application window and set the state for the application window or a Synergy window, we added the subfunctions WP_POSITION and WP_STATE to W_PROC. In order to retrieve the position and state for a window, the subfunctions WIF_POSX, WIF_POSY, and WIF_STATE have been added to the %W_INFO function.

  The new APP_STATE initialization setting allows the state to be initialized in **synergy.ini**.
- We added four new subfunctions to %U_WNDFONT: D_SETWNDFONT, which sets a font for a window; D_GETWNDFONT, which retrieves font information for a window; D_FONTHANDLE, which returns the handle for a font; and D_FONTNAME, which retrieves the name for a font. (6.3)
- We added optional arguments to %U_WNDFONT(D_CHOOSEFONT) to include proportional fonts, use a sizing character, add a help button, and specify a title. (6.3)
- We added an optional argument to U_UPDATE which, if passed and false, turns off screen updates in Windows environments. (This argument is ignored on OpenVMS and UNIX.) (6.1.2)
- We added a function, %U_WNDFONT, which enables you to set or retrieve font information within UI Toolkit on Windows. This function can set a specific font, or it can display a standard Font dialog from which the user can select a font. Related to this change, we removed the “Font…” menu entry from the System menu. (6.1)

Version 3

This section briefly describes new features in UI Toolkit version 3 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Version 3.7 of the Toolkit is identical in most respects to version 3.5, except that version 3.7 was compiled and linked with Synergy DBL version 5.7. You must use version 3.7 of the Toolkit if you are using DBL version 5.7. Use Toolkit 3.5 with DBL version 5.1 or 5.3.

Features that may break your existing code

We added the following features, which may break your existing code or require that you recompile your code when you update to UI Toolkit version 3.

- Enumerated input fields within a script that do not specify an associated allow list, selection list, or selection window now generate an error when compiled by Script, Proto, or Window Designer. This change may break your existing code. (3.7.8)
- (OpenVMS) If you are using the Toolkit shared image file **tklib_sh.exe**, you must relink your applications with this version. Relinking is necessary due to expansions required to support new features in some of the global data sections. We allocated additional space in these sections to avoid requiring a relink in the future.

  Note also that if you are including **tklib.olb** in your own shared image, the change in data space sizes will force you to relink your application with this version. (3.7.6)
- The arrive and leave methods for a list are now called only once: when the selected item changes. If no change is effected, neither method is called. Special cases include D_LINSERT (which leaves and arrives even though the new item is at the same logical index), and D_LDELETE (which does not leave the item being deleted, but arrives at the new item that took its place). D_LMOVEUP and D_LMOVEDN do not call the arrive or leave methods; the item is the same although its index has changed.

  Note that an attempt to move to a non-existent item will generate a leave and arrive at the same item. Commands that are queued using L_QUEUE and processed together in a single call to L_PROCESS or L_SELECT are treated as one movement. For instance, if you queue a D_LFIND and process a D_LDOWN, there will be only one leave (from the original item) and one arrive (to the item resulting from the D_LDOWN). If you need to arrive at the intervening points, call L_PROCESS for each operation. This may break your existing code if you depend on your method being called at one of the points where we have eliminated it. (3.7.6)
- The L_INPUT subroutine now calls L_PROCESS to update the current item before performing input. This significantly improves the consistency of using L_INPUT, but it also means that you should include the fourth argument (the list data) in order to preserve the list item data. (3.7.4)

- We made it easier to overload Toolkit subroutines. Subroutines with the prefix "TKP_" are now used internally by the Toolkit. You must rename any of your own subroutines that have this prefix to use a different prefix. If you have defined a routine that *replaced* one of the Toolkit's distributed USR_ routines, do *not* rename it, even if it has the prefix "USR_". If you were calling any of the Toolkit's distributed USR_ routines, for example USR_HELP, call them now with the prefix "TKP_". This modification may break your existing code. (3.7.4)
- When U_FINISH is called, all channels opened by U_START or U_OPEN are closed. If opened in output mode, they are purged. The only exception is when a terminal is opened by U_START. In this case, it will remain open after U_FINISH. This change may affect your existing code if you are currently depending on a Toolkit channel to remain open after U_FINISH. (3.5.9)
- The input window ID argument to all Toolkit I_*xxx* calls is now required as documented. In previous versions, if the window ID was omitted, the Toolkit defaulted to the last input window accessed. Given that this was undocumented, and that the resulting confusion would be great were the wrong window accessed, this has been changed. This change may break your existing code if you had discovered and were depending on this undocumented feature. *(*3.5.9)
- When specifying the number of decimal places in I_FLDMOD (D_FLD_DEC), the field type is now automatically converted to D_FIXED. Previously, the decimal places were ignored if the type was not set to D_FIXED. Note that this change may affect your existing code if you were depending upon setting the type to D_DECIMAL to clear the number of decimal places. (3.5.14 and 3.7.1)
- Unformatted fixed-point fields (decimal with precision) now always display their full precision. Previously, the full precision would only be displayed if the field did not contain spaces in those positions. Note that this change may affect your existing code if you were depending on spaces in the contents of the field not to be displayed as 0s to the right of the decimal. (3.5.14 and 3.7.1)
- When drawing an input field's allow list or selection list from the Data Dictionary, any quoted entries (single or double quotes) will be unquoted. This allows you to specify a blank entry in the Data Dictionary. Previous to this version, you could not. Note that this change may affect your existing code if you previously had any quoted entries specified in the Data Dictionary. Previously, these would have included the quotation marks in the allow list or selection window. To achieve the same results, you must now add the opposite quotation marks around the entry. (3.5.15 and 3.7.2)
- In previous versions of the Toolkit, the placement of selection windows associated with input fields was incorrect if the input window was placed at column 1 and had a border. This has been corrected. This change may break your existing code if you were previously compensating for this error.
- In previous versions of the Toolkit, the placement of selection windows associated with input fields was one row higher and one column to the left of where it should have been if the input window had a moveable view. This has been corrected. This change may break your existing code if you were previously compensating for this error.

- Window libraries are now stored in a different format. If you are upgrading to version 3.7 or are upgrading from a version prior to 3.5, you must recompile and relink your source code and rebuild your window libraries with Script or Proto after you install this version of the Toolkit. (3.5)
- To avoid conflicts with common fields, we changed the global data section names throughout the Toolkit to begin with "DTK_". The Toolkit also uses names that begin with "GLST_" for common field names. Do not use names beginning with "DTK_" or "GLST_" for common fields or global data sections. This change requires that you recompile all modules which include **tools.def** or any other Toolkit include file which declares a global data section. (3.5)
- This version of the Toolkit increases the maximum number of channels logged with the Toolkit environment from 98 to 255. This modification requires you to recompile and relink your code. Also, you should change any channel number fields that you have declared as **d2** to **i2**, **i4**, **d3**, or some other format capable of storing the maximum 255. (3.5)
- This version of the Toolkit increases the maximum number of input fields in an input window or input set from 98 to 253. This modification requires you to recompile and relink your code and to rebuild your window libraries with Script. Also, many of the fields in the distributed include files have been changed to integer type to facilitate this modification. Please inspect your code to ensure that you are not depending on a decimal storage and that you allow for the new maximum. (3.5)
- U_START will now close through channel 255, unless you pass a new optional argument that specifies the last channel Toolkit may use. (3.5)
- To improve performance, the list processing subroutines (L_*xxx* and USR_L*xxx*) have been converted to use 4-byte quad-aligned integers for their controls. Changes were also made to **tools.def**, **lstprv.def**, **lstprv.dbl**, and **lstusr.dbl**. No changes should be necessary, as long as you use these subroutines as they have been documented. (3.5)
- In previous versions of the Toolkit, the renditions for centered or right-justified text in a .WINDOW script were applied inconsistently. If a margin existed, any blanks to the left or right of the justified text received the renditions of the margin; otherwise, blanks to the left received the renditions for the left-most justified character, and blanks to the right had no renditions. This has been changed to operate as follows: If the renditions are specified on a previous line of the script, they apply to the entire line except the margins. If they are specified on the same script line as the justified text, they apply only to the justified text, and the blanks on either side receive any previously specified renditions (either from a previous script line or from a .FILL command). This change may break your existing code if your existing scripts depend upon the inconsistent behavior of previous versions. (3.3.2)

New features

Window scripts

- The DRAGBAR and NOCELL qualifiers of the .BORDER command are now supported within selection windows.
- We added support for check boxes and radio buttons. (3.7.7)
- The .LISTCLASS script command now accepts a continuation character with the arguments to the headers or footers qualifiers. (3.7.6)
- We added a new drill method for input fields. (3.7.5)
- You can now specify an input time-out (a "wait") within the Toolkit input routines. You can specify the wait at the field level or at a global level. (3.7.4)
- We added a new qualifier to the script .BORDER command, which allows you to designate that a window should have a border only if the border does not require a full character cell for display. The new qualifier is NOCELL, which takes the place of **on** or **off** (and is mutually exclusive of them). If you use NOCELL, the window will have a border in Windows environments, but not in any other supported environments at this time. For windows created at runtime, use the new WB_NOCELL qualifier in the W_BRDR subroutines. (3.5.14 and 3.7.1)
- In previous versions, the .FIELD script statement's SELECT qualifier did not allow a 0 for the window height, even though the height is optional. We changed this so that a 0 for the height is the same as not specifying the height. (3.5.9)
- In a window script, you may now use "\." to signify a single period (".") in the text. Previously, you couldn't have a leading period in a line of text in a window. (3.5.9)
- (Windows) We added a new qualifier to the .BORDER script command to specify whether or not a window should have a drag bar on Windows. (3.5.3)
- The .DICTIONARY script command can now be continued on more than one physical line. (3.5)
- We added a new optional qualifier to the .DICTIONARY script command and the IB_DICTIONARY subroutine that allows you to include the same structure more than once within an input window, if you give each subsequent incidence of the structure a unique *alias_name*. (3.5)
- The maximum number of elements in a .STRUCTURE statement has been expanded to 507. (3.5)
- The .ENTRY menu script command can now be continued on more than one physical line. (3.3.2)
- Script now allows windows with a display area of up to 50 x 255 to have a .PLACEMENT command. (3.3.2)
- We added a new optional qualifier to the .PLACEMENT script command, UNBOUNDED, which allows you to specify absolute window placement, which is not bounded by the Toolkit's screen body. (3.3.2)

- Script processing and building of input windows on the fly have been significantly optimized for DBL version 5. (3.3)
- The .LIST menu script command can now be continued on more than one physical line. (3.1.2)

Data Dictionary interface

- The Toolkit now supports the new Synergy Data Dictionary format justification specification. (3.5.5)
- The interface to the Synergy Data Dictionary now supports the AMPM and NOW options on time fields. (3.3)

Utility programs: Script, Scridl, and Proto

- If a selection window does not contain any selectable entries, a "No entries in the selection window" error is now generated. (3.7.6)
- Script now issues a warning when tokens are truncated. Token length has always been limited to 80 characters, but previously Script would truncate the token without a warning. (3.5.14 and 3.7.1)
- In previous versions, both Script and IB_FIELD would allow the enumerated qualifier on non-numeric fields. This was unintentional and had undesirable results at runtime. This has been corrected to issue the error message "Field must be numeric" when the enumerated qualifier is encountered on a non-numeric field at window building time (in Script or IB_FIELD). (3.5.10)
- Script, Proto, and Window Designer now support script file physical lines of up to 255 characters in length. Previously, the limit was 132. (3.5.9)
- The **Scridl** utility now outputs the dictionary filenames to the script only if they have overridden the defaults. (3.5.5)
- If the Script program encounters errors, it will now exit with a status of D_EXIT_FAILURE (2 on OpenVMS, 1 in other environments). Otherwise, it will exit with a status of D_EXIT_SUCCESS. (3.5.4)
- If you're using the **Scridl** program to create an ICS Definition Language (IDL) file to load into a version 3.4 or higher Synergy Data Dictionary, please note the following: when the "Load dictionary schema" utility prompts you to merge the schema, answer "Yes." You will then be prompted for how to handle new and existing definitions found in the IDL file. Select "Yes" at the "Add new?" prompt, and select "Overlay" at the "Change existing?" prompt. (3.5.2)
- You must now have a development license (PSDE or PSW) to compile a script file from the stand-alone Script compiler. (3.5)
- To comply with the Synergy Application Standard (SAS), Proto has been changed to use F4 for "Exit," and F2 for "Directory." (3.3.3)
- **Scridl**, the Script-to-IDL Conversion utility, now supports the AMPM and NOW options on time fields. (3.3)

Environments

- The E_METHOD subroutine enables you to register methods for environment-level events. This feature does nothing in version 3.5 of the Toolkit. (3.7 only)
- (Windows) A new option has been added to the E_SECT subroutine. D_CAPTION enables you to address the application window caption when running in a Windows environment. (3.5)
- The number of available environment levels is now modifiable through U_START. (3.3)

Input windows

- We added a new subroutine, I_FRAMES, which allows you to show the frames on input fields without processing I_INPUT. This routine does nothing in UNIX or OpenVMS environments. (3.7.9)
- We added two new menu entries, I_NEXTCTL and I_PREVCTL, as well as two new I_NEXT field qualifiers, *NEXTCTL* and *PREVCTL*. These access the next and previous controls, including buttons. (3.7.9)
- The force buffer is now cleared when a field exits without pending input. As a result, I_FORCE now returns only the pending input for the last field accessed. (3.7.8)
- We added a hyperlink prompt method for input fields. (3.7.7)
- We added two new input routines, I_ENABLE and I_DISABLE, which can enable and disable one or more fields in a given input window. (3.7.6)
- The window library argument is optional in the following Toolkit routines: U_LDWND, I_LDINP, M_LDCOL, U_POPUP, and U_FLASH. If the window library is not passed, **g_utlib** will be used. (3.7.6)
- You can now change static prompts at runtime through I_PROMPT. (3.7.6)
- We added a leave method for input fields. (3.7.6)
- You can now call the I_INPFLD and I_INPUT subroutines recursively. (3.7.6)
- We added a new drill method for input fields. (3.7.5)
- The E_MODE menu entry now toggles between “insert” and “overstrike” modes in single-line input fields. (3.7.5)
- We added the ability to include standard “OK,” “Cancel,” and “Help” buttons in input windows when running in a Windows environment. (3.5.11)
- In previous versions, two types of problems were experienced with input fields whose display length was shorter than the potential input length (via a format). First, clearing of the field only encompasses the length of the display, leaving stray characters in the window after input. These characters could also be interpreted by the input processor as part of the next input for that field. This has been corrected to clear the wider of the input area and the display area. Second, any justification of the display was performed within the bounds of the larger of the input or display lengths. This has been corrected to justify only within the length of the display (i.e., length of the format string if present, length of the default format if not). *To enable this latter correction, you must recompile any affected window scripts*. (3.5.10)

- In previous releases of version 3.5 of the Toolkit, if an input window contained more than 100 input fields and some of the fields after the 99th field had associated selection windows, these windows would be attached to the wrong fields. This has been corrected. *If you were experiencing this problem, you must recompile the script for those windows*. (3.5.9)
- In previous versions, when **g_date_order** was set to 2 (indicating *YYMMDD* date ordering for entry and display), period dates were nevertheless required to be entered period first, then year. This has been changed to interpret the input as year first, then period if **g_date_order** is 2. (3.5.9)
- We added a new global flag in **tkctl.def** named **g_plc_col_args**. If this flag is true, columns whose IDs are passed as optional arguments to the various input routines are placed and removed by those routines. If this flag is false, these columns are not placed and removed by those routines. This feature is useful in the Windows environment, where such columns usually define keys that are automatically supported as part of the controls in this environment, and the excessive placing and removing of columns causes the menu bar to "flicker." Thus, **g_plc_col_args** is set by U_START to false in Synergy for Windows, and true in all other environments. You may alter its value at any time thereafter. This flag affects the following subroutines:

  I_INPUT
  I_INPUT_P
  I_INPFLD
  I_INPFLD_P
  L_INPUT
  L_INPUT_P
  L_INPFLD
  L_INPFLD_P
  L_SELECT
  L_SELECT_P
  S_SELECT
  S_SELECT_P
  T_EDIT
  T_EDIT_P
  T_VIEW
  T_VIEW_P
  %U_MSGBOX
  %U_MSGBOX_P

  See the documentation for one of these subroutines for more information. (3.5.14 and 3.7.1)
- Input window names may now be up to 15 characters long. In previous versions, they were limited to 10 characters. (3.5.9)

- To facilitate redirection of input, the Toolkit now allows you to configure the number of successive end-of-file characters to allow on input. The new field **g_eof_max**, which is set at 100 by default, specifies this maximum. After **g_eof_max** successive EOF characters (^D on UNIX, ^Z on OpenVMS) have been encountered, the program will stop. Note that you can set this field to 0 to disable this behavior. (3.5.8)

  A message also displays on the screen when too many EOF characters are encountered. (3.5.9)
- When calling IB_INPUT to create an input window at runtime, you can now pass a null string (" ") for the window name, and a unique name will be assigned, just as in W_PROC. (3.5.5)
- I_INPUT(_P) and I_NEXT (using *MENU*) now support a new menu entry, "I_FRSTCLR", which positions the input context to the first empty field in the input set. Although this has always been the default behavior if no input context has been established for the set, "I_FRSTCLR" allows you to undo a previously established context. (3.5.4)
- You can now terminate Toolkit field input normally with activation characters other than ENTER, using the new routine I_ACTIVATE. To use this feature successfully, you must be running Synergy DBL version 5.3.10 or higher on OpenVMS, or 5.1.15 or higher on other platforms. (3.5.4)
- We added a new subroutine, I_DSPAREA, which allows you to set the display area for an input window. (3.5)
- A new special set name, *CURR*, has been added to I_NEXT. When this is specified, I_NEXT will use the currently loaded set. (3.5)
- I_NEXT also has three new special field name settings. *MENU* sets the context based on the contents of **g_entnam**, *NMBR*, *field#* sets the context to *field#*, and *NONE* sets the context to no field. (3.5)
- A new .FIELD qualifier allows you to specify an arrive method for an input field. This qualifier will contain the name of a subroutine that the Toolkit will call before processing the field. The .FIELD script command and the I_FLDMOD and IB_FIELD subroutines enable you to specify this qualifier. (3.5)
- You can now pass up to 20 additional optional arguments to I_INPUT or I_INPUT_P. These arguments, if passed, are in turn passed to any method subroutine. (3.5)
- A new function, %I_GETSTRING, enables you to retrieve strings referenced by an input window's internal pointers. (3.5)
- We added four new functions to retrieve information about an input field's structure position, size, dimension, and precision. (3.5)
- You can now modify the display format of a field at runtime using I_FLDMOD and IB_FIELD subroutines. (3.5)
- Integer input fields are now supported in the .FIELD script command and the I_FLDMOD or IB_FIELD. (3.5)
- I_FLDMOD can now be used to increase the length of an input field as long as the new length does not exceed the length of the field when it was originally created. (3.5)

- We added a new subroutine, I_SETINF, which allows you to retrieve information about an input set. (3.5)
- Script processing and building of input windows on-the-fly have been significantly optimized for DBL version 5. (3.3)
- A new subroutine, I_TXTPOS, enables you to set the initial position in a text field for subsequent input. (3.3)
- A new subroutine, I_SNAPSHOT, has been added to save or restore the controls for an input window. (3.3)
- We added a new field qualifier, to specify that a negative value or 0 may be entered in the field. In a script file, the new .FIELD qualifier is negative(orzero). In I_FLDMOD and IB_FIELD, it is D_FLD_NEGORZERO. (3.3)
- I_LDINP now allows you to rename a window when loading, enabling you to create multiple instances of the same window. (3.3)
- The optional *field_id* and *selection_id* arguments to the IB_END subroutine are no longer needed and are ignored. (3.3)
- We added the ability to specify whether you want to retain the position within a text field. (3.3)

List processing

- We increased the allowable search data size from 80 to 256 to allow for longer searches within L_FINDSPEC and D_LFIND in L_PROCESS and L_SELECT. (3.7.8)
- We added a new list method for handling mouse double-click events for Windows applications. (3.5.14 and 3.7.1)
- We added two new qualifier arguments to L_STATUS: D_LHDRLNS and D_FTRLNS, which return in the next argument the number of header and footer lines, respectively, in the list. (3.5.14 and 3.7.1)
- We added the ability to specify that a list should have scroll bars in a Windows environment but not under other environments. Two new options string qualifiers, NOCELLHBAR and NOCELLVBAR have been added. If these qualifiers are included in the options string of the list class, they will suppress horizontal or vertical scroll bars, respectively, on UNIX and OpenVMS platforms, while showing them as needed on Windows. (3.5.11)
- You can now perform a wrap-around find in list processing. A new optional ninth argument to L_FINDSPEC specifies whether or not a search should wrap. (3.5.7)
- Subroutines L_SELECT and L_SELECT_P now display a message on the information line while a find (S_FIND) or find next (S_FINDNEXT) is being performed. This message is contained in the text file (**DBLDIR:syntxt**) under facility DTK, mnemonic FIND_MSG. (3.5.7)

- We added three new subroutines for performing input within lists: L_CHR, L_INPFLD, and L_INPUT. These subroutines are required for the Windows environment, but are also compatible in other (UNIX and OpenVMS) environments. (3.5.2)
- We added "arrive" and "leave" method support for list processing. (3.5.2)
- You can now create list classes at runtime with the new subroutine L_CLASS. (3.5)
- You can now specify the name of a subroutine for loading items into a list. This can be done using the .LISTCLASS script command or the L_CLASS or L_METHOD subroutines. (3.5)
- We added three new information arguments to L_STATUS: D_LDISITEM, D_LONENABLED, and D_LHEIGHT. (3.5)
- We added optional arguments to subroutines L_SELECT and L_SELECT_P, which enable you to override the default "Find" window. (3.5)
- We added a new function, %L_FINDWND, which creates a "Find" window for L_SELECT based upon a field in the list's input window. (3.5)
- We added a new optional argument to subroutines L_SELECT and L_SELECT_P, which allows specification of the name of a subroutine to call before character input is performed. (3.3.2)
- A new subroutine, L_RESTART, has been added that enables you to restart a list from scratch, without deleting and recreating it. (3.2.1)
- A new subroutine, L_DATA, has been added that enables you to manipulate the data in the list without changing your current list context. (3.2.1)
- You can now use the request code D_LRESTORE as a synonym for D_LCANCEL in L_PROCESS. (3.2.1)
- We added two new request codes to L_PROCESS to extend a previously closed end of the list using the demand-loading mechanism: D_LEXTENDTOP and D_LEXTENDBOT. (3.2.1)

Menu subroutines

- We added the ability to signal a menu entry from within a method routine, even if that routine was invoked by the Synergy DBL runtime. A new routine, M_SIGNAL, takes only one argument: the menu entry name to signal. It can be called as a function or subroutine. The return value is true if the entry name is non-blank; otherwise, it is false. (3.7.4)
- The window library argument is optional in the following Toolkit routines: U_LDWND, I_LDINP, M_LDCOL, U_POPUP, and U_FLASH. If the window library is not passed, **g_utlib** will be used. (3.7.6)
- A new field in **tools.def**, **g_entusr**, contains any user text for the last menu entry selected. If the entry had no user text associated with it, **g_entusr** is blank. The **g_entusr** field also works for submenus. Previously, there was no way to access user text associated with a submenu entry. (3.5.9)

- You can now disable or enable submenu entries with M_DISABLE and M_ENABLE respectively. (3.3)
- M_LDCOL now allows you to rename a column when loading, enabling you to create multiple instances of the same column. (3.3)

Selection windows

- A new argument to S_SELBLD enables you to suppress the blank column that occurs to the left and right of each entry in the selection window. (3.5)
- A new optional qualifier, NOPAD, has been added to the .SELECT script command, enabling you to suppress the blank column that occurs to the left and right of each entry in the selection window. (3.5)
- We added two new subroutines for processing selection windows as check boxes: S_SELECTCB and S_UPDATECB. (3.5)
- When using S_SELLD to reload the entries in a selection window, you can now specify which quick-select characters to use. (3.5)

Text processing

- The text in the information line is now shown while a text window is being edited. The direction and mode messages are displayed right-justified in the information line with a vertical bar before each message. (3.7.6)
- A new optional argument has been added to the T_VIEW and T_VIEW_P subroutines, which returns the key pressed. (3.3.5)

Utility subroutines

- (Windows) We added the ability for your Toolkit application on Windows to respond to window mouse events. These events include mouse clicks, double-clicks, closing a window, and moving a window. (3.7.7)
- (Windows) We added the function %U_WNDSTYLE, which enables you to change the vertical spacing on a window-by-window and list-by-list basis. (3.7.7)
- The window library argument is optional in the following Toolkit routines: U_LDWND, I_LDINP, M_LDCOL, U_POPUP, and U_FLASH. If the window library is not passed, g_utlib will be used. (3.7.6)
- (Windows) We added a new function, %U_PRINTQUERY, for retrieving information about the currently selected printer from Print Manager. (3.7.6)
- (Windows) We added the %U_PRINTSETUP function, which allows access to the Windows Print Setup standard dialog box under Windows. (3.7.5)
- We added three new routines to facilitate checking version numbers on products: U_MINVERSION, U_CHECKVERSION, and U_PARSEVERSION. (3.7.4)

- We added a new function, %U_WINHELP, which pops up a Windows help file in the Windows environment. (3.5.14 and 3.7.1)
- We added a new function, %U_GETFILENAME, which prompts the user for a filename, with optional browse capability. In a Windows environment, this routine invokes the standard file "Open" or "Save as" common dialog box. In non-graphical environments, the user is prompted for a filename, and a list of files in the specified path can optionally be browsed. (3.5.14 and 3.7.1)
- (Windows) We added a new function, %U_ICON, which will set the icon for the application or a window in a Windows environment.
- U_SAVE now has an optional fourth argument, which is returned with 0 if the save operation was successful, or with the DBL error number if not. Previously, all errors in this routine were fatal. (3.5.10)
- Starting with version 3.5.4 of the Toolkit, U_START set all environment parameters to their default states. This has been changed to detect and record the current states when U_START is called. (3.5.9)
- U_ABORT will now stop with an exit status of D_EXIT_FAILURE (2 on OpenVMS, 1 in other environments). (3.5.4)
- We added a new function, %U_MSGBOX, which displays a Windows-style message box with various options. (3.5.1)
- We added a new subroutine, U_BEEP, which rings the terminal bell. (3.5)
- In addition to its ability to split long message lines, U_MESSAGE now also recognizes the two-character sequence \n as indicating a forced new line break. (3.3.2)
- A new subroutine, U_RESIZE, enables you to resize the screen. (3.3)
- A new subroutine, U_ABOUT, enables you to display an "About" help window for your application. (3.3)
- A new argument to the U_START subroutine enables you to set the number of available environment levels. (3.3)
- An optional argument has been added to U_START, which specifies the first channel number available to the Toolkit. (3.3)
- Since there is no window memory pool size in Synergy DBL version 5, the *pool_size* argument to U_START is now ignored. (3.3)
- U_LDWND now allows you to rename the window when loading, thus enabling you to create multiple instances of the same window. (3.3)

Help processing

- The distributed version of the USR_HELP subroutine now responds to a second selection of the "O_HELP" menu entry after the help window has been displayed. In this case, a second help window named "h_help" will be displayed if found. Exiting from this window will bring the user back to the initial help window. In previous versions, "O_HELP" while in help would return "O_HELP" to the calling routine. (3.5.12)

Customizing

- We added keymap sequences for Function Keys 13 through 20 for ANSI keyboards. They now function as Shift F3 through Shift F10. (3.7.5)
- We made it much easier for you to overload the USR_*xxx* subroutines with your own customized subroutines. You now also have the option of naming your own subroutines. (Previously, you had to use the same name as the Toolkit-supplied USR_ subroutine.) (3.7.4)
- (OpenVMS) We made it easier to rebuild the **tklib_sh.exe** shared image:
  - We collected the data segments in front of the image, because they change in size less frequently than the code segments.
  - We expanded many of the data segments to allow for future growth without changing the positions of other segments.
  - We changed the **build_tklib_sh.com** file to link against older versions of the OpenVMS shared executables, to enable compatibility with versions of OpenVMS back to v5.5.
  - We included the **ddlib.olb** routines (DD_*xxx*) in the Toolkit shared image.

  For all of these reasons, you will want to use our **build_tklib_sh.com** if you need to rebuild the Toolkit shared image. However, you may find that you no longer need to rebuild the shared image because of these changes, in conjunction with the new USR overloading capability. These modifications will also make it possible for us to distribute new versions of the Toolkit without requiring you to relink your applications against the shared image. Be aware, however, that when upgrading to version 3.7.4, you are required to relink your applications if you are using the shared image. (3.7.4)
- We added two new key maps to the distributed key map file (**dtkmap.ism**): XTERM (for X-windows terminals) and WY60 (for Wyse-60 and compatibles). (3.7.4)
- We added another "user hook" subroutine, USR_EDTDSP, which enables you to specify the display of a user-type field when it is being edited. (3.5)
- We added a new return status value for the USR_CHKFLD subroutine: D_EMITTEDERR. (3.5)

- Text messages that were previously stored in **dbltxt.ism** have now been moved into the Synergy text file **syntxt.ism**. The text message format has changed also. The **Severity Code** and **Runtime Error #** fields have been removed and the record size increased. To modify Toolkit text messages, you must use the Synergy Control Panel distributed with Synergy DBL. (3.3)
- We added a new terminal key map to the distributed version of **dtkmap.ism**: GENERIC. GENERIC uses ^K, ^J, ^H, ^L, and ^E for the Up, Down, Left, Right, and F2 functions, respectively. This simplifies the process of setting up a key map for a terminal that does not emit ANSI sequences for the arrow keys. (3.2.2)

File-stack subroutines

- You can now specify the default file stack cache size with the environment variable DTKFSWINSIZ. This size specifies the amount of memory allocated by the file stack if a larger size has not been specified to FS_INIT, and if a larger size is not needed by the file stack routines. Any value specified will be rounded up to the nearest multiple of 8192. The default size is 8192 if no value is specified here. Increasing the size may result in improved performance for applications which traverse a large number of records on the file stack (for example, large lists). (3.5.3)
- We added a new argument to FS_INIT so you can specify the amount of memory allocated by the file stack. (3.5)

Miscellaneous

- An EXAMPLES subdirectory contains sample code for Toolkit applications. (3.7.6)
- The error message limit of 80 characters has been extended to avoid the possibility that messages will be clipped. (3.7.6)
- You can now call the Toolkit window debugger from the Synergy DBL debugger prompt (Synergy DBL 5.7.5 and greater only). At the debug prompt, type “wndbg” to invoke the Toolkit window debugger. If the Toolkit is not initialized (that is, before U_START or after U_FINISH), the command simply returns without doing anything. If the program was not linked with the Toolkit, a “Cannot access external routine” error is generated. Otherwise, the Toolkit debugger takes over input until a “quit” command is issued. At that point, the Synergy DBL debugger resumes control with a new prompt. Note that on OpenVMS systems, if your application is linked against the Toolkit shared image (**tklib_sh.exe**), you must invoke **openelb** for the shared image in your application before this feature is enabled. Thus, the statement

  ```
  xcall openelb(tklib_sh)
  ```

  where *tklib_sh* is the name of a logical symbol assigned to the Toolkit’s shared image file, must be executed prior to using this feature on OpenVMS, unless you directly link in **tklib.olb**. We do not perform this OPENELB automatically for performance reasons, as it slows every XSUBR call. You should only include it if needed. (3.7.5)

- The file **chninf.def** is not included in the Toolkit distribution. This file contains the MAX_CHN_DATA and MAX_CHN_INF definitions, which may be modified to change the size of the channel information memory area. These definitions were previously held in the file **chninf.dbl**; however, this file is still included in the distribution so that changes to the definitions can be compiled into the Toolkit library. (3.7.4)
- In previous releases of Toolkit 3.5, the interrupt key (usually ^C) did not always work when enabled. We corrected this for the Toolkit, but it also requires DBL 5.1.17 or greater. (3.5.9)
- We added an online quick-reference file called **QREF_DTK.TXT**. You can find it in the Toolkit directory. It lists the syntax for all Toolkit subroutines. This file is a simple text file that enables you to perform quick word searches. You can modify it to make your own customized quick-reference card.
- Toolkit source code has been modified to use the ANS DIBOL form of the IF-THEN-ELSE statement. It has also been modified to declare all numeric subroutine arguments as type "N". This means that the compiler switches "**-aN**" are no longer needed to compile distributed source files. All **.def** files have also been modified to allow inclusion within multiple subroutines separated by .END. (3.5)

4

Composer

Version 9

This section briefly describes new features in Composer version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- We added support for **d28** and **d28.28** fields. (9.3)
- We now support precision (up to 28 places) on values for the Range, Default, and Selections properties when the field has precision. (9.3)
- We added a Find function to Object Manager's menu, which enables you to search for a particular object in Object Manager. This is especially useful when there are many objects and you don't know which script an object is in. CTRL+F also works when Object Manager is focused. Find options are saved in **composer.ini**. Uncompiled nodes are never searched. Search is always forward, circular, and case-insensitive. (9.1.3)
- We changed Composer's help system to use **composer.chm** (compiled HTML help) instead of **composer.hlp** (WinHelp). This will be compatible with Windows Vista. (9.1.1b)

Version 8

This section briefly describes new features in Composer version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Composer version 8.

- Composer no longer supports Workbench projects, which means Composer and Workbench can no longer share a project file. (Composer still supports its own **.psc** project format.) Composer can still be launched from within Workbench for the purposes of editing a script file. (8.3)

New features

- We have simplified the Compile Scripts feature of Composer. To specify which scripts are to be compiled, the name of the window library into which they are to be compiled, and whether the window library is to be overwritten or appended to if it already exists, select File > Compile Scripts Setup. You can also compile the selected scripts from this dialog. Later, you can compile scripts using these options by selecting File > Compile Scripts. (8.3)
- We added the ability to include individual elements of an array defined in the Repository as an input field in an input window. (8.3)
- We updated the Palette Entry dialog to include Windows system colors. Additionally, we made the dialog box easier to use by making it apply only to the color palette entry that is selected in the Rendition window when you click the Change Colors button. The name for the dialog box is now Palette Entry #, where # is the number of the palette entry that will be affected by changes. (8.3)
- To eliminate project file synchronization issues between Workbench and Composer, we now require that when using these products together, Composer be launched from within Workbench. Workbench can no longer be launched from Composer. (8.1)

- Synergy/DE now supports the concept of a "system-specific" initialization file (**synergy.ini**) versus a "user-specific" initialization file (**synuser.ini**). Settings in the user-specific file override settings in the system-specific file. (For more information on this, refer to the "Environment Variables" chapter of *Environment Variables and System Options*.) The **synuser.ini** file is created (and searched for) in the Synergex subdirectory of your local application data directory. To determine where the **synuser.ini** file is located on your particular system, you can run the **synckini** utility. Refer to the "General Utilities" chapter of *Synergy Language Tools* for details.

  As the above enhancement is a runtime-level change, it affects settings used by Composer such as colors, fonts, APP_HEIGHT, APP_WIDTH, etc. One explicit change made to Composer related to this new feature is the following: when colors or fonts are updated using Composer, these settings are written to the [colors] and [fonts] sections, respectively, of **synuser.ini**, not **synergy.ini**. (8.1)

- Composer now treats **composer.ini** as a "user-specific" initialization file. This file is created (and searched for) in the Synergex subdirectory of your local application data directory. To determine where the **composer.ini** file is located on your particular system, you can run the **synckini** utility. Refer to the "General Utilities" chapter of *Synergy Language Tools* for details. The specific effects that this change has on Composer are as follows:

  - As a result of this change, the Synergy/DE installation no longer installs a **composer.ini** file nor creates a **composer_old.ini** copy of the previous version. The **composer.ini** file is created by Composer the first time you run it following installation. Because it is a "created" file, it is not removed on an uninstall or upgrade.
  - Composer no longer uses SFWINIPATH to locate **composer.ini**. SFWINIPATH is reserved for use in locating "system-specific" initialization files such as **synergy.ini**.
  - If you have a **composer.ini** file from a previous version that you want to continue using, run **synckini** after installing this version. Use the output from **synckini** to determine the location of **synuser.ini**. Then, move your **composer.ini** file to the same directory. Be sure to remove the following three lines from the [TOOL_BUTTONS] section:

    ```
    TOOLTIP4=Workbench
    BITMAP4=%synergyde%\workbench\wkbtlbr.bmp
    CMDLINE4=%synergyde%\workbench\win\vs.exe
    ```

  (8.1)

- Composer now supports the new UI Toolkit field properties, input length, display length, and view length, for both local and repository fields. If specified, these values represent overrides to Toolkit's default computations for each. (8.1)

- To support the new field properties above, we changed the behavior of visually resizing fields. Visually resizing a field affects the view length only. It does not affect field size or enumerated length. Additionally, if a view length has been specified for a field, modifying its size, decimal places, negative allowed, or enumerated length property does not affect the visual size of the field. (8.1)
- Before saving, Composer now creates a backup of any script file it is saving over, and then it deletes the backup once the save is successful. If for any reason Composer or the system aborts in the middle of a save over an existing script file, the original script file will have been renamed to the same name with a **.bak** extension. If a **.bak** file already exists, it is saved as **.bk0**, **.bk1**, etc., up to **.bk9**. (8.1)
- We changed Composer's start-up processing to test whether Workbench is installed. If Workbench is not installed, drilldown buttons are omitted from the method properties in the Properties window. (8.1)

Version 7

This section briefly describes new features in Composer version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Composer version 7.

- The Synergy/DE Client installation does not create/update the **composer.ini** file. If a **composer.ini** file does not already exist, Composer creates one the first time you run it following installation. This Composer-created **.ini** file will not contain the toolbar button or menu entry definitions for launching other Synergy/DE tools. To restore these definitions, add the following lines to the top of your **composer.ini** file. You should add the last three lines only if Workbench is installed. (7.5)

```
[TOOL_BUTTONS]
TOOLTIP1=Repository
CMDLINE1=dbr RPS:rps
BITMAP1=0,%synergyde%\composer\rps.bmp
TOOLTIP2=Synergy Control Panel
CMDLINE2=dbr DBLDIR:synctl
BITMAP2=0,%synergyde%\composer\synctl.bmp
TOOLTIP3=UI Toolkit Script Compiler
CMDLINE3=dbr WND:script
BITMAP3=0,%synergyde%\composer\script.bmp
TOOLTIP4=Workbench
BITMAP4=0,%synergyde%\workbench\wkbtlbr.bmp
CMDLINE4=%synergyde%\workbench\win\vs.exe
```

New features

- We added a drilldown button for the Method property for a button object. When pressed, the button's method name is generated (if blank) and Workbench is launched (if Workbench is installed). A corresponding button method template is inserted into Workbench's editor, similar to the input field methods (arrive, leave, etc.). (7.3)
- We added "Compile All Scripts" to the File menu. This enables you to compile all script files within a project in a single window library. (7.1)
- The Open and Save As dialogs default to the last directory accessed. (7.1)
- Clicking on an input field's drilldown button inserts a corresponding code template into Workbench's visual editor. (7.1)

- We added read-only and disabled properties for input fields. (7.1)
- We added the View submenu to the context menu for objects displayed in the Application window. (7.1)
- Composer now launches and shares its project file with Workbench. (7.1)

Version 6

This section briefly describes new features in Composer version 6 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- We added the ability to compile scripts within Composer. There is now a menu entry “Compile Script...” on the File menu. This will allow the current script to be compiled in a specified library. Any errors are displayed to an error window as well as an error file. (6.3)
- Composer will now automatically set an input window’s “Button/Border” property based on where in the window the mouse is clicked when creating the first button. The border closest to the initial mouse click is chosen (right or bottom). In previous releases of Composer, you had to manually set this property. (6.3)
- We changed the way you move fields and prompts with the mouse. Now, if you simply click and drag on either the field or the prompt, you will move both. If you hold the ALT key down while you click, you will move only the one you clicked on. “Move” from the “Object” menu still moves both together, and both still move together if they are container in an object group that is being moved. (6.3)

5

Workbench

Version 9

This section briefly describes new features in Workbench version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- Workbench now supports the environment variable SYNCSCOPT for Synergy/DE .NET Component projects. (9.1.3)
- Workbench is now compatible with the new SMC file structure. (9.1.3)

Installation and project conversion

- Workbench 9.5 uses SlickEdit 2010 (version 15). All pre-9.3 projects must be converted. This conversion will occur automatically when a project or workspace is opened for the first time in version 9.5, assuming the files are writable.

  Once a project is converted, it is no longer compatible with earlier versions of Workbench. We recommend that you save a copy of your original project files until you are sure everything is working correctly. (9.5)
- We have upgraded to version 14 of SlickEdit. (9.3)
- You no longer need to provide a SlickEdit key to license Workbench. It is now pre-licensed. (9.3)
- When a project is opened for the first time in version 9.3, that project is automatically converted. (Note that your **.vpj** and **.sde** files must be writable in order for this project conversion to take place; if they are read-only due to version control locks or other reasons, an error will be generated and the workspace will be closed. We cannot close a workspace when you use the “Insert Project into Workspace” feature, so in that case, if the project is not writable, it will not be converted. After the files are made writable, the project will be converted on the next open.)

  Project conversion adds the new Prototype, Link, and Synergy/DE Options entries to the menu for this project. The new 9.3 configurations are not added, and any values currently stored in a command are left as is. (9.3)
- The syn_ver value within Synergy/DE project files is now 6. (9.3)
- We have upgraded to SlickEdit version 12.0.3. (9.1.3)

- Starting in version 8.3, on a new installation, the Synergy/DE and Synergy/DE Client installation programs no longer set the VSLICKCONFIG environment variable. When VSLICKCONFIG is not set, SlickEdit creates its own directory based on the logged-in user (see below). Additionally, we no longer require that each client of a shared installation or each user of a terminal services installation create a directory on the client machine, copy configuration files to it, and set the VSLICKCONFIG environment variable. Starting Workbench runs **update_synergy**, which initializes Workbench and creates the necessary configuration files in the SlickEdit configuration directory. (9.1)
- Workbench 9 uses SlickEdit version 11 and thus requires that all projects be converted. This happens automatically when a project or workspace is opened for the first time in version 9. Once a project is converted, it is no longer compatible with earlier versions of Workbench. We recommend that you save a copy of your original project files until you are sure everything is working correctly. (9.1)
- Starting in version 8.3, on an upgrade installation, the Synergy/DE and Synergy/DE Client installation programs retain any previous VSLICKCONFIG setting. After you upgrade, we strongly recommend that you run the **wbupgrade.exe** program. (If you ran **wbupgrade.exe** when you upgraded to version 8.3, you do not need to run it again.) (9.1)
- IMPORTANT: You do not need to upgrade Synergy/DE Clients for version 9. Only the shared installation needs to be upgraded. If you did not run **wbupgrade.exe** when you upgraded to version 8.3, and you do not run it after upgrading the shared installation, the clients will continue to use VSLICKCONFIG. (We assume that VSLICKCONFIG is set at the user level pointing to a directory on the local machine, and set at the system level pointing to the wbconfig directory on the shared machine.) The same is true for a Workbench installation on a Terminal Services machine. Once Workbench has been upgraded, the Terminal Services users will continue to use VSLICKCONFIG. (We assume that VSLICKCONFIG is set at the user level pointing to a user-specific directory on the Terminal Services machine, and set at the system level pointing to the wbconfig directory.)

  Although local installations, Synergy/DE Clients, and installations of Workbench on a Terminal Services machine will continue to run with VSLICKCONFIG set, *we strongly recommend that each Workbench user run the **wbupgrade.exe** program* to eliminate the use of VSLICKCONFIG from here on out. (See entry below.) (9.1)
- IMPORTANT: After upgrading a local or shared installation of Workbench (including one on a Terminal Services machine), each user should run the **wbupgrade.exe** program located in the \workbench\win directory. (If you ran **wbupgrade.exe** when you upgraded to version 8.3, you do not need to run it again.) This program creates the SlickEdit user-specific configuration directory, copies all files from the current VSLICKCONFIG directory to it, and then deletes the system and user-level VSLICKCONFIG settings. Note: If you are upgrading an installation on a Terminal Services machine, you must reboot after the administrator has run the **wbupgrade.exe** program. This ensures that the deletion of the system-level VSLICKCONFIG setting is recognized by all users. (9.1)

- The **wbupgrade.exe** program also deletes any **dbltags.vtg** file (from an earlier version of Workbench) found in the SlickEdit configuration directory. This file gets recreated the first time you request tagging. The reason we do not want to use a pre-existing file is because it points to a second file that we no longer distribute (**synproto.dbl**). If your tags are not working and you have not run **wbupgrade.exe**, you should delete the **dbltags.vtg** file from your SlickEdit configuration directory.

  In summary:

  - If VSLICKCONFIG is set, the value of the environment variable is the SlickEdit configuration directory.
  - If VSLICKCONFIG is not set, the SlickEdit configuration directory is C:\Documents and Settings*user*\My Documents\My SlickEdit Config*version*, where *user* is the user name of the currently logged-in user and *version* is the current version of SlickEdit in the format *x.x.x* (for example, 11.0.2). This is the recommended configuration.

  (9.1)

COM, Java, and .NET projects

- Workbench now allows you to pass the **-w** flag to **gencs.exe** from within a .NET Component project. The **-w** option causes the assembly to be generated to use WCF contracts. To use this option in Workbench, you must do the following:

  1. Select Project > Project Properties.
  2. On the Tools tab, select Generate C# Classes.
  3. At the end of the command line, add a space followed by "-w".

  (9.5.1a)

Customizations

- SlickEdit has replaced the VSLICKCONFIG environment variable with SLICKEDITCONFIG. If you customized your configuration directory with VSLICKCONFIG, you will need to update your environment, as VSLICKCONFIG will no longer work. (9.3)
- All keyword customizations (additions/deletions) will be lost when upgrading to version 9. Customizations made in version 9 will be retained when upgrading to future versions. (9.1)

Synergy Method Catalog

- Workbench 9.3 supports only version 9.3 or higher Synergy Method Catalog files. (9.3)

Synergy/DE Options Dialog

- The caption of the Synergy/DE Options dialog now includes the type of project in which you are currently working. (9.5.1a)
- On the Compile tab, the "Disable the specified warnings (-WD)" field now includes a drilldown button that enables you to select which warnings should be disable with the -WD compiler option. (9.5)
- We now offer a switch for the command that opens the Synergy/DE Options dialog. By adding "-ac" as a parameter, the command will open the Synergy/DE Options dialog to the active configuration, rather than to "All Configurations." (9.3.1a)

Synergy/DE Projects

- Workbench now executes project open commands when switching project configurations. This allows more fine-tuned control of your environment when working in Synergy/DE projects. To go along with this change, Workbench also offers a SYNBITSIZE environment variable that will indicate either 32 or 64 depending on the current configuration in Synergy/DE, Synergy/DE Application, Synergy/DE Executable Library, and Synergy/DE Object Library projects. Use the SYNBITSIZE variable when defining Synergy logicals to keep your 32-bit and 64-bit build environments separate.

  For Workbench users who set PATH in the project open tab, Workbench now resets the PATH environment variable to the value of PATH from when the project was opened, whenever the active project or configuration is changed. This helps to prevent PATH from growing too large from recurring set statements. (9.5)
- By request, Debug32 is now the default configuration in new Synergy/DE, Synergy/DE Application, Synergy/DE Executable Library (ELB), and Synergy/DE Object Library (OLB) projects. (9.3.1b)
- When certain tools in Workbench (for example, Generate C# Classes) encounter an error, the error output will now be slightly more verbose to hasten discovery of the problem (for example, licensing errors). (9.3.1b)
- We added a "Synergy/DE Application" project type, which is separate from the "Synergy/DE" project type. The "Synergy/DE Application" project includes defaults for the creation of a single DBR executable from the sources in the project. The "Synergy/DE" project includes defaults geared more towards a general development area with multiple main-routine source files. (9.3)
- We added a "Synergy/DE Object Library" project type to enable you to better manage OLBs. (9.3)
- We added a "Synergy/DE Executable Library" project type to enable you to better manage ELBs. (9.3)

- We added the Synergy/DE Options dialog to enable you to modify project-specific options, including compiler options, linker options, librarian options, and **dblproto** options. When a Synergy/DE, Synergy/DE Application, Synergy/DE Executable Library, or Synergy/DE Object Library project is open, either select "Synergy/DE Options" from the Build menu, or select "Synergy/DE Options" on the Tools tab in the Project Properties dialog, and click the Options button. (9.3)
- You can delete existing prototype files prior to prototyping using the options on the Prototype tab in the new Synergy/DE Options dialog. (9.3)
- When you configure **-qvariant** and **-qrelaxed:end** in the Compile tab of the Synergy/DE Options dialog, the options will change how your source files are tagged. This will provide better context information and more accurate results. (9.3)
- Workbench now supports the **-single** and **-qrelaxedend dblproto** features. The Prototype tab of the Synergy/DE Options dialog contains options for this feature. (9.3)
- A new option, "Compile sources into a single object file," allows you to compile all of your source files into one object file, which improves linking performance. This option is enabled by default for Synergy/DE Application, Synergy/DE Object Library, and Synergy/DE Executable Library project types on the Compile tab in this Synergy/DE Options dialog. (9.3)
- Workbench now offers support for 64-bit project configurations when the end user has concurrent 32-bit and 64-bit Synergy/DE installations. (9.3)
- Synergy Language projects (Synergy/DE, Synergy/DE Application, Synergy/DE Executable Library (ELB), and Synergy/DE Object Library (OLB)) include four separate configurations in the project templates. These configurations cover Release and Debug for 32-bit and 64-bit tool sets. The Debug configurations specify the **-d** compiler flag by default. (9.3)
- We added **.dbc** as a supported file extension for Synergy Language. This is the recommended file extension for Synergy class definitions. (9.3)
- Workbench now refers to Synergy Language by its proper name. The DBL entry no longer exists in the File > New list. Instead, you should select Synergy Language as the document mode. After you select it once, it will appear at the top of the list in the recently used file area for easier access. This also affects the Options dialog. Make sure to look for Synergy Language instead of DBL. (9.3)
- The Workbench build system supports the binary prototype file extension. (9.3)
- "Synergy/DE" projects no longer include **axlib.elb** and **tklib.elb** as a default part of the link command. (9.3)
- Workbench build commands (other than Execute and Debug) now default to macro commands to allow for better integration with the new project build system. You can replace these macro commands any way that you want to. (9.3)

- The Workbench build system now supports very large projects by utilizing the redirected input functionality in **dblproto** and the compiler. (9.3)
- The optional batch file that Workbench can generate as a part of its project build system wraps the commands in SETLOCAL and ENDLOCAL statements, by default, to preserve the environment that the batch file may be run in. (9.3)

Tagging

- Builtin functions now specify return types. Parameters for routines and functions now specify IN/OUT/INOUT and type information when applicable. (9.5.1a)
- **SynArrayList.dbl** and **SynSelect.dbl** are now distributed as "builtin" files with Workbench and are tagged automatically when a new DBL tag file is built. (9.3)

User interface

- When creating new COM, Java, or .NET Component projects in Workbench, the SMC directory field in the Component Information dialog will default to the value of the XFPL_SMCPATH environment variable, if present. (9.3)
- Synergy/DE .NET component project support has been enhanced with a new .NET Environment Configuration dialog. This enables you to select the correct environment batch file and the target Framework version. (9.3)
- The Workbench online help is now compiled HTML Help rather than WinHelp. (9.1)
- We removed the "Keyword Case" field from the DBL Options dialog because we are no longer performing keyword completion. When syntax expansion adds text, it now uses the case of the first character of the word that triggered syntax expansion to begin. (9.1)
- We removed the "Minimum Expandable Keyword Length" field from the DBL Options dialog, as the option can now be set in the Indent tab of the Extension Options dialog. (9.1)
- The "Use SmartPaste" option has been removed from the DBL Options dialog since it is now a part of SlickEdit's Extension Options dialog. (9.1)
- The DBL Options dialog no longer contains an option called "Add End After Begin." If syntax expansion is turned on (via the "Syntax expansion" option on the Indent tab of the Extension Options dialog), Workbench always adds END when BEGIN is seen. (9.1)
- We enhanced the .NET Component Information dialog. We added options to generate structure members as fields instead of properties and to generate multiple copies of classes. (9.1)

Utilities

- "Generate Synergy Test Skeletons" has been removed from the Build menu in the distributed project templates. The tool will continue to be distributed, however, and you can access the dialog box by entering "SynStartSkeletonGen" on the Workbench command line. If you use this feature frequently, you may want to add it as a menu option (using Project Properties > Tools) in Workbench. (9.3)
- When launching the Method Definition Utility, Workbench will first try to use an existing SMC path that the project is already using (from previous launches of the MDU or from the Component Information dialog). If no SMC can be found, Workbench will proceed to try XFPL_SMCPATH, then DBLDIR. (9.3)
- We added a menu entry to the Synergy/DE project type that enables you to generate Synergy Language prototypes for the source files in your project. The "Generate Synergy Prototypes" entry is located on the Build menu. (9.1)

Visual editor

- Workbench now has support for XMLDoc comments for Synergy Language. When documentation comments have been written for a routine, calling the routine or performing a mouseover of the routine name will show a parsed view of the doc comments. To ease the documenting of routines, typing ";;;" on the line prior to a routine will generate a documentation comment template containing "summary" and "returns" regions, as well as a region for each parameter, if there are any. (9.5)
- We have improved support for .REGION/.ENDREGION. You no longer need to use show_code_block and hide_code_block to trigger a region to collapse. When you open a document containing a .REGION/.ENDREGION pair, Workbench will collapse the blocks for you by default. If you already had some blocks closed or open in the file, your view won't be changed. You can use the default key binding (Ctrl+\) to expand and collapse views, or bind your own key to "plusminus". To collapse all of the .REGION/.ENDREGION blocks in the document, use View > Hide #region Blocks, or enter "hide_dotnet_regions" from the command line. (9.5)
- Workbench now supports the .REGION/.ENDREGION syntax, which makes collapsible regions possible. To trigger the collapsible region in Workbench, type "hide_code_block" on the SlickEdit command line or bind a key to the command. (9.3)
- Workbench now fully supports enumerations in Synergy Language. Keyword coloring, syntax expansion, indentation and tagging support are all included in this version. (9.3)
- We have fixed the known bugs with SmartPaste and now enable it by default in new installations. We recommend that you turn the feature back on in upgraded installations and try it again. (9.3)

- We added Keyword Case options back to Workbench. The original three options for lowercase, uppercase, and initial capital have been reinstated. The option for determining the case based on the first letter typed that was available in 9.1.1-9.1.5 is still available. To access the case options, select Tools > Options and then, in the left pane, navigate to Languages > Application Languages > Synergy Language > Formatting. It affects the Syntax Expansion and Keyword Completion features. (9.3)
- We improved the way Workbench handles comment lines. Previously, it would treat trailing white space as text when determining whether to split a line or indent a new line when Enter was pressed. As a result, comment lines could be split and generate additional comment characters on the new lines. Workbench now works harder to ignore trailing white space when determining indents, and attempts to clean up trailing white space when Enter is pressed. (9.3)
- SlickEdit has introduced a new feature called Adaptive Formatting, which is configurable at the language level. To ensure code quality and integrity, we are leaving this feature turned off by default for Synergy Language. To enable this feature, select Tools > Options and then, in the left pane, navigate to Languages > Application Languages > Synergy Language > Adaptive Formatting, and select the "Use Adaptive Formatting" option. (9.3)
- The new **synexceptions.dbl** file (split off from **synclasses.dbl**) is automatically tagged in the DBL tag file. (9.1.5)
- Workbench no longer generate tags for imports. If you want tags for your imported classes and namespaces, you must either add the source files to your workspace or create a separate tag file for those sources. (9.1.5)
- Workbench tagging now includes System.Object and System.String classes and members. (9.1.5)
- Workbench supports the new Toolkit %B_INFO function, including context-sensitive routine help and keyword color coding. (9.1.3)
- Synergy/DE project types are now grouped under Synergy/DE in the New Project menu. (9.1.3)
- Syntax expansion is now enabled for .MAIN, .SUBROUTINE, and. FUNCTION. (9.1.3)
- Workbench now provides additional support for class fields that have the CONST, READONLY, and VOLATILE modifiers. (9.1.1b)
- Field lists are now available for the system-supplied classes (for example, System.Exception). (9.1.1b)
- We updated the aliases in the distributed **dbl.als** file to provide better information for aliases when they show up in the auto-complete list. (9.1.1b)
- Synergy/DE-specific "Item templates" are now available in Workbench. Item templates enable you to add entire new source files to your project, based on a piece of template code. To access these templates, select "New Item from Template" from the File menu and then browse the templates available under SynergyDE. (9.1.1b)

- We added support for the "Surround Selection With…" context menu entry in Workbench. This menu entry enables you to surround a selected region with a code block. The text generated by the command can be modified in **dbl.als**. (9.1.1a)
- Workbench now highlights all operators and library symbols with the colors that are defined by the "Operators" and "Lib symbols" tokens, respectively, in the Color Coding Setup dialog. (9.1)
- Workbench no longer supports keyword completion. It now uses the "auto-complete" feature of SlickEdit, which can be accessed via the Auto-Complete tab of the Extension Options dialog for the **dbl** file extension. If you select both the "Enable auto-completion" and "Keywords" check boxes on that tab, when you type the beginning of a keyword, a drop-down list of possible keywords is displayed. (This drop-down list is a general SlickEdit feature that also displays Syntax expansion, Alias expansion, Symbols, and Word completion sections.) (9.1)
- Workbench has increased its support for syntax expansion. (Syntax expansion is controlled by the "Syntax expansion" check box on the Indent tab of the Extension Options dialog for the **dbl** file extension.) Previously it only expanded BEGIN, USING, and a few others. Now it expands BEGIN, CASE, CLASS, DO, FOR, FUNCTION, GLOBAL DATA, GROUP, MAIN, METHOD, NAMESPACE, PROPERTY, SUBROUTINE, THEN, TRY, and USING when the spacebar is pressed immediately following the keyword. (Some expansions are also triggered by pressing ENTER.) (9.1)
- When syntax expansion is turned on, if you also select "Extended Syntax Expansion" in the DBL Options dialog, Workbench also generates the following keywords: ENDCOMMON, ENDEXTERNAL, ENDFUNCTION, ENDLITERAL, ENDMAIN, ENDRECORD, ENDSTRUCTURE, and ENDSUBROUTINE. (9.1)
- Workbench now supports CTRL+] on more beginning/ending pairs (e.g., BEGIN/END, DO/UNTIL, GLOBAL/ENDGLOBAL). With beginning/ending pairs, pressing CTRL+] while your cursor is on one member of the pair will move your cursor to the other member of the pair. In addition, SlickEdit now provides a visual cue, by highlighting both members of a pair when the cursor is on one or the other.
- Workbench has full support for Synergy objects in the tagging and indentation engines:
  - When you request a field list (ALT+period), Workbench displays all variables that are defined as class types within the current routine being edited, in the same manner as it displays variables.
  - When you type a method name after a class handle, followed by a parenthesis, Workbench brings up the context-sensitive help for the method.
  - The following items are now supported in the Classes tab of the Project toolbar: imports, classes, namespaces, global structures, properties, class fields, class groups, and methods.

  (9.1)
- Local data and catch variables are now supported in the Fields tab of the Project toolbar. (9.1)

- For Workbench to control indentation, “Syntax indent” must be set as the “Indent style” for the **dbl** file extension in the Indent tab of the Extension Options dialog. If “Indent style” is set to “None” or “Auto,” the behavior is controlled by SlickEdit, not Workbench. (9.1)
- We improved the performance of tagging. (9.1)
- All Synergy/DE projects in a workspace are automatically retagged on open if the version of the running tagging engine is different than the version stored in the workspace. (9.1)
- We added the argument type to the information displayed as part of context-sensitive help. (9.1)
- If a function, subroutine, or method parameter is specified as optional, it will be enclosed in square brackets within the context-sensitive help. (9.1)
- We added support for the new compiler qualifiers in the .INCLUDE repository directive. This includes better support when repository definitions are used to define routine parameters. (9.1)
- Workbench now has better support for group subroutine arguments. (9.1)
- Workbench now processes .INCLUDEs while processing the source files. As this processing may require a lot of memory and processor overhead, you may want to set the new WBNOINC environment variable to suppress this feature. Setting WBNOINC means that Workbench will not process global data sections, structures, and other items in the include files. Running Workbench with WBNOINC set is equivalent to the behavior in version 8.3 of Workbench. (9.1)

Version 8

This section briefly describes new features in Workbench version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Workbench version 8.

- In previous versions of Workbench, if you used an environment variable in a project open command, you used the "%VAR%" form to refer to it. SlickEdit now requires that you use the "%(VAR)" form. (8.3)
- If you have Workbench customizations or user-specific settings that you want to maintain, *you must upgrade* your existing installation. If you uninstall, the VSLICKCONFIG setting will be lost and SlickEdit will create new configuration files. See "New features" below for more information. (8.3)
- Workbench 8.3 uses SlickEdit version 10 and therefore requires that all existing Synergy/DE projects be converted. With a few exceptions (see "New features" below), this happens automatically when a project or workspace is opened for the first time in 8.3. (8.3)
- Workbench 8.3 requires a SlickEdit version 10 key. This key is specified using the Synergy Configuration Program *after* you complete the installation. This key is required on both a new installation and an upgrade. (8.3)
- (Windows 2000/XP) The installation now defaults to C:\Program Files\Synergex\synergyde. Therefore, if you uninstall your previous version of Workbench and install version 8 to the default directory, you will lose your Workbench customizations. *You should change the default installation directory to be your previous Synergy/DE directory.* If you "upgrade" to version 8, you will not have this problem, as an upgrade defaults to the location of your currently installed version. When Workbench is re-installed to the same directory, most customizations are automatically saved. See the Workbench release notes (**REL_WB.TXT**) for more information regarding saving/restoring Workbench customizations. (8.1)
- Workbench can now be installed to a directory that has spaces in its path. If you choose to install Workbench to the new default directory (C:\Program Files\Synergex\synergyde), or to any other directory with spaces, you will need to update existing projects and commands to include double quotes around the directory paths so that the DBLDIR logical is translated correctly. Refer to the Workbench release notes (**REL_WB.TXT**) for more information. (8.1)

New features

- Workbench has been upgraded to use Slickedit 10.0.3. (8.3.1b)
- Launching the Synergy Online Manuals from Workbench will launch either Adobe Reader or Adobe Acrobat, depending on which is the default application for the **.pdf** file type on your machine. (8.3.1b)
- In the Compile Scripts Setup dialog, the library name and the list of script files are now saved as relative paths in the **.sde** file for the project. (8.3.1b)
- We added two new entries to the Build menu for Synergy/DE projects:
  - Compile Scripts Setup, which enables you to specify compilation options and compile one or more Toolkit window scripts residing in a Synergy/DE project.
  - Compile Scripts, which enables you to easily compile all or selected Toolkit window scripts residing in a Synergy/DE project.

  (8.3)
- The Text file field in the Repository panel of the Component Information dialog, the Repository text file field in the Method Definition Utility Options dialog, and the Text file field in the Generate Synergy Test Skeleton Options dialog will be prepopulated to reflect the name of the repository main file. If you entered text in the Main file field or Repository main file field by browsing for it, Workbench will copy the repository main filename and change the last occurrence of the characters “main” to “text”. However, if you typed a filename rather than browsing for it, this defaulting will not occur. (8.3)
- We added support for the commands “box” and “comment-erase” in DBL source files. Note that if you do not select “First line is top” and “Last line is bottom” in the Comments tab of the Extension Options dialog, “comment-erase” will only remove a single comment from the beginning of each line in the selected area. (To access the Extension Options dialog, select Tools > Options > File Extension Setup.) If you want to use the built-in SlickEdit command “comment”, you will need to do the following:

  1. Select Macro > Menus.
  2. Select _mdi_menu from the list and click the Open button.
  3. Double-click on Do&cument.
  4. Select Comment Lines and click the Auto Enable button.
  5. Make sure Requires block selection is not checked.
  6. Click OK in the Auto Enable Properties for comment dialog.
  7. Click OK in the Menu Editor dialog.

  (8.3)

- To eliminate project file synchronization issues between Workbench and Composer, we now require that when using these products together, Composer be launched from within Workbench. Workbench can no longer be launched from Composer. (8.1)
- Synergy/DE (including Workbench) now supports the concept of a "system-specific" initialization file (**synergy.ini**) versus a "user-specific" initialization file (**synuser.ini**). Settings in the user-specific file override settings in the system-specific file. (For more information on this, refer to the "Environment Variables" chapter of *Environment Variables and System Options*.) After loading environment variables from the [synergy] section of the **synergy.ini** file, Workbench will load environment variables from the [synergy] section of the **synuser.ini** file, possibly overriding environment variables in **synergy.ini**. (8.1)

Installation

- On a new installation, the Synergy/DE and Synergy/DE Client installation programs no longer set the VSLICKCONFIG environment variable. When VSLICKCONFIG is not set, SlickEdit version 10 creates its own directory based on the logged-in user. (See following entries for more information.) Additionally, we no longer require that each client of a shared installation or each user of a terminal services installation create a directory on the client machine, copy configuration files to it, and set the VSLICKCONFIG environment variable. Starting Workbench runs **update_synergy**, which initializes Workbench and creates the necessary configuration files in the SlickEdit configuration directory. (8.3)
- On an upgrade installation, the Synergy/DE and Synergy/DE Client installation programs retain any previous VSLICKCONFIG setting. After you upgrade, we strongly recommend that you run the **wbupgrade.exe** program. See related entry below. (8.3)
- IMPORTANT: You do not need to upgrade Synergy/DE Clients for version 8.3. Only the shared installation needs to be upgraded. The clients will continue to use VSLICKCONFIG. (We assume that VSLICKCONFIG is set at the user level pointing to a directory on the local machine, and set at the system level pointing to the wbconfig directory on the shared machine.) The same is true for a Workbench installation on a Terminal Services machine. Once Workbench has been upgraded, the Terminal Services users will continue to use VSLICKCONFIG. (We assume that VSLICKCONFIG is set at the user level pointing to a user-specific directory on the Terminal Services machine, and set at the system level pointing to the wbconfig directory.)

  Although local installations, Synergy/DE Clients, and users of Workbench on a Terminal Services machine will continue to run with VSLICKCONFIG set, we *strongly recommend that each workbench user run the **wbupgrade.exe** program* to eliminate the use of VSLICKCONFIG moving forward. See next entry for more information. (8.3)

- IMPORTANT: After upgrading a local or shared installation of Workbench (including one on a Terminal Services machine), each user should run the **wbupgrade.exe** program located in the \workbench\win directory. This program creates the SlickEdit 10 user-specific configuration directory; copies all files from the current VSLICKCONFIG directory to it; and then deletes the system and user-level VSLICKCONFIG settings. Note: If you are upgrading an installation on a Terminal Services machine, you must reboot after the administrator has run the **wbupgrade.exe** program. This ensures that the deletion of the system-level VSLICKCONFIG setting is recognized by all users. (8.3)
- The **wbupgrade.exe** program also deletes any **dbltags.vtg** file (from an earlier version of Workbench) found in the SlickEdit configuration directory. This file gets recreated the first time you request tagging. The reason we do not want to use a pre-existing file is because it points to a second file that we no longer distribute (**synproto.dbl**). If your tags are not working and you have not run **wbupgrade.exe**, you should delete the **dbltags.vtg** file from your SlickEdit configuration directory. (8.3)
- In summary:
  - If VSLICKCONFIG is set, the value of the environment variable is the SlickEdit configuration directory.
  - If VSLICKCONFIG is not set, the SlickEdit configuration directory is C:\Documents and Settings*user*\My Documents\My SlickEdit Config*version*, where *user* is the user name of the currently logged-in user and *version* is the current version of SlickEdit in the format *x.x.x* (for example, 10.0.2). This is the recommended configuration.

  (8.3)

Project conversion

- Workbench 8.3 uses SlickEdit version 10 and therefore requires that all existing Synergy/DE projects be converted. With a few exceptions (see note below), this happens automatically when a project or workspace is opened for the first time in 8.3.

  Due to the extent of SlickEdit's changes to the version 10 project file (VPJ), we now store most Synergy-specific information in an associated SDE file (***project_name*.sde**). An SDE file is created for an existing project as part of the conversion process. The VPJ file will continue to store default file extension information, as well as non-Synergy information.

  Note: The following projects created in version 7.3.1 do not convert correctly:

  - Synergy/DE Java;Release
  - Synergy/DE COM;Global
  - Synergy/DE COM;Release

  If you attempt to open these projects in 8.3, the conversion will cause the projects to be seen as non-Synergy projects. (8.3)
- Workbench 8.3 also supports the conversion of project templates from Synergy/DE versions 7.3, 7.5, and 8.1. (Version 7.1 project templates cannot be converted because they exist in a file that is uninstalled with 7.1, **prjpacks.slk**.) (8.3)

Component projects

- Workbench now supports xfNetLink .NET projects. You must have Framework version 1.1 or higher installed to create a Synergy/DE.NET component project. (8.3)

  When attempting to create a Synergy/DE .NET component project, Workbench will determine the highest version of the .NET Framework SDK that is installed. If that version is not valid for the current version of Workbench (or Framework is not installed at all), an error is generated and no project is created. If the Framework version is valid, Workbench invokes the macro **syn_set_sdkvars** *version*, where *version* is the Framework version. (For more information about **syn_set_sdkvars**, see the "Developing Your Application in Workbench" chapter of the *Getting Started with Synergy/DE* manual.) (8.3)

- The "Synergy/DE Web" project type is no longer supported. If you attempt to open a Synergy/DE Web project in 8.3, the conversion will cause the project to be seen as a non-Synergy project. (8.3)
- We added support for Java packages. You can now specify a package name for the Java classes you are generating using the new Package field in the Component Information dialog for Synergy/DE Java component projects. The default package name is the name of the JAR file. The maximum length of a package name is 101 characters. (8.1.7)

Customizations

- If you are upgrading from Workbench version 7.3.1 or higher (assuming VSLICKCONFIG is set), many customizations are saved/restored automatically. Here are some exceptions:
  - The Synergy toolbar has been replaced in version 8.3 so that it is consistent with the look and feel of SlickEdit 10. Any customizations you have made to this toolbar will be lost.
  - Customized menus will be lost when upgrading to 8.3, except for Build menu entries, which are stored with the project.
  - Modifications to distributed macros are lost on upgrade, with the exception of **dbl.e**. See the "Important Notes and Warnings" section of **REL_WB.TXT** if you have customized this file.

  Note that your own macros are not overwritten, but they must be reloaded into Workbench after installation. For each macro, at the Workbench command line, type LOAD *macro*.e (where *macro* is the path and name of the macro). (8.3)

SlickEdit Version 10

- SlickEdit 6.0c (distributed with Workbench 7.5 and 8.1) used to create user1 and user2 menu commands for all projects when they were created. SlickEdit 10 no longer does this. To create a user-defined tool entry, enter a new tool name of “user 1” or “user 2” in the Tool name field of the Tools tab in the Project Properties dialog, and fill in the rest of the fields as desired. The command “project-user1” or “project-user2” (as a menu entry, toolbar, or shortcut key command) will execute the specified command line for user 1 or user 2, respectively. (8.3)
- SlickEdit 10 allows you to unset the read-only flag of a file that is archived in a version control system. Generally we do not recommend that users change these attributes. When prompted to unset the flag, you should reply “No”. (8.3)
- Because of changes required for compatibility with SlickEdit 10, we no longer automatically build the tags database until a user specifically requests it. This means that if you request References or Calls or uses information before you request the tags database, symbols in the Synergy prototype file will not be seen. To specifically request that the tags be created for this file, type the character “.” in any open Synergy Language source file. (8.3)
- SlickEdit has changed the enabled status of some menu entries between SlickEdit 6.0c (which was distributed with Workbench versions 7.5 and 8.1) and SlickEdit 10 (which is distributed with Workbench version 8.3). (8.3)

Version 7

This section briefly describes new features in Workbench version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Workbench version 7.

- File specifications within Visual SlickEdit version 6 project files are relative to the location of the project file itself. This may affect your use of the project file if you were previously depending on explicit drive specifications. (7.5)

New features

- The default for the "Start tagging after seconds idle" value ("Tools" > "Configuration" > "Tagging Options…" is now three seconds. The original default value of 0 caused delays while the user was editing a file. (7.5)
- When a user creates a project using Project > New, Workbench now prompts to create the directory specified in the "Location" field if it does not already exist. (7.5)
- We added a new menu entry, "Workbench Help Topics", to the Synergy/DE menu. This menu entry launches procedural-style Workbench help. (7.5)
- It is now possible to change the command lines for existing tools and add new tools that Workbench will use when creating a new Synergy/DE project. Follow these steps:

  1. Select "Project" > "New".
  2. Select the "Customize" button.
  3. Select the project type you want to customize.
  4. Select the "Edit" button.
  5. Select the "Tools" tab on the "Project Package" dialog.
  6. Select the tool you wish to change.
  7. Make the desired changes to the command line and select "OK".

  Note: We do not support or recommend changes to the project packages for "Synergy/DE COM Component;Release" or "Synergy/DE Java Component;Release". Synergex reserves the right to override user settings during a Workbench installation/upgrade. (7.5)
- Opening a project or a workspace now changes the current working directory to the directory where the project is stored. (7.5)

- Workbench has been upgraded to use Visual SlickEdit version 6. (Your SlickEdit version 6 key must be configured using the Synergy Configuration Program after you have completed the Workbench installation.) (7.5)
- As a result of the upgrade to Visual SlickEdit version 6, all project files must be converted. This conversion occurs automatically when you open a (SlickEdit) pre-version 6 project file. (7.5)
- You can now associate an ActiveX control with a project. To do so, right-click on the project and select "Add to Project." In the dialog box, select the **.ocx** file you want to associate with the project. This association is saved automatically when the project is saved. Once an ActiveX control is added to the project, a utility is run that loads the classes and their methods and fields specified in the ActiveX control into the project's tags database. You can also browse an ActiveX control from Workbench. (7.5)
- A Java archive (**.jar**) file can now be associated with a Synergy/DE Web project, such that its classes, methods, and fields can be viewed within Workbench. (7.5)
- The Component Information dialog for Synergy/DE component projects has been modified to support Java class wrappers. Generating a Synergy/DE Component project will create the Java source code and other files needed to build the Java JAR file. These JAR files make it possible to call Synergy logic from Java applications and JSP pages. (7.5)
- We added a "Use alternate field names" check box to the Component Information dialog. Check this box to specify that you want the field names in your structure to use the value in the Alternate name field in Repository instead of the value in the Name field. (7.5)
- We added a Generate Javadoc check box to the Component Information dialog for Java component projects. Check this box to specify that Javadoc HTML files should be generated for this component. (7.5)
- You can launch a new utility from Workbench that reports on unused variable information generated by the compiler. (7.5)
- We restructured the Synergy/DE menu. (7.5)
- You can launch the new test skeleton generator from Workbench. (7.5)
- The "Launch WebBuilder" menu entry and toolbar button now tests if UltraDev 4 is installed and if so, launches it. If UltraDev 4 is not installed, it tests for Dreamweaver 4. If Dreamweaver 4 is installed, it launches it. If Dreamweaver 4 is not installed, it tests for Dreamweaver 3. If none of these versions are installed, an error message is displayed. (7.5)
- Workbench now supports the "group" qualifier when creating the tag file. (7.5)
- We added a new Launch Home Page menu entry and toolbar button that will launch the web browser defined in Tools > Configuration > Web Browser, opening the specified URL. The URL is stored in the web project file for Synergy/DE Web projects and is retrieved each time the home page is launched. (7.3)

- We restructured the Synergy/DE menu column. We added Dreamweaver and Online Manuals to the menu, and removed Proto, ReportWriter, Script-to-Repository Conversion, Synergy Control Panel, Make Synergy, and Unmake Synergy. We also added a new Utilities submenu. (7.3)
- We restructured the Synergy/DE toolbar. We added Dreamweaver, Online Manuals, and the Method Definition Utility. We removed Proto, ReportWriter, Script-Repository Conversion, and Synergy Control Panel. The Synergy/DE toolbar will not be updated (when you install version 7.3) if you have customized it. (7.3)
- Workbench now includes directory and extension-specific aliases. Directory aliases simplify the specification of path names. Extension-specific aliases are used for syntax and code expansion, enabling you to generate entire sections of code automatically. (7.3)
- The default Execute command no longer uses the SYN_DBG environment variable for executing in debug mode. The default is now to execute in non-debug mode. To execute in debug mode, select Build > Debug. (7.3)
- We added a new File Extension Maintenance dialog that enables you to add/remove Synergy/DE file extensions. This dialog is accessible only when a Synergy/DE project or no project is active. (7.3)
- Licensee name and registration string information is now available from the "About Workbench" dialog. (7.3)
- We changed the name of the "Synergy/DE for Windows" project to "Synergy/DE" and have added two new project types: Synergy/DE Component and Synergy/DE Web. (7.3)
- Each method in the template files now includes the **WND:tools.def** file. (7.3)
- Workbench can now launch the online manuals. The default location is c:\synergyde\manuals. You can, however, change this location by running SynSetDocs, a new Visual SlickEdit® command. If you run this command and pass a directory path, the path will be set as the new location. To reset to the default location (c:\synergyde\manuals), run SynSetDocs *without* a parameter. (7.3)
- Workbench now inserts routine call references into the tags database for all subroutines and function calls. These entries in the tags database will make browsing the "Calls or uses" trees more reliable. Also, Workbench users can now search for all references to a routine within a workspace by placing the cursor on the routine they want to find references for and pressing CTRL+/. (7.3)
- We added a new template for button methods (**button.tpl**), such that pressing the drilldown button for a button method in Composer will insert the button code template into Workbench's editor. (7.3)
- We added two new input field templates for display (**display.tpl**) and edit format methods (**editfmt.tpl**). When the drilldown button for the Display or Edit format method is pressed, the corresponding template code is inserted into Workbench's editor. (7.3)

- We added two new template tokens: #STRUCTURE# and #FIELD#. Within a code template, #STRUCTURE# is replaced with the structure of a field (repository or local) or blank if there is none. #FIELD# is replaced with the name of the field. (7.3)
- We added the following commands: syn_set allows an environment variable to be set at the project level, syn_set_global sets an environment variable at a global level (which gets reset whenever a project is made active), and syn_set_synergy_ini tells Workbench to load the environment variable from within the [synergy] section of the **synergy.ini** file. If you specify a directory location as a parameter to syn_set_synergy_ini, **synergy.ini** will be loaded from that location instead of SFWINIPATH. These settings will apply for the life of the active project. These commands can be run either from the Visual SlickEdit command window or when a project is opened (Open Command Tab). (7.3)
- You can now call Synergy business logic from COM-compliant clients such as Visual Basic (VB) and Active Server Pages (ASP) by creating COM type libraries that will allow VB and ASP programmers to call groups of related routines as components. For example, you can bundle a group of order entry routines into a type library, and then an ASP or VB programmer can create an order object in their application and call your Synergy routines as follows: myorder.getStatus(orderNumber). This is done by creating a Synergy/DE Component Project in Workbench, defining the Synergy methods you want grouped in the type library using the Synergy Method Definition Utility, and then selecting Build from the Build menu. The steps for creating a COM type library from Workbench are fully documented in the "Creating Synergy COM Components" chapter of the *Developing Distributed Synergy Applications* manual. (7.3)
- A type library (**.tlb**) file can be associated with a Synergy/DE Web project, such that its classes, methods, and fields can be viewed within Workbench. (7.3)
- We added a Component Information dialog for Synergy/DE component projects. The information in this dialog is used to create type libraries. Building a Synergy/DE Component project will create a type library and optionally register it. These type libraries make it possible to call Synergy logic from COM-compliant front ends such as VB and ASP. (7.3)
- We added a new Synergy Type Library Configuration utility to configure registry settings on the client machine. This utility allows the user to add and modify settings at the default and interface levels for *xf*NetLink COM. This utility is available from the SynergyDE > Utilities menu in Workbench, as well as from the SynergyDE > Utilities folder on the Start menu. (7.3)
- We added an SMC/ELB Comparison Utility (**smc_elb.exe**), which compares method names in an SMC against the method names in an ELB. This utility is automatically run when you build a Synergy type library. You can also run it from the Synergy/DE > Project > SMC/ELB Comparison menu in Workbench. (7.3)
- We added context help for many of the Synergy Language statements (READ, READS, etc.). (7.1.3)

6

Repository

Version 9

This section briefly describes new features in Repository version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Repository version 9.

- As required by the corresponding Synergy Language change in version 9.5.1, the size of Boolean fields and templates is now 4 instead of 1. This modification could break your code if you have written Boolean data to a file or if you .INCLUDE Repository structures containing Booleans as named records. You must update your repository by doing one of the following. (You should update your repository even if you are not using Booleans as described above.)
  - Change the size of Booleans to 4. To do this, generate your repository to schema, then do a Search and Replace of "Type BOOLEAN Size 1" with "Type BOOLEAN Size 4". (Note that there are three spaces before "Size".)
  - Change Booleans to Integers, leaving the size at 1. To do this, generate your repository to schema, and then do a Search and Replace of " Type BOOLEAN" with " Type INTEGER". (Note that there are two spaces before "Type". This is so that you don't accidentally replace "Type BOOLEAN" in "Coerced Type BOOLEAN".)

  (9.5)
- The data type of the seventh argument to DD_ALIAS has been changed from alpha to numeric. (9.3.1a)

New features

Repository database

- A coerced type can now be specified for a field or template when the Type is User, the Class is Date, and the User string is "^CLASS^=YYYYMMDDHHMISSUUUUUU". (9.5.3)
- Repository now supports the specification of an Alias name when defining a field of type Struct (structfield). (9.5.1b)
- Repository now uses unique temporary filenames constructed using %DATETIME and prefixed with "RPS_". (9.5)
- We added support for two new data types for fields and templates: Boolean and User-defined Binary. Boolean has been added to the drop-down selection list for Type. When you select a Boolean data type, Size is set to 1 and cannot be changed. Binary has been added to the drop-down selection list for Class and is intended for use with *xf*ODBC. (9.3)

- We added support for the Struct (structfield) data type for fields. When you select Struct, the "User data" field changes to Structure, and is where you enter the structure name. (9.3)
- Repository now supports an Enumeration definition type. Fields and templates can be defined to reference an enumeration and you can .INCLUDE a repository enumeration into your Synergy Language source file. (9.3)
- We added the DD_ENUM subroutine to the Repository subroutine library to support the enumeration definition type. To use this subroutine, you must recompile code that references **ddinfo.def** (and if you have your own copy of the **dcs**, it must be modified). (9.3)
- We changed the Repository from having a limit of 999 fields/groups per structure including groups within the structure, to a limit of 999 fields/groups at the structure level and for each group in that structure. Internally, we maintain, but no longer use, the **si_nmflds** variable (field/group count for the whole structure). Instead, we use the **si_childct** and **fi_childct** variables to compute the total count when necessary. If the total count equals or exceeds 999, we set **si_nmflds** to 999. This is not used internally, but if you use DD_STRUCT in the Repository Subroutine Library and the **si_nmflds** value returned is 999, your application must iterate through the structure and its groups to compute the total field count. (9.3)
- The Repository window library (**rpsctl.is?**) and original window script file (**rpsctl.wsc**) are now distributed in the RPS directory instead of the RPSDAT directory. The sequential file containing the repository messages, **rpstxt.ddf**, is also distributed in RPS instead of RPSDAT. Having the window library and default repository files in the same directory was inconvenient for users who set RPSDAT to point to a non-default repository because they also had to move **rpsctl.ism** (and then remember to copy the new window library when they upgraded to a new version). (9.3)
- We added a new coerced type (nullable decimal) for implied-decimal fields and two new coerced types (decimal and nullable decimal) for decimal fields. (Coerced types are used by *xf*NetLink .NET.) (9.3)
- Repository now supports a "Null allowed" attribute for fields and templates. This value is used by *xf*ODBC to determine the null property for the column in the system catalog. The default value is Default, which means that SODBC_NONULL is used. (9.3)
- Repository now allows the precision of an implied-decimal field to be a maximum of 28. The previous maximum was 10. Note that we did not modify the size of UI Toolkit's Range minimum and Range maximum fields, as that would have required a repository conversion. If you need to specify range values larger than the pre-9.3 limit, set them in the window definitions within your script file instead. (9.3)
- When loading fields from an .INCLUDE file, we now allow a maximum line length of 300. (Previously it was 150.) (9.3)
- The **rpsutl** program now disables file flushing (DBLOPT 36). This improves performance when DBLOPT 36 is set in the environment. (9.3)
- (Windows, UNIX) You are no longer required to link with **WND:tklib.elb** when linking with the Repository subroutine library, **RPSLIB:ddlib.elb**; nor are you required to call U_START before DD_INIT. (9.1.3)

- The "Load field from .INCLUDE" feature (Field Functions > Load Fields) now handles the following version 9 syntax: PUBLIC, PRIVATE, PROTECTED, CONST, READONLY. (9.1.3)
- We added support for a coerced type value in the Field and Template definitions. The coerced type can be used to specify a non-default data type on the client side when the structure that the field belongs to is included in an *xf*NetLink .NET assembly. (9.1.3)
- (Windows, UNIX) If RPSTMP is not set, Repository's temp files are now created in the directory that TEMP is set to. If TEMP is not set, they are placed in the current directory, as they were previously. (9.1)

Repository user interface

- The Long Description for fields and templates is no longer in a separate window, only accessible when viewing the "list". It is now another tab in the Field Definition and Template Definition tabsets. (9.5.3)
- In version 9.5.1a, we changed the drop-down list for implied-decimal field coerced types to display "Default" instead of "decimal". Because *xf*NetLink Java supports type coercion in 9.5.3, we added "decimal" back to the list. For Java clients, implied-decimal data types can be coerced to decimal, while choosing "Default" will result in the documented default behavior. For .NET clients, selecting either "decimal" or "Default" will result in the same behavior that "Default" used to get. (9.5.3)
- We added UI support for the definition of Alias structures. An alias is an alternate name for a structure. Aliases are useful when converting an application to use the Repository and you don't want to change structure names in your code, or when you are converting records to structures and you need to define a structfield of that type, but you still have it as a record also. NOTE: If you previously created aliases in your repository via schema, *you must run the verify utility* to update a new alias counter maintained with each structure. (9.5.1b)
- The **rpsutl** utility now supports a **-h** option for displaying a usage screen. (9.5.1a)
- (Windows) The **rpsutl** utility was modified to sleep for 5 seconds when an error message is displayed, allowing the user time to read the message. (9.5.1a)
- When the field data type is implied-decimal, the drop-down list for the Coerced type field now displays "Default" by default instead of "decimal". This is to make it less confusing for users of Java, where an implied-decimal data type is mapped to either a double or BigDecimal data type depending on field size. (Note that type coercion is not yet supported for *xf*NetLink Java.) (9.5.1a)
- We made several improvements to the Find function for lists. You can do either a "contains" or a "starts with" find; the find wraps rather than requiring that you toggle direction; and there is a Find Next option. (9.3)
- When you enter View mode, a message informs you of this fact and reminds you that changes will not be saved. (9.3)

- We improved the error message that displays when the Repository files are in use, not found, or read-only. (9.3)
- The error message that displays when overwriting an existing file with the Generate Definition File or Print Repository Definitions utilities has been modified to more clearly explain what the Yes and No options do. (9.3)
- You are no longer prompted to save changes when exiting a structure modification in which you had only viewed a key definition (not modified it). (9.3)
- (Windows) The INCLUDE File field in the Load Fields dialog is now a scrollable field on Windows, allowing you to select a filename whose full path is up to 255 characters. (9.1)

Version 8

This section briefly describes new features in Repository version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

Repository database

- We added a new option, Generate structure timestamps, to the Generate Repository Schema dialog. If selected, this option adds the MODIFIED keyword and the date and time the structure was last modified to each structure definition in the schema. When the schema is reloaded, this value, rather than the current date and time, will be used as the timestamp value for the structure. This feature is supported in the **rpsutl** utility with the **-t** option, used in conjunction with **-e** (the export option). See "Generating a Repository Schema" in the "Utility Functions" chapter and "Rpsutl Command Line Syntax" in the "Synergy Data Language" chapter of the *Repository User's Guide*. (8.3)
- We added a "defaulting" feature to all main and text file fields in Repository dialogs. After you enter the name of a repository main file (or select it via the browse button) and exit the field, Repository enters a default repository text filename by copying the main filename and changing the last occurrence of the characters "main" to "text". (8.3)
- You can now specify the location of the .new files when merging a schema into an existing repository using the **rpsutl** utility. See "Rpsutl Command Line Syntax" in the "Synergy Data Language" chapter of the *Repository User's Guide*. (8.3)
- We increased the maximum number of fields that can reference a given template from 3,000 to 6,000. If you have a repository that exceeds 3,000, you can either run the version 8.1.7d or higher Verify Repository utility to add the missing reference records, or you can generate your repository to schema and reload it using 8.1.7d or higher. (8.1.7d)
- An error message now displays if you attempt to exceed the maximum number of references that a template can have (6,000) when entering data in the repository. Previously, only the Load Schema utility would detect and prevent an excess, and the Verify Repository utility would report on it, but was not able to correct it. (8.1.7d)
- We added support for a new flag on key definitions. The Excluded by ODBC flag can be used to control a key's inclusion by *xf*ODBC in the system catalog. Set this flag if you want this key to be excluded from the system catalog, which prevents *xf*ODBC from attempting to use it for optimization. (8.1.7)

- Repository now supports defining up to 999 fields per structure. (The maximum number of aliased fields per aliased structure is still 650.) (8.1.7)
- The default Excluded by Web flag value, set when loading fields from an include file, has been changed. NONAME_xxx fields (unnamed filler) are no longer set to be "excluded." (8.1.7)
- Repository now supports the definition of up to 650 fields per structure and 650 aliased fields per aliased structure. (The Connectivity Series **dbcreate** utility and Workbench have been modified to support this new maximum.)

  Previously the maximum was 500, but due to the design, you could never actually have defined the maximum number of fields, keys, relations, local formats, aliases, and tags at the same time because they all shared the same 64K-byte global data area, and 64K is not enough room for all those definitions to co-exist.

  To enable us to support a repository containing the maximum of each definition type, we made each definition type use their own 64K data area. (The maximum fields that can be stored in 64K is 650.) (8.1.5b)
- The internal format of the repository main and text files did *not* change in version 8. Version 8 Synergy/DE tools can open both version 7 and version 8 repositories. However, it is the "created" version that *pre*-version 8 Synergy/DE tools test to determine if they can open a given repository. (The "created" version is the Synergy/DE version that a given repository was created with.) Therefore, pre-version 8 tools cannot open version 8 repositories. Note that modifying a version 7 repository with the version 8 Repository program does not make it a version 8 repository. A version 8 repository is created when you use the "Create new repository" utility, or when you create a new repository via the "Load Repository schema" utility or the **rpsutl** program. (8.1)
- (Windows, UNIX) All repository files are now created with a default page size of 2,048 for improved performance. (8.1)
- We added support for three new attributes to the field and template definitions: display length, view length, and input length. Display length specifies the number of characters that you want to display in a field and overrides the default display length computed by UI Toolkit. View length specifies the number of characters that you want to use to determine the width of the field on the screen and overrides the default width determined by Toolkit. Input length specifies the maximum number of characters you want the user to be able to enter in the field and overrides the default input length computed by Toolkit. (8.1)
- (OpenVMS) The **rpsmain.ism** and **rpstext.ism** files are now installed with protections of world:rwe when the files do not already exist. (8.1)

Repository user interface

- We modified the Key Definition input window to support the new Excluded by ODBC check box. (8.1.7)
- (Windows) You can now enter filenames with up to 255 characters in all of the filename fields in the Repository utilities. UNIX and OpenVMS are still limited to a maximum of 40 characters. (8.1.5)
- We modified the Template Definition and Field Definition input windows to support their new attributes. (8.1)

Subroutine library

- To support the Excluded by ODBC attribute of keys, we added **ki_odbcvw** to the **k_info** record. (8.1.7)
- To support the new display length, view length, and input length for fields and templates, we added **fi_displen**, **fi_viewlen**, and **fi_inplen** to the **f_info** record and **ti_displen**, **ti_viewlen**, and **ti_inplen** to the **t_info** record in the **ddinfo.def** file. (8.1)
- We added a new function to the DD_KEY subroutine, DDK_SLIST, which returns the current structure's key names in sequence order. See DD_KEY in the "Subroutine Library" chapter of the *Repository User's Guide* for more information. (8.1)

Synergy Data Language

- We added the following keywords to the KEY statement: ODBC VIEW and ODBC NOVIEW. (8.1.7)
- We added the following keywords to the FIELD, GROUP, and TEMPLATE statements. (8.1)

  INPUT LENGTH
  NOINPUT LENGTH
  DISPLAY LENGTH
  NODISPLAY LENGTH
  VIEW LENGTH
  NOVIEW LENGTH
- We added the NOSIZE keyword to the GROUP statement. This keyword will be generated when a group field references a template, but wants to keep the size of the group unspecified. (The default behavior is for the group to inherit the size of the template.) (8.1)

Version 7

This section briefly describes new features in Repository version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Repository version 7.

- We modified both the Load Repository Schema utility and the **rpsutl.dbr** program to delete any repository files that they create if any errors occur during the load process. This modification may break your code if you are using **rpsutl.dbr** and testing the existence of the **.new** files to determine if the load succeeded or failed. You should test the exit status from **rpsutl.dbr** instead. (7.1)
- We made numerous changes to the **ddinfo.def** file. Check this distributed file in the RPSLIB directory for details. This modification will break existing code if you have your own copies of any of the record definitions contained within **ddinfo.def**. (7.1)

New features

Repository database

- We added support for an "Excluded by Web?" flag for fields and templates. If Excluded by Web is set, the field is not available to Professional Series WebBuilder as a selectable field, nor will it be included in a type library or Java class created by *xf*NetLink. (7.5)
- We added two new attributes, "display method" and "edit format method," to field and template definitions. Display method specifies the subroutine (method) called by UI Toolkit whenever the field is about to be displayed. Edit format method specifies the subroutine (method) called by UI Toolkit whenever the text in the field is being formatted for editing purposes. (7.3)
- The format of the repository data files has changed. Existing version 6 repositories must be converted using the repository conversion program, **rpscnvt.dbr**, before they can be used with Repository and other version 7 tools (Synergy Language, UI Toolkit, ReportWriter, and *xf*ODBC). See the release notes file, **REL_RPS.TXT**, for detailed information on running **rpscnvt**. (7.1)
- All distributed Repository ISAM files are in REV 4 ISAM format. They will not be accessible by versions of Synergy/DE prior to version 7. These files include **RPSDAT:rpsmain.ism**, **RPSDAT:rpstext.ism**, and **RPSDAT:rpsctl.ism**. (7.1)
- We added support for rendition specifications within field and template definitions. Renditions include color and attributes (highlight, reverse, blink, and underline). (7.1)

- We added two new attributes, “read-only” and “disabled,” to field and template definitions. Read-only renders a field read-only, and its contents cannot be modified. Disabled fields are similar to read-only fields but cannot receive focus. See .FIELD in the “Script” chapter of the *UI Toolkit Reference Manual* for more information about the read-only and disabled attributes. See “Defining display information” in the “Repository” chapter of the *Repository User’s Guide* for more information. (7.1)
- We added a new attribute subtype (class) for user data type field and template definitions. When “User” is selected as the data type, you can now select a subtype—Alpha (default), Numeric, or Date. These subtypes are used by the *xf*ODBC user-defined processing routines and are available in the **gs_inpfld** structure within UI Toolkit’s user-defined processing routines. (7.1)
- We added six new attributes to key definitions: key density, three different key compression options, and segment data type and sort order. (7.1)
- We added seven new attributes to file definitions: record type, page size, density, addressing, compression, static RFA, and portable integer specifications. (7.1)
- We added support for an ODBC table name that can be associated with a given file/structure combination. If specified, this name (rather than the structure name) is used as the table name within the *xf*ODBC system catalog. (7.1)
- We added support for the definition of a group that defines its members by referencing another structure. These are referred to as *implicit* groups. (7.1)
- You can now define a range minimum and maximum for a date or time field. (7.1)
- Repository now validates the size of integer fields and templates. Valid sizes are 1, 2, 4, and 8. (7.1)

Repository user interface

- Since the ODBC Name attribute of fields and templates is used by more Synergy/DE development tools than just ODBC, we changed the name to “Alternate name”. In this version, the prompts in the Repository maintenance program have been changed. (7.5)
- (Windows) The Repository icon is now displayed in the title bar of all input windows, help windows, and lists. (7.3)
- When loading fields from an include file, Repository now creates long descriptions from any additional comment lines found for a field definition. If present, the comment on the field definition line is used as both the short description and the first line of the long description. (7.3)
- (Windows) We added drilldown buttons to the Repository filename fields in the Utilities dialogs. These buttons enable you to browse and select files. (7.3)
- Field input is now preserved when a button or menu entry is selected. This makes it easier to move around within the Field Definition and Template Definition tabbed dialogs because pending input is preserved—even if you don’t move off a field after entering data. (7.3)

- (Windows) We added Close boxes and Minimize and Maximize buttons to all input windows and lists. We added Add, Copy, and Delete buttons to all lists, where appropriate. We also added an Assign button to the File Definition list and input window, and an Attributes button to the Structure Definition list and input window. (7.3)
- (Windows) You can launch Workbench from the Method tab so that you can quickly define methods for fields or templates. Workbench either takes you to the code for an existing method or adds template code for a new method to the end of the method file. (7.3)
- We modified the Template Definition and Field Definition input windows to support their new attributes. (7.3)
- We modified these input windows to support their new attributes: File Definition, Template Definition, Field Definition, Key Definition, and Assigned Structure. (7.1)
- Most Yes/No selection windows have been changed to check boxes. (7.1)
- We removed the one-line status window used to display processing messages. These messages are now displayed in the center section of the footer. (7.1)
- We modified the help windows to no longer use reverse video, in order to improve their appearance on Windows. (7.1)

Repository utilities

- We added a new utility, Compare Repository to Files (**fcompare**), which compares the definitions in the repository to the actual ISAM definitions that they represent. (7.3)
- We modified both the Load Repository Schema utility and the **rpsutl.dbr** program to delete any repository files that they create if any errors occur during the load process. This modification may break your code if you are using **rpsutl.dbr** and testing the existence of the **.new** files to determine if the load succeeded or failed. You should test the exit status from **rpsutl.dbr** instead. (7.1)
- The Load Repository Schema utility now enables you to replace the original repository files when merging schema definitions. (7.1)
- The Generate Repository Schema utility and the **rpsutl.dbr** program now generate an additional line of header information that identifies the export options that were specified. (7.1)
- The loading of repository schemas now supports unquoted method names. (7.1)
- The Generate Repository Schema and Print Repository Definitions utilities now generate an additional line of header information that identifies the Repository version number with which the repository files were originally created. (7.1)

Subroutine library

- In the Repository Subroutine Library, we added **fi_altnm** to the **f_info** record and **ti_altnm** to the **t_info** record in the **ddinfo.def** file. To support backward compatibility, the **f_odbcnm** and **t_odbcnm** fields remain as overlays. (7.5)
- To support the Excluded by Web attribute of fields and templates, we added **fi_webvw** to the **f_info** record and **ti_webvw** to the **t_info** record in the **ddinfo.def** file. (7.5)
- We added support for a new .define, **DDINFO_INGLOBA**L, in the distributed **ddinfo.def** file, for use when including **ddinfo.def** in a global data section. Define DDINFO_INGLOBAL before including **ddinfo.def** to prevent the inclusion of STACK RECORDs and STRUCTUREs. (7.3)
- To support the display method and edit format method attributes of fields and templates, we added **fi_dispmeth** and **fi_editfmtmeth** to the **f_info record**, and **ti_dispmeth** and **ti_editfmtmeth** to the **t_info** record. (7.3)
- Any applications that use the Repository Subroutine Library must be recompiled and relinked with **RPSLIB:ddlib.elb**. (7.1)
- We made numerous changes to the **ddinfo.def** file. Check this distributed file in the RPSLIB directory for details. This modification will break existing code if you have your own copies of any of the record definitions contained within **ddinfo.def**. (7.1)
- To support the renditions attributes of fields and templates, we added **fi_color**, **fi_attrib**, **fi_highlight**, **fi_reverse**, **fi_blink**, and **fi_underline** to the **f_info** record and **ti_color**, **ti_attrib**, **ti_highlight**, **ti_reverse**, **ti_blink**, and **ti_underline** to the **t_info** record. (7.1)
- To support the read-only and disabled attributes of fields and templates, we added **fi_readonly** and **fi_disabled** to the **f_info** record, and **ti_readonly** and **ti_disabled** to the **t_info** record. (7.1)
- We added three new definitions to the **ddinfo.def** file: C_ALPHA, C_NUMERIC, and C_DATE. These definitions should be used when the **fi_type** (or **ti_type**) field equals T_USR. (7.1)
- To support the new key definition attributes, we added **ki_segdtyp**, **ki_segord**, **ki_density**, **ki_cmpidx**, **ki_cmprec**, and **ki_cmpkey** to the **k_info** record. (7.1)
- To support the new file definition attributes, we added **fli_rectyp**, **fli_pagesize**, **fli_density**, **fli_addressing**, **fli_compress**, **fli_staticrfa**, and **fli_portable** to the **fl_info** record. We also added **flsi_rectyp**, **flsi_pagesize**, **flsi_density**, **flsi_addressing**, **flsi_compress**, **flsi_staticrfa**, and **flsi_portable** to the **fls_info** record. (7.1)
- We added support for the passing of an array of **k_info** arguments to the DD_FILESPEC subroutine. (7.1)
- To support implicit groups, we added **fi_struct** and **fi_prefix** to the **f_info** record. (7.1)
- The **si_nmkeys** and **flsi_nmkeys** fields are now **d3**s. (7.1)

Synergy Data Language

- We added the following keywords to the FIELD, GROUP, and TEMPLATE statements: (7.5)

  WEB VIEW
  WEB NOVIEW
  ALTERNATE NAME
  NOALTERNATE NAME

- In the Synergy Data Language (schema), both the old syntax (ODBC NAME and NOODBC NAME) as well as the new syntax (ALTERNATE NAME and NOALTERNATE NAME) are allowed by the "load repository schema" utility. The "generate repository schema" utility will continue to generate the old syntax in version 7.5, but starting with version 8, it will generate only the new syntax. Version 8 will continue to load both old and new. (7.5)

- We added the following keywords to the FIELD, GROUP, and TEMPLATE statements: (7.3)

  DISPLAY METHOD
  NODISPLAY METHOD
  EDITFMT METHOD
  NOEDITFMT METHOD

- We added these keywords to the FIELD, GROUP, and TEMPLATE statements: (7.1)

  READONLY
  NOREADONLY
  DISABLED
  NODISABLED|ENABLED
  COLOR
  NOCOLOR
  HIGHLIGHT
  NOHIGHLIGHT
  REVERSE
  NOREVERSE
  BLINK
  NOBLINK
  UNDERLINE
  NOUNDERLINE
  NOATTRIBUTES
  REFERENCE
  PREFIX

- We added three options to the STORED keyword in the FIELD, GROUP, and TEMPLATE statements: ALPHA, NUMERIC, and DATE. (7.1)

- We added these keywords to the KEY statement: (7.1)

  DENSITY
  NODENSITY
  COMPRESS
  NOCOMPRESS

- We added these keywords to the segment specification of the KEY statement: (7.1)

  SEGTYPE
  SEGORDER
  NOSEGORDER

- We added these keywords to the FILE statement: (7.1)

  RECTYPE
  PAGE SIZE
  DENSITY
  NODENSITY
  ADDRESSING
  COMPRESS
  NOCOMPRESS
  STATIC RFA
  NOSTATIC RFA
  PORTABLE
  NOPORTABLE

- We modified the syntax of the ASSIGN keyword in the FILE statement. The following is the new syntax: (7.1)

  [`Assign` *structure[*`ODBC Name` *name][, structure[*`ODBC Name` *name]][, . . .]]*

Version 6

This section briefly describes new features in Repository version 6 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Repository version 6.

- The "enumerated" attribute can now only be specified for a field or template that has a selection list, selection window, or allow list. This is to prevent its use in conjunction with other attributes such as "check box." This modification may break your Synergy Data Language (formerly IDL) files. You should remove any references to "enumerated" where the field or template does not contain a selection list, selection window, or allow list. This change may also cause some field/template definitions to be invalid when edited through the Repository maintenance program. (6.1)
- Repository no longer allows invalid Synergy Language identifier characters in definition names. To facilitate this change, the **ddcnvt** conversion program replaces all spaces found in definition names with underscores. This modification may break existing schema files. See "Repository database" on page 6-18 for more information. (6.1)
- Because the "format" attribute stored with a field definition is used by both ReportWriter and UI Toolkit, we changed the REPORT FORMAT keyword to FORMAT. Note that version 6 will continue to load both REPORT FORMAT and FORMAT, but it will generate only FORMAT. This also means that all local format definitions must precede any field definitions for a given structure. This modification may break schema files that were created manually. (6.1)
- We enhanced error checking. For details, see the **REL_RPS.TXT** file. This modification may break your schema files. (6.1)
- We made numerous changes to the **ddinfo.def** file. Check this distributed file in the RPSLIB directory for details. This modification will break existing code if you are using the tag information in **s_info** or have your own copy of the **dcs** control structure.
- We added a new subroutine, DD_TAG, for accessing tag information. Tag information is no longer returned with the structure information from DD_STRUCT. This modification will break existing code if you are accessing the tag information that was previously defined in the **s_info** record (**ddinfo.def**). (6.1)

New features

General

- The Professional Series Repository was distributed previously as the Synergy Data Dictionary component of Synergy ICS. Existing ICS version 3 data dictionaries must be converted using the dictionary conversion program, **ddcnvt.dbr**, before they can be used with Repository and other version 6 tools (Synergy Language, UI Toolkit, and ReportWriter). See the release notes (**REL_RPS.TXT**) for version 6 for detailed information on running **ddcnvt.**

  The Repository maintenance program, **rps.dbr**, replaces the Data Dictionary and ICS Utilities portion of the ICS product. In addition to the maintenance program, the Repository distribution also includes three stand-alone utility programs and the repository subroutine library, **ddlib.elb**.

  The three distributed command line utilities are **ddcnvt.dbr**, **rpsutl.dbr**, and **rpsxref.dbr**. The first utility, **ddcnvt.dbr**, is the dictionary conversion utility mentioned above. The second, **rpsutl.dbr**, is the Synergy Data Language loader/generator. The third, **rpsxref.dbr**, is the cross-reference file generator.

  All distributed Repository executables are now dynamically linked against **WND:tklib.elb**. WND must be set to the UI Toolkit directory (done during installation) in order for the Repository programs to run. (6.1)

- With the removal of ICS, a number of filenames, directories, and program logicals have changed. Repository is installed in the RPS directory beneath your Synergy/DE directory. Below RPS are the RPSDAT and LIB directories. (6.1)

  The following filenames have changed:

Filename Changes	
From	**To**
ICS:icsdd.dbr	RPS:rps.dbr
ICS:ddutl.dbr	RPS:rpsutl.dbr
ICS:ddxref.dbr	RPS:rpsxref.dbr
ICS:icsload.ddf	RPS:rpsload.ddf
ICS:icsdd.ico	RPS:rps.ico
ICS:QREF_ICS.TXT	RPS:QREF_RPS.TXT
ICS:REL_ICS.TXT	RPS:REL_RPS.TXT
ICSDAT:icstxt.ddf	RPSDAT:rpstxt.ddf

Filename Changes (Continued)	
From	**To**
ICSDAT:ddctl.wsc	RPSDAT:rpsctl.wsc
ICSDAT:ddctl.ics	RPSDAT:rpsctl.ism
ICSDAT:ddmain.ics	RPSDAT:rpsmain.ism
ICSDAT:ddtext.ics	RPSDAT:rpstext.ism
ICSDAT:icsmap.ics	RPSDAT:rpsmap.ism

The following environment variables have changed:

Environment Variable Changes	
From	**To**
ICS	RPS
ICSDAT	RPSDAT
ICSMFIL	RPSMFIL
ICSTFIL	RPSTFIL
ICSXFIL	RPSXFIL

- The Repository's text messages are now read from the file **DBLDIR:syntxt.ism** using the UI Toolkit subroutine U_GETTXT. This file is opened by Toolkit during U_START. The SYNTXT logical can be used to override the location of the **syntxt.ism** file. The facility for the Repository's messages is RPS. The distributed sequential file containing the repository messages is **RPSDAT:rpstxt.ddf**. (6.1)
- A new environment variable, RPSTMP, enables you to control the location of temporary work files used by Repository. Use RPSTMP to specify the directory in which all Repository temporary files will be created. (6.1)

Repository database

- We added a new attribute, “change method,” to field and template definitions. This value specifies the subroutine (method) name invoked by UI Toolkit after field validation. (6.3)
- We added a new attribute, “ODBC name,” to field and template definitions. This value specifies the column name for the field within the ODBC catalog. (6.3)
- (Windows) We added two new attributes, “font” and “prompt font,” to field and template definitions. These values specify the input field and input prompt fonts, respectively, to be used on Windows. (6.3)
- The “enumerated” attribute can now only be specified for a field or template that has a selection list, selection window, or allow list. This is to prevent its use in conjunction with other attributes such as “check box.” This modification may break your Synergy Data Language (formerly IDL) files. See “Features and fixes that may break your existing code” on page 6-15 for more information. (6.1)
- Repository no longer allows invalid Synergy Language identifier characters in definition names. An invalid character is any character other than A-Z, 0-9, $, or “_”. Definition names containing any other characters, or names that don’t begin with a letter, will generate an error.

  To facilitate this change, the **ddcnvt** conversion program replaces all spaces found in definition names with underscores. The **DDCNVT.LOG** file reports all name changes. To determine if you have other invalid characters in any of your definition names, run the Validate Repository utility. You can then generate your repository to a schema, make the changes, and then reload it. This modification may break existing schema files. (6.1)
- The Repository now supports the following increased maximums: (6.1)

Item changed	From	To
Size of field	9999	99999
Number of structures	3782	9999
Number of keys per structure	32	99
Levels of template inheritance	3	unlimited
Maximum field and prompt positions (specified in the Display Information window)	99	999

- We increased the maximum number of tag definition criteria per structure from 2 to 10. Each criterion can now reference a different field. However, this extension was made in anticipation of extensions to other Synergy/DE products. Synergy/DE ReportWriter and Synergy/DE *xf*ODBC still only support a maximum of two criteria that reference the same field. (6.1)
- We reduced the maximum number of selection list entries from 100 to 99 to match UI Toolkit. (6.1)

- We added a date and time stamp to file, structure, template, and global format definitions. This information can be accessed with the repository subroutine library, **ddlib.elb**. (6.1)
- We added a new attribute, "temporary," to file definitions. If a file definition is defined as temporary, it will be excluded from the list of available files in ReportWriter. (6.1)
- We added two new view flags, similar to the existing report view flag, for fields and templates. If the field/template is not "viewable" by Synergy Language, it will not be included by the Synergy compiler when you use the .INCLUDE REPOSITORY compiler directive. If the field/template is not "viewable" by UI Toolkit, it will not be accessible by the Script compiler, Window Designer, or Composer. (6.1)
- A short description can now be specified for key definitions. (6.1)
- We increased the size of the range minimum and maximum fields in **ddinfo.def** to 28.10; however, Repository will only allow specification of a **d18** until the full 28.10 is supported by UI Toolkit. (6.1)
- We added a new binary data type for fields and templates. Binary fields are stored as alpha fields with the class "binary." This data type was added for users of Connectivity Series. (6.1)
- The following Script field qualifiers are now supported: radio buttons, check boxes, wait, and input field methods (arrive, leave, drill, and hyperlink). These can be specified for templates and fields. (6.1)
- The user data string for a field or template can now be overridden even if the data type has not been. For example a user-defined field can reference a user-defined template yet override the user data string. This feature was previously supported only through the Load Repository Schema utility. (6.1)
- You can now specify a paint value of "None" or "Character." Previously there was no way to indicate that the field contained *no* paint character, as opposed to containing a paint character of *blank*. This meant that UI Toolkit always had to assume it was a paint character of blank, and hence there was no way for the input window's paint character to be in effect. (6.1)
- You can now define a Synergy Language group in the Repository by
  - defining the group and its members using the Field Definitions list.
  - loading a record containing groups from an include file using the "Load fields" function.
  - defining the group and its members using the Synergy Data Language.

  You can then .INCLUDE records containing groups into Synergy Language programs. The new "language view" flag prevents a field's inclusion in the Synergy Language record definition. The purpose of this feature is to exclude overlay fields that have been added solely for the purpose of referencing fields within groups by ReportWriter, Script, Composer, and Window Designer. (See the following list.)

When using groups, note the following:

- Fields within groups cannot be used as key segments, tag fields, or aliased fields. Only fields (or groups) defined at the highest (structure) level can be used for these purposes. Overlay fields that overlay group members can be defined at the highest level.
- Access to fields within groups by ReportWriter, Script, Composer, and Window Designer is only possible using overlay field definitions. The "report view" and "script view" flags can be used to prevent a group or field's inclusion in a report or window definition.
- Overlays specified on field definitions within a group must be relative to another field within that same level.
- The Generate Cross-Reference File utility only supports the name linking of fields at the highest (structure) level.

(6.1)

Repository user interface

- The user interface for field and template definitions now uses a single tabbed input window instead of five separate input windows. (6.3)
- We added termination processing to the main Repository input windows. When the user completes input, the message "Make corrections or press CTRL to complete input" will appear on the information line, instead of the cursor returning to the first input field. This change was made in order to be consistent with other Synergy/DE applications. On Windows, standard buttons, which override the termination message, have been added to the main Repository input windows. (6.1)
- We moved the Abandon menu entry from the various Input and Edit menus to the General menu. This allows the Input and Edit menus not to be placed when running on Windows, since they now contain nothing but reserved menu entries. (6.1)
- To handle newly supported field and template attributes, we redesigned the Script and Report Information input windows. These windows have been replaced by the Display, Input, Method, and Validation Information input windows. Display Information contains all attributes related to how a field is displayed by UI Toolkit or ReportWriter. Input Information contains all attributes related to performing Toolkit input to a field. Method Information contains all method specifications. Validation Information contains all attributes related to Toolkit input field validation. (6.1)
- The template override fields, which used to be one-character fields following each field/template attribute (made visible with the "View template overrides" menu entry), are now a set of check boxes at the bottom of the various field/template input windows. These check boxes can be enabled and disabled. The new "Access template overrides" menu entry is a global setting. Once this is turned on for a field or template, it remains on for all fields/templates until turned off. (6.1)

- We changed the user interface for tag definitions to match that of fields, keys, and so forth. The specification of whether or not a tag exists and its type has been moved to the structure definition. (6.1)
- The user interface for the maintenance of groups was added. To enable you to define a group, a new Group? field has been added to the Field Definition input window. Once this field has been set to "Yes," you may select "Edit group members" (CTRL+G) to display a list into which you can insert group member definitions. This is a recursive mechanism, thereby allowing you to define an unlimited number of nested groups. (6.1)
- The shortcut for "Load fields" is now F9 instead of CTRL+G. We made this change to allow CTRL+G to be used as the shortcut for "Edit group members." (6.1)
- A help window, **h_general**, is displayed when you select the Help button from any of the Repository input windows. (6.1)
- We removed the "All relations for all structures?" field from the Generate Repository Schema input window. This field was for compatibility with older versions only and duplicated functionality provided by other input window options. (6.1)

Synergy Data Language

- We added the following keywords to the FIELD and TEMPLATE statements: (6.3)

  ODBC NAME
  NOODBC NAME
  FONT
  NOFONT
  PROMPTFONT
  NOPROMPTFONT
  CHANGE METHOD
  NOCHANGE METHOD

- We added the following keyword to the FILE statement: NOTEMPORARY. (6.3)
- Because the "format" attribute stored with a field definition is used by both ReportWriter and UI Toolkit, we changed the REPORT FORMAT keyword to FORMAT. Note that version 6 will continue to load both REPORT FORMAT and FORMAT, but it will generate only FORMAT. This also means that all local format definitions must precede any field definitions for a given structure. This modification may break schema files that were created manually. (6.1)

- We added the following keywords to the FIELD and TEMPLATE statements: (6.1)

  LANGUAGE VIEW
  LANGUAGE NOVIEW
  SCRIPT VIEW
  SCRIPT NOVIEW
  WAIT *time*
  WAIT GLOBAL
  WAIT FOREVER
  WAIT IMMEDIATE
  NOWAIT
  RADIO
  CHECKBOX
  NORADIO
  NOCHECKBOX
  ARRIVE METHOD
  NOARRIVE METHOD
  LEAVE METHOD
  NOLEAVE METHOD
  DRILL METHOD
  NODRILL METHOD
  HYPERLINK METHOD
  NOHYPERLINK METHOD

- The new binary data type is generated as the following: (6.1)

  ```
  Type ALPHA   Stored BINARY
  ```

- We added the following keyword to the FILE statement: TEMPORARY. (6.1)
- We added a short description to the KEY statement: (6.1)

  ```
  [Description "description"]
  ```

- The TAG statement is now generated as follows: (6.1)

  ```
  Tag type field  EQ | NE | LE | LT | GE | GT  value
  [ AND | OR  field  EQ | NE | LE | LT | GE | GT  value...]
  ```

  Note that the version 3 tag syntax is still supported when a schema is loaded.

- New GROUP and ENDGROUP keywords are used to designate the beginning and end of a group element. The GROUP statement syntax is almost identical to the FIELD statement, with two exceptions: SIZE is optional for a GROUP, and OVERLAY has no arguments. (6.1)
- We enhanced error checking. For details, see the **REL_RPS.TXT** file. This modification may break your schema files. (6.1)

Subroutine library

- We added two new definitions to the **ddinfo.def** file. FI_VW and FI_NVW are to be used when testing the **fi_dblvw**, **fi_rptvw**, and **fi_scrptvw** and corresponding **ti_*xxx*** fields. These definitions are very important, because—unlike most other yes/no fields—a value of 0 represents Yes and a value of 1 represents No. (6.3)
- To support the change method attribute of fields and templates, we added **fi_changemeth** to the **f_info** record and **ti_changemeth** to the **t_info** record. (6.3)
- To support the ODBC name, font, and prompt font attributes of fields and templates, we added **fi_odbcnm**, **fi_font**, and **fi_promptfont** to the **f_info** record and **ti_odbcnm**, **ti_font**, and **ti_promptfont** to the **t_info** record. (6.3)
- We made numerous changes to the **ddinfo.def** file. Check this distributed file in the RPSLIB directory for details. This modification will break existing code if you are using the tag information in **s_info** or have your own copy of the **dcs** control structure. Other changes include the following:
  - The main and text file channels have changed from **i2**s to **i4**s.
  - The **fi_selinfo** field is now **fi_sellst**.
  - The information previously returned in **fti_selinfo** is now returned in **fi_seltyp**, **fi_selrow**, **fi_selcol**, **fi_selwin**, and **fi_selht**.
  - The field **fti_sellst** is now **fti_entlstary**. You can also use **fti_entlstary** to return allow list entries.

  You must update all copies of **dcs** to match the one distributed in **ddinfo.def**. (6.1)
- We added two new optional arguments to DD_INIT. They are the names of the main and text file that are opened. (6.1)
- If a call to DD_INIT fails, one of three possible error codes is now returned: (6.1)

  E_OPNERRM Cannot open the repository main file.

  E_OPNERRT Cannot open the repository text file.

  E_BADVERS Repository is not a version 6 repository.
- We added a new subroutine, DD_CONTROL, that enables you to access repository control record information. This includes the repository version and the date and time of the last repository modification. (6.1)
- We added two new subfunctions, DDF_GROUP and DDF_ENDGROUP, to the DD_FIELD subroutine. These subfunctions will allow you to access group fields in Repository. (6.1)

- To support groups, we added a new field to the **s_info** record. The **si_childct** field is the number of fields and groups defined at the structure level. The **s_nmflds** field will now represent the total number of fields/groups at all levels of the structure. Note that for the **f_info** record, **fi_seqnm** now designates the field's sequencing within its member group (or structure); it does not represent its sequencing among all levels of the structure. In other words, the sequence number of the first member of any group is 1. MAXFLDS is still the maximum number of total fields/groups for a structure. Also, the **fi_pos** value for group members is relative to the parent group. (6.1)
- We added a new subroutine, DD_TAG, for accessing tag information. Tag information is no longer returned with the structure information from DD_STRUCT. This modification will break existing code if you are accessing the tag information that was previously defined in the **s_info** record (**ddinfo.def**). (6.1)

Utilities

- The Generate Definition File utility now omits overlay fields that are flagged as non-viewable to the Language. It will create unnamed fields as placeholders for any non-overlay non-viewable fields to maintain the integrity of the record definition. (6.3)
- You can now change repositories from within the Repository maintenance program. The Set Current Repository entry on the Utilities menu will temporarily change RPSMFIL and RPSTFIL to the repository specified, without having to exit the program. Also, the Create New Repository utility now enables you to indicate if the repository being created should be set as the current repository. (6.1)
- The Generate Definition File utility now allows you to specify the repository filenames. (6.1)
- You can view your group definitions using the Print Repository Definitions, Generate Repository Schema, or Generate Definition File utilities. (6.1)
- We removed RECORD, COMMON, and other text strings used in the Generate Definition File utility from the text message file. This is to prevent users from accidentally modifying these strings, since they must be understood by the Synergy compiler. (6.1)

Version 3

This section briefly describes new features in Repository version 3 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to Repository version 3.

- The ALIAS keyword must now be grouped along with the structure that it aliases. This modification may break your existing IDL files. (3.5)
- REPORT VIEW is now the default mode for fields. This keyword is not generated unless it is overriding a template's REPORT NOVIEW setting. This modification may break your IDL files if you have removed this attribute for fields that are to have NOVIEW. (3.1)
- The Generate Dictionary Schema utility now outputs each alias along with the structure it aliases, rather than after all structures. This modification may break your existing IDL files. If you have existing IDL files that contain alias definitions, modify the files to group the aliases along with their "aliased" structure. See "Utilities" on page 6-29 for more information. (3.5)

New features

General ICS

- We added two new key maps to the distributed key map file (**icsmap.ics**): XTERM (for X-windows terminals) and WY60 (for Wyse-60 compatibles). (3.5.4 and 3.7.4)
- The "Delete word" shortcut has been changed to CTRL+U. (3.5)
- We added a new environment variable called ICSXFIL. If ICSXFIL is defined, the dictionary cross-reference filename is the value of ICSXFIL. This file is used to add related files to a report, based on dictionary name links. (3.5)
- We changed the **ICSDAT:dictmain.ics** and **ICSDAT:dicttext.ics** default dictionary filenames to **ICSDAT:ddmain.ics** and **ICSDAT:ddtext.ics**. You can still use your own naming convention by using the ICSMFIL and ICSTFIL environment variables. (3.1)
- ICS now supports an implied-decimal data type, which is a decimal field that has a precision value. Implied-decimal replaces the fixed-point data type. If no precision value is specified, the field is a normal decimal field. (3.1)
- ICS now supports an integer data type. (3.1)

Data Dictionary

- We added four new attributes to the Key Definition input window: “Insert dups at,” “Modifiable?,” “Null key?,” and “Null key value.” (3.5.6 and 3.7.6)
- A timestamp is now maintained in the control record of the Data Dictionary main file. (3.5.6 and 3.7.6)
- Precision is now only allowed with decimal and user-type fields. Previously, it was allowed on any data type. This modification may invalidate existing data dictionaries. To determine if you have any fields on which a precision value was erroneously allowed, run the ICS Validate Dictionary utility. Then manually update any necessary fields and templates. (3.5)
- We added a flag to field and template definitions to determine name links. (3.5)
- We changed the character used in alpha formats from “A” to “@.” The dictionary conversion program converts all of your alpha formats. (3.1)
- The Data Dictionary now supports the following increased maximums and definition name sizes:

Item changed	Old	New
Field name size	15	30
Field report heading size	25	40
Field size (or size of one array element)	999	9999
File name size	8	30
Global format name size	8	30
Implied-decimal precision	9	10
Key name size	15	30
Key size	99	255
Local format name size	8	30
Number of fields per structure	250	500
Record size	9999	99999
Segment size (literal type)	15	30
Short description size	25	40
Structure name size	8	30
Template name size	8	30

- There is now no limit to the number of users who can view a structure at the same time. (3.1)
- The ENUMERATED script qualifier no longer requires that you have an allow list or selection list or window defined. You can associate a selection window that's built on-the-fly with the input field at runtime. (3.1)
- You can now specify the justification for formats, which will affect how a format is truncated when applied to a field. (3.1)
- We added three new field and template script qualifiers: RETAIN POSITION, TIME NOW, and TIME AMPM. (3.1)
- A template can now reference another template. A template that is referenced by another template is called a parent. When a field or template references a template, the attributes of that template are inherited. You can override any or all attributes. (3.1)
- Version 2.*x* data dictionaries must be converted for use with version 3 using the **ddcnvrt.dbr** program distributed with version 3. (3.1)

ICS Definition Language

- We added the INSERT, MODIFIABLE, NULL, and VALUE keywords to keys. (3.5.6 and 3.7.6)
- The ALIAS keyword must now be grouped along with the structure that it aliases. This modification may break your existing IDL files. (3.5)
- The keywords that used to be ignored when used with an incorrect data type now cause an error when the dictionary is validated. (For example, if you specify the RANGE keyword with an alpha field, an error will be generated.) You can run the new Validate Dictionary utility to check for these occurrences. (3.5)
- We added the NONAMELINK keyword to fields and templates. This keyword overrides the default name linkage to the parent template. (3.5)
- REPORT VIEW is now the default mode for fields. This keyword is not generated unless it is overriding a template's REPORT NOVIEW setting. This modification may break your IDL files if you have removed this attribute for fields that are to have NOVIEW. (3.1)
- We added the ALIAS keyword, which defines an alias field or structure. (3.1)
- The DECIMAL type for formats has been changed to NUMERIC. (3.1)
- The FIXED data type is no longer supported. ICS now supports implied-decimal fields. Specify implied-decimal fields using DECIMAL as the data type, then specify the number of digits to the right of the decimal point.
- The keyword for specifying the number of digits to the right of the decimal point has been changed from DECIMAL to PRECISION. The PRECISION keyword is valid only when the field's data type is decimal or user. (3.1)
- The DUPS and ORDER keywords are no longer generated if the key is a foreign key. (3.1)
- The keyword for specifying the echo character for a field or template has been changed from ECHO to NOECHOCHR. (3.1)

- We added three new keywords for field and template definitions: TIME NOW, TIME AMPM, and RETAIN POSITION. (3.1)
- We added the following ICS Definition Language keywords to override attributes of a referenced template: (3.1)

NODATE	NOTERM
DATE NOSHORT	NOAUTOMATIC
DATE NOTODAY	NODEFAULT
NOTIME	NOFORMAT
TIME NONOW	NOINFO
TIME NOAMPM	NOUSER TEXT
NOPOSITION	NOUSER TYPE
NOFPOSITION	NOHELP
NOPROMPT	NOALLOW
NOUPPERCASE	NOSELECT
NOBLANKIFZERO	MATCH NOCASE
NONEGATIVE	MATCH NOEXACT
NORETAIN POSITION	NORANGE
DECIMAL_REQUIRED	NOENUMERATED
ECHO	REPORT NOVIEW
NOPAINT	NODESC
NOREQUIRED	NOLONGDESC
NOBREAK	NOHEADING

- If a field references a template and doesn't override any attributes of that template, you need to specify only the field name and the template name in your FIELD statement. For example,

```
Field AMOUNT    Template MONEY
```

  You can also specify any attributes that override the template. (3.1)

- For consistency, FORMATS must now precede the "type" data with a TYPE keyword. (3.1)
- The TYPE and SIZE keywords for the FIELD and TEMPLATE statements must precede any other keywords (with the exception of a referenced TEMPLATE or PARENT). (3.1)
- A field description that doesn't reference a template or that overrides a referenced template's type or size must now precede the *type* and/or *size* data with the keywords TYPE or SIZE, respectively. A format description must also precede the *type* data with the keyword TYPE. (3.1)

Subroutine libraries

- The DD_KEY subroutine has a new function, DDK_TEXT, which returns text information about a key. (3.5.6 and 3.7.6)
- In record *dcs* of the **ICSLIB:ddinfo.def** file, *mchn_r* and *tchn_r* are now **i2** fields. This modification may affect your code if you define local copies of the dcs structure. (3.5.4 and 3.7.4)
- We added the DD_ALIAS subroutine, which returns general information about an alias structure. (3.5)
- The DD_FIELD subroutine has a new function, DDF_SLIST, which returns the current structure's field names in sequence order. (3.5)
- The DD_STRUCT subroutine (DDS_INFO function) has been enhanced to allow the passing of an alias structure name. If an alias structure name is passed, the name of the aliased structure can be returned to the caller. (3.5)
- The DD_FORMAT subroutine now returns either global or predefined format names. If you pass DDM_INFO, DD_FORMAT first looks for a predefined date or time format name and then a global format name. (3.1)
- The DD_NAME subroutine has two new functions, DDN_DFMT and DDN_TFMT, which enable you to find out the number of predefined date and time formats in a dictionary and to get a list of their names. (3.1)
- Various fields in the **ddinfo.def** file have been modified. If you pass your own data structures to the DD_ subroutines, you will need to modify your structures. Please review the new **ddinfo.def** structures carefully. (3.1)
- The DDL_LIST function of the DD_FILE subroutine is now called DDL_STRS. It returns a list of the structures assigned to a file. (3.1)
- The DDS_LIST function of the DD_STRUCT routine is now called DDS_FILS. It returns a list of the files to which a structure is assigned. (3.1)
- The definition file to .INCLUDE when using the subroutine library is now **ICSLIB:ddinfo.def**. (3.1)

Utilities

- The Print Dictionary Definitions utility now includes a module for printing tag information so that structure tag information can be printed in the output file. Print Dictionary Definitions no longer prints the Order and Dups values if the key is a foreign key. (3.5.6 and 3.7.6)
- We added the Generate Cross-Reference Utility, which creates an ISAM file that matches fields with access keys based on dictionary name links. (3.5)

- The Generate Dictionary Schema utility now outputs the following:
  - Relations along with their defining structure, rather than after all structures. (3.5)
  - Each alias along with the structure it aliases, rather than after all structures. This modification may break your existing IDL files. If you have existing IDL files that contain alias definitions, modify the files to group the aliases along with their "aliased" structure. (3.5)
- The Load Dictionary Schema utility error messages that are output to the log file no longer include the line number, but they more uniquely identify the structure, file, field, and so forth. (3.5)
- We added full support for merging ICS schema files into existing data dictionaries. (3.5)
- We added the Validate Dictionary utility, which can be used after the Load Schema utility to verify the integrity of each definition. (3.5)
- The Generate Definition File utility now generates one-dimensional arrays as "bracketed" arrays, rather than "pseudo" arrays. For example, **[5]a10** instead of **5a10**. (3.3)
- You can now include wildcards (* or ?) when specifying definition names in the Print Dictionary Definitions and Generate Dictionary Schema utilities. (3.1.1)
- You can now generate partial schema files with the Generate Dictionary Schema utility by specifying individual structure, file, template, or format names. (3.1.1)
- The Load Report Definitions and Unload Report Definitions utilities are now part of Report Writer. (3.1)
- The ICS utilities support the following increased sizes: (3.1)

Item changed	Old	New
Filename fields	30	40
Generate Definition File header name	15	30
Generate Definition File prefix string	8	15
Generate Definition File global data section name	15	30

7

ReportWriter

Version 7

This section briefly describes new features in ReportWriter version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following change may break code when you update to ReportWriter version 7.

- (VAX OpenVMS) From version 7.1.3b onward, the **SYNRPT.EXE** shared image on VAX systems is correctly linked with a vector table. Prior 7.1 versions of the shared image did not link against this vector table. If you are migrating from a prior 7.1 version, you must relink any application that is linked against **SYNRPT.EXE**. (7.3.3)

New features

- ReportWriter now allows users to use a comma as the decimal separator in an expression. (7.5.1)

Version 6

This section briefly describes new features in ReportWriter version 6 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

Environment variables

- We added the SYNCENTURY environment variable, which specifies the two-digit year that will be used to determine the default century. SYNCENTURY defines a "sliding window" for Synergy/DE default century methods by allowing certain years to use the current system century and other years to use the next or previous century. (6.3)

Repository-related modifications

- ReportWriter now uses the environment variables RPSMFIL, RPSTFIL, RPSXFIL, and RPSDAT when searching for Repository data files. The previously supported environment variables were ICSMFIL, ICSTFIL, ICSXFIL, and ICSDAT, respectively. (6.1)
- With the addition of group support in the Professional Series Repository, ReportWriter now enables you to include a group, defined at the structure level, as a single field. ReportWriter does not support the access of group members. To access group members, overlay fields that offset into the desired group must be defined in Repository. (6.1)
- ReportWriter supports the new "temporary" flag in Repository file definitions. If a file has this flag set, ReportWriter will not include it in the list of Available Files. (6.1)

User interface

- The "Abandon" menu entry is now on the General menu and is only enabled when appropriate. (6.1)
- Report Definition Language files that contain invalid temporary field names are no longer supported. Valid names start with an alpha character, contain no more than 30 characters, and contain only alpha, numeric, underscore, or dollar sign characters. If any of your reports have invalid temporary field names (for example, names that contain spaces), do the following:

  1. Generate the reports to a schema (Report Definition Language) file.
  2. Change the invalid names to valid names.
  3. Reload the reports.

  (6.1)
- Larger reports are now supported during the report execution phase, which enables you to create and generate more complex reports. (6.1)

Version 3

This section briefly describes new features in ReportWriter version 3 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

Break lines

- You can now specify that the report is to include the number of breaks that occurred for each break level. (3.5)
- You can use the new intrinsic %BRKCNT in a calculation field to access the number of different values that have been processed for a given break field. (3.5)
- In addition to specifying print conditionals for pre- and post-break print fields, you can now assign a conditional to the break line itself. (3.1)
- You can suppress the blank line that precedes a post-break line. (3.1)
- You can specify the number of blank lines to be printed between break sets. (3.1)

Detail lines

- You can now specify print conditionals for one or more lines within a multi-line detail record. (3.5)

Displaying reports

- When a report is displayed on the screen, you can now use menu entries to view the pages of that report. Pressing the Exit shortcut will end the display function and return you to the menu. (3.5)

Environment variables

- We added the environment variable ICSXFIL. If ICSXFIL is defined, the dictionary cross-reference filename is the value of ICSXFIL. This file is used to add related files to your report, based on dictionary name links. (3.5)
- We added the environment variable ICSRFIL. If ICSRFIL is defined, the report definition filename is the value of ICSRFIL. (3.1)
- You can set the new environment variable RPTEURO to cause Report Writer to use European formatting for numeric values (commas and periods reversed). (3.1)

Field specifications

- You are no longer allowed to create temporary field names that contain spaces. (3.5 and 3.7)
- When prefixing a field name with its file and/or structure name, the delimiter is now a period (.) instead of a colon (:). Because of this, the %SUM intrinsic function now has the following form:

  `%sum(`*field*`(`*beg*`:`*end*`))`

  The report conversion program converts all such occurrences. (3.1)

Filenames

- We changed the way Report Writer creates temporary filenames. Instead of calling TNMBR, it now calls JBNO to get the numeric portion of the filename. (3.7.9)
- The default dictionary filenames are now **ICSDAT:ddmain.ics** and **ICSDAT:ddtext.ics** instead of **ICSDAT:dictmain.ics** and **ICSDAT:dicttext.ics**. You can still use your own naming convention by using the ICSMFIL and ICSTFIL environment variables. (3.1)

Files to read

- You can now add secondary files to your report based on dictionary "name links" rather than relations. Possible name links are stored in the file **ICSDAT:ddxref.ics**, which is created by the ICS Generate Cross-Reference utility. (3.5)

Formats

- We changed the character used in alpha formats from "A" to "@." The report conversion program converts all of your alpha formats. (3.1)
- You can now specify the justification for formats, which affects how a format is truncated when applied to a field. (3.1)

Headers and footers

- You can now print fields in your report or page headers and footers. You can modify the formats of these fields and assign conditionals to them. (3.1)

Integers

- Integers are supported in data files processed by Report Writer. (3.1)

Maximums

We increased the following Report Writer maximums: (3.1)

Changes to Report Writer Maximums		
Item changed	**Old**	**New**
Calculation field expression size	80	150
Calculation fields per report	30	99
Conditionals per report	30	99
Criteria per conditional	4	5
Enumerated fields per report	20	99
Environment fields per report	30	99
Field header size	25	40
Fields per detail line	30	99
Fields per detail record	100	990
Fields per header line	N/A	99
Lines per detail record	6	10
Post-break print fields	10	99
Pre-break print fields	10	99
Question fields per report	30	99
Range references	N/A	300
Report name size	25	40
Report or page header or footer size	1194	2550
Report width	199	255
Selection criteria per report	5	25
Sort field breaks per report	5	10
Subscript references	100	300
Subtotal access fields per report	30	99

Changes to Report Writer Maximums (Continued)		
Item changed	**Old**	**New**
Temporary field description size	25	40
Temporary field name size	8	15
Temporary fields per report	100	495
Text fields per report	30	99

Question fields

- Question fields of the "Inherit" data type now inherit more attributes from the parent field. (3.1.6)

Report definitions

- Version 2.5 or 2.7 report definition files must be converted for use with version 3, using the **rptcnvrt.dbr** program distributed with version 3. (3.1)

Report Definition Language

- The following modification may break your existing RDL files. The RDL (Report Definition Language) generation and loading of the LT and LE selection/conditional operators were swapped. However, because they were both generated and loaded incorrectly, reports reloaded from RDL were correct. For example, a report that used LT was generated in the RDL as LE. When that RDL report was re-loaded, it was converted back to LT. So unless you looked closely at the RDL, it would go unnoticed as a problem. *If you have existing RDL files, examine them closely for the occurrence of LE and LT. Correct them before reloading with V3.1.7 of the Report Writer or they will get swapped!* (3.1.7)
- We developed a definition language for Report Writer called the Report Definition Language (RDL). We also added two utilities to generate and load the RDL. (3.1)

Report modification dates

- You can now toggle the report name selection list to show the report creation and modification dates. (3.1)

Report output

- We added a Worksheet output option to the Generate report file function. This option generates an export file for applications such as spreadsheets. (3.5)

Sorting

- (OpenVMS) The maximum number of fields to sort has been changed to 9. (3.5)

Subscripted and ranged fields

- Due to the possibility of an "Invalid subscript" error or a "Size of integer field must be 1, 2, 4, or 8" message, Report Writer no longer allows integer fields to be ranged. If you have a report that is ranging on an integer field, delete the field from the print line or conditional and then re-insert it. (3.5.6 and 3.7.6)
- You can now subscript in more instances when selecting an arrayed field.

  An exception is specifying an arrayed field within a subtotal access field definition, within a field range definition, or within a conditional. To use a specific array element in such cases, you must subscript within a calculation field and then specify the calculation field in the subtotal access field, range, or conditional definition. (3.1)
- You can now range any selected field, specifying a subset of the data. This ranging can be fixed (for example, NAME(1:3)) or variable (for example, NAME(1:NAME_LEN)).

  An exception is ranging an arrayed field. To range an arrayed field, you must subscript within a calculation field and then range the calculation field. (3.1)

User-replaceable subroutines

- We added the subroutine USR_RW_INIT to enable you to overload Toolkit methods within Report Writer. USR_RW_INIT is called immediately after Report Writer starts. You can overload Toolkit methods within this routine using the E_METHOD subroutine. (3.5.6 and 3.7.6)
- We changed the way Report Writer handles the user-defined subroutine USR_RW_GETVAL. Previously, Report Writer called this subroutine once for each file that was selected in a report instead of once for each user-defined data type field in each record. (3.5.4 and 3.7.4)
- Two new user-replaceable subroutines, USR_RW_GETVAL and USR_RW_USAGE, enable you to provide a mechanism for the use of user-defined data type fields in Report Writer. (3.5)
- Report Writer has a new user-replaceable subroutine, USR_RW_LSTINFO. This subroutine is invoked by a menu entry selected while the list of available fields, files, structures, or relations is displayed. You can use this routine to provide more detailed information about the item that is currently highlighted. (3.1)

User interface

- The "Arrange field position" shortcut has been changed to CTRL+N. The "Delete word" shortcut has been changed to CTRL+U. (3.5)

Starting Report Writer

- Report Writer now allows users the option to continue or quit when either an invalid start-up message is received or the message controller (on OpenVMS) is not running. (3.5.6 and 3.7.6)
- We added three new parameters to the argument string for SPAWNing Report Writer: *page_header*, *separator*, and *name_link_file*.

Subroutine library

- We added the Report Writer subroutine RW_GETRPT, which enables you to get either a sequence of report names, or all reports in a given reports file. (3.5.4 and 3.7.4)

8

*xf*ODBC

Version 9

This section briefly describes new features in *xf*ODBC version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*ODBC version 9.

- With version 9.5, the Synergy/DE Data Provider for .NET no longer supports Visual Studio 2008 or .NET Framework 3.5. It now supports only Visual Studio 2010 and .NET Framework 4. Note the following:
  - With Visual Studio 2010, you may be able to use an entity data model (EDM) created with Visual Studio 2008, provided you import the entire project with the Conversion Wizard. Note however, that this does not always work, particularly for complex EDMs. In some cases, you may need to delete and regenerate the EDM.
  - The Entity Data Model Wizard for Visual Studio 2010 includes a new option, "Pluralize or singularize generated object names" that could break code written for EDMs generated in Visual Studio 2008. When selected (which is the default), this option instructs the wizard to automatically pluralize or singularize names in the resulting model. For example, the PLANTS table in the sample database becomes PLANT in an EDM.

  (9.5)

New features

Dbcreate and the *xf*ODBC Database Administrator (DBA)

- We added a field, Signed, to the Column window of the *xf*ODBC Database Administrator (the DBA program). This field indicates whether a numeric column in the system catalog is signed or unsigned (which is determined by whether "Negative allowed" or a positive range of values is selected for the corresponding field in the repository). (9.3.1a)
- We updated *xf*ODBC to support the following new Repository data types: Boolean, user-defined binary, and struct. (**Dbcreate** does not support the new Repository enum data type. Instead it generates integer columns for fields with this data type, which provides ODBC access only to the integer values for enum fields.) (9.3)
- We updated **dbcreate** to use the new Repository "Null allowed" field to determine how to set the "Null allowed" property for system catalog columns. If the Repository "Null allowed" field is set to Yes, the "Null allowed" property of the system catalog column is set to Y (for Yes). If set to No, the "Null allowed" property is set to N (for No). If the "Null allowed" field is set to Default, the SODBC_NONULL setting determines how the "Null allowed" property is set for the system catalog column. (9.3)

- We updated **dbcreate** for SODBC_NONULL environment variable changes:
  - **Dbcreate** now uses the SODBC_NONULL setting only when generating system catalog columns from repository fields whose "Null allowed" setting is Default. See entry above.
  - When SODBC_NONULL is set to 0, **dbcreate** now sets the "Null allowed" property to Y for all system catalog columns generated from repository fields with "Null allowed" settings of Default. (Previously, 0 caused **dbcreate** to set the "Null allowed" property to Y for all columns except date columns.)
  - We updated **dbcreate** for the new data types. When SODBC_NONULL is set to 4, **dbcreate** now sets the "Null allowed" property to Y for all columns generated from fields whose "Null allowed" setting is default, except integer columns, binary columns, Boolean columns, and non-date columns that are part of the first key. (Previously with this setting, **dbcreate** set the "Null allowed" property for all columns to Y except for non-date columns that were part of the first key.)
  - The default for SODBC_NONULL is now 4. Previously, if you didn't set SODBC_NONULL, **dbcreate** set the "Null allowed" property for all system catalog columns to Y.

  Note that these changes may change "Null allowed" settings for columns in a system catalog if you regenerate the system catalog. If you previously left SODBC_NONULL unset, set SODBC_NONULL to 0 to maintain the same system catalog column "Null allowed" settings for columns whose repository field "Null allowed" setting is Default. (9.3)
- We updated **dbcreate** so that it generates a caution ("Table *table_name* is already defined") when it encounters duplicate structures. Previously, **dbcreate** generated this as an informational message instead, so duplicate structures were reported only when **-v** was specified. (9.1.5)
- We improved **dbcreate** logging to include the name of structures that have no link to a file. In previous versions, **dbcreate** logged an error in this situation ("ddc_rel:RELATION#:No file link found for related structure"), but didn't list the name of the structure. (9.1)

Installation

For other new installation features, see "Installation" on page 9-2.

- (Windows) The *xf*ODBC Client can now be installed on 64-bit Windows and can coexist with the 64-bit Synergy/DE Client. (9.3)

DSN configuration window

- (Windows) We updated the xfODBC Setup dialog box so that you can now enter a value of 0 in the "Max number of rows" field to turn off multimerge optimization. (9.1)

SQL enhancements

- We updated *xf*ODBC so that you can now use CAST and CONVERT as arguments to other ODBC functions. In previous versions, attempting this caused a "Function 30 not implemented yet" error. (9.5)
- We added support for subqueries in INSERT statements and in SET clauses for UPDATE statements. For example, for INSERT:

```
INSERT INTO orders (or_number, or_item, or_price)
  SELECT 12, in_itemid, in_price FROM plants
    WHERE in_name = 'Wedelia'
```

  For UPDATE:

```
UPDATE orders SET
  or_vendor =
    (SELECT DISTINCT vend_key FROM vendors
     WHERE vend_name = 'Border Imports'),
  (or_item, or_price) =
    (SELECT DISTINCT in_itemid, in_price
     FROM plants
     WHERE in_name = 'Wedelia')
WHERE or_number = 3
```

  (9.5)

- Views are read-only, so we updated *xf*ODBC to fail with an error if you attempt to use UPDATE, INSERT, or DELETE with a view. (9.5)
- If a SQL statement has a FROM clause and a WHERE clause, and the FROM clause has an inline view with a join, *xf*ODBC now correctly applies the criteria for the outer WHERE clause to the legs of the inline view, improving performance. (9.3.1a)
- We updated *xf*ODBC to support the following SQL functions:

ABS	Returns the absolute value of a numeric expression
BITAND	Returns the result of a bitwise AND operation
BITOR	Returns the result of a bitwise OR operation
BITXOR	Returns the result of a bitwise exclusive OR operation
REPLACE	Searches and replaces strings within a string
REVERSE	Reverses the order of characters returned for a string expression.

  (9.3)

- We updated *xf*ODBC's support of the TO_CHAR and TO_DATE scalar functions to include a mask for a two-digit meridian indicator (AM, PM, am, or pm) for date/time fields. (9.3)
- We added support for the CAST scalar function, which enables you to convert an expression or a null value to a specified data type. (9.1.5)

- We added support for the TOP and SKIP subclauses, which enable you to specify the number of rows to be returned for a query (TOP) and the number of rows to be trimmed from the beginning of a result set (SKIP). Together, these can be used for paging, which is particularly useful for creating cached result sets for websites that use ADO.NET. (9.1.5)
- We added support for CASE clauses, which evaluate lists of conditions and return results for conditions that are true. (9.1.5)
- We now support SOME and ANY in WHERE clauses. (9.1.3)
- (SCO OpenServer) We improved performance for some ORDER BY operations. (9.1.3)
- We added support for the GREATEST and LEAST scalar functions, which return the greatest or least value from a group of expressions. (9.1.1b)

*xf*ODBC driver

- (Windows) We updated the *xf*ODBC driver so that it now returns "not supported" for SQL_ASYNC_MODE, which is an ODBC API version 3.0 information type for the SQLGetInfo function. Although the *xf*ODBC driver supports the ODBC version 2.5 API (level 1), rather than 3.0, some applications request SQL_ASYNC_MODE anyway, so we added support for this information type to prevent unnecessary errors. (9.5.3)
- We updated Synergy driver logging (i.e., the logging you get when you set logfile and loglevel in a connect file) so that it no longer displays passwords. (9.3)
- (Windows, UNIX) We updated the *xf*ODBC driver so that it optimizes "SELECT COUNT(*) FROM *table_name*" queries. The *xf*ODBC driver now uses an internal count of records (instead of reading the entire table) if the following are true: the table is stored in an ISAM file, SODBC_MCBA is not set, there is no tag for the table, and there are no filter conditions or joins. (9.3)
- We enhanced the optimizer to take more key choices into account for statements with combinations of restrictions and join criteria. (9.1.3)
- (Windows) We increased the maximum number of rows that can be returned when a statement is optimized with multimerge to 999,999 rows. You can now pass up to 999999 in a SET OPTION MERGESIZE command or in the "Max number of rows" field in the xfODBC Setup dialog box (the dialog box used to configure DSNs for *xf*ODBC). (9.1)
- (UNIX, Windows) The *xf*ODBC driver now supports access to integer data on align-sensitive platforms. (9.1)

Synergy/DE Data Provider for .NET

We introduced the Synergy/DE Data Provider for .NET in version 9.1.5b. See "Accessing Synergy Data in a .NET Environment" in the "Accessing a Synergy Database" chapter of the *xfODBC User's Guide* for information on this product.

- We added support for Visual Studio 2010. (9.3.1b)
- We added the Abs, BitWiseAnd, BitWiseOr, BitWiseXor, Reverse, and Replace canonical functions. (9.3)

Version 8

This section briefly describes new features in *xf*ODBC version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*ODBC version 8.

- **Dbcreate** and the *xf*ODBC driver now distinguish between signed and unsigned fields. This prevents negative values from being stored in non-negative fields, which enables *xf*ODBC to better work with ADO.NET and Microsoft Access. Note, however, that this modification may cause errors when you access your data if unsigned fields contain negative values. (8.3)
- (OpenVMS) We changed the name of **xfodbcusr.exe** to **xfodbcusr_so.exe**, and we changed the XFODBCUSR logical to XFODBCUSR_SO. If you have created your own **xfodbcusr.exe** file for *xf*ODBC, you must rename it to **xfodbcusr_so.exe** and change your XFODBCUSR logical name to XFODBCUSR_SO. (8.1.7)
- (UNIX, OpenVMS) We replaced both the CONNECTEXE logical (OpenVMS) and the CONNECT environment variable (UNIX) with the CONNECTDIR environment variable. Now all platforms—Windows, UNIX, and OpenVMS—use the CONNECTDIR environment variable. (8.1)
- Because Connectivity Series now encrypts user and password information passed in connect strings, we now distribute a **net.ini** file (with an encryption setting) in the connect\synodbc\lib directory. If you've created your own **net.ini** file, add your settings to the new file. Your file will be overwritten if it's in the connect\synodbc\lib directory. (8.1)

New features

Dbcreate and the *xf*ODBC Database Administrator (DBA)

- The Table List window in the Database Administrator Program (DBA) now displays the system catalog creation date for catalogs on the local system. (8.3)
- We updated **dbcreate** so that it can generate a date from the following timestamp format: YYYYMMDDHHMISSUUUUUU (where UUUUUU is the microsecond). And we updated TO_DATE and TO_CHAR to recognize the UUUUUU mask. (8.3)
- (Windows x64 editions, HP Tru64, HP-UX, Sun Solaris, OpenVMS I64) The **dbcreate** utility now generates an error if an integer used in a repository structure is not aligned on a native boundary. The *xf*ODBC driver does not support unaligned integers on these platforms. (8.3)
- (UNIX, OpenVMS) The **dbcreate** utility now generates a caution message if it encounters a tag or condition that causes optimization to be reduced for a table. (This was fixed for Windows in 8.1.7d.) (8.3)

- We improved **dbcreate** and the *xf*ODBC driver so that they now distinguish between signed and unsigned fields. When **dbcreate** generates a system catalog, it checks the "Negative allowed?" repository setting to determine if the resulting column will be signed or unsigned.
  - If the "Negative allowed?" setting is No, the resulting column will be unsigned.
  - If the "Negative allowed?" setting is Yes, Only, or OrZero, the resulting column will be signed unless a range that includes only positive values is assigned to the field, in which case the column will be unsigned.

  This enhancement prevents negative values from being stored in non-negative fields, which enables *xf*ODBC to better work with ADO.NET and Microsoft Access. Note that this may cause errors when you access your data if unsigned fields contain negative values. To prevent this, do one of the following:
  - Use **fcompare -dv** after regenerating your system catalog to make sure there are no unsigned columns that have negative values. If there are, either update the data, or change the validation information ("Negative allowed?" and range fields) for the field and regenerate the system catalog.
  - Revert to the previous *xf*ODBC behavior by setting the SODBC_NOUNSIGNED environment variable to any value before generating the system catalog. This instructs **dbcreate** to ignore the "Negative Allowed?" field (which is set to No by default) and to set all fields to signed unless they have validation ranges that are limited to positive values (in which case the resulting columns will be unsigned).

  (8.3)
- (Windows) The **dbcreate** utility now generates a caution message if it encounters a tag or condition that causes optimization to be reduced for a table. (This was fixed for UNIX and openVMS in 8.3.) (8.1.7d)
- (UNIX) The **dbcreate** utility now allows 8-byte integer fields on all supported UNIX platforms except SCO OpenServer. (8.1.7a)
- (Windows) The Log file field in the Compare System Catalogs to Files window of the DBA program now supports paths that contain spaces. (8.1)
- (Windows) The text fields in the Generate System Catalog window of the DBA program are now scrolling input fields rather than multi-line text fields. (8.1)

Connect files

- (Windows) You can now include quoted paths in the dictsource and datasource lines of a connect file. (8.1)

Installation

For new installation features, see "Installation" on page 9-6.

DSN configuration window

- (Windows) The xfODBC Setup dialog box, which enables you to create and configure *xf*ODBC DSNs, has been updated to include a "Max number of rows" field, which corresponds to the SET OPTION MERGESIZE *max_rows* option. This option enables you to control the way *xf*ODBC optimizes SELECT statements that have one or more OR clauses. Additionally, we increased the default value for this field to 10,000 and the maximum value of this field to 999,999. We also removed the vestigial "Host OS" field. (8.3)

ODBC API enhancements

- (Windows) To support ADO.NET, we made a number of improvements to the *xf*ODBC driver's support of the ODBC API:
  - SQLColAttributes now correctly describes read-only, nullable, and unsigned types.
  - SQLColAttributes now correctly returns data when passed with SQL_COLUMN_OWNER and SQL_COLUMN_TABLE_NAME (unless the column has aggregate or subquery values).
  - SQLColAttributes now returns the name of the connect file when passed with SQL_CURRENT_CATALOG.
  - SQLColAttributes now supports SQL_DESC_TYPE, SQL_DESC_LENGTH, SQL_DESC_PRECISION, SQL_DESC_SCALE, and SQL_DESC_OCTET_LENGTH.
  - When passed with SQL_DATABASE_NAME, SQLGetInfo, now returns the name of the current database.
  - When passed with SQL_SERVER_NAME, SQLGetInfo returns the host name. (When using SQL OpenNet, the server name is taken from the *@server_name* section of the connect string. Otherwise, the local IP address from GetHostName or localhost is used.)
  - SQLGetInfo now returns true for SQL_LIKE_ESCAPE_CLAUSE and a single backslash (\) for SQL_SEARCH_PATTERN_ESCAPE, and you can now use the backslash character as an escape character for percentage (%) and backslash characters in LIKE clauses. Additionally, you can now use the following escape syntax (braces are optional):

    ```
    LIKE search_string {ESCAPE 'c'}
    ```

    where *c* is the single character in *search_string* that the escape will apply to.
  - ODBC API functions that search system catalogs now recognize the backslash character (\) as an escape character for the percent sign (%), the underscore character (_), and the backslash character in search strings.

  (8.3)
- *xf*ODBC now supports the SQL_LOGIN_TIMEOUT option for SQLSetConnectOption and SQLGetConnectOption. With SQLSetConnectOption, this option specifies the amount of time (in seconds) to wait for a login request to complete. With SQLGetConnectOption, SQL_LOGIN_TIMEOUT returns the login time-out setting in seconds. (8.1.7)

- To better support Cognos Impromptu, we've updated the *xf*ODBC driver to support read-only transaction mode. Both connection handles and statement handles can be now created as read-only (though by default, they're created with read/write access). To support this, *xf*ODBC now recognizes the following:

  SQLSetConnectOption(SQL_ACCESS_MODE, SQL_MODE_READ_ONLY)

  Additionally, the following functions now return SQL_MODE_READ_ONLY when the read-only attribute is set for the connection handle or statement handle.

  - SQLGetConnectOption(SQL_ACCESS_MODE) now returns SQL_MODE_READ_ONLY when the read-only attribute is set for a connection handle.
  - SQLGetInfo(SQL_SCROLL_CONCURRENCY) now returns SQL_SCCO_LOCK and SQL_SCCO_READ_ONLY, the two values supported by SQL_CONCURRENCY on SQLSetStmtOption and SQLGetStmtOption.
  - GetInfo(SQL_DATA_SOURCE_READ_ONLY) always returns N, even if the connection is set to SQL_MODE_READ_ONLY.

  (8.1.7)

- (Windows) *xf*ODBC now supports quoted table names and quoted schema names (table owner names) as arguments for ODBC API catalog functions (SQLStatistics, for example). (8.1.5d)

Optimization enhancements

- We improved *xf*ODBC so that it now optimizes SQL statements with restrictions that include an AND clause and an IN clause. For example, for the following statement, *xf*ODBC will now use both **id** and **num** (if they are key segments) to optimize the query. In previous versions, *xf*ODBC would have used **id**, but not **num**.

  ```
  SELECT myfld FROM mytbl WHERE id = 1
      AND num IN (48, 49, 50)
  ```

  Note that this optimization also applies to equivalent SQL statements that use a set of OR clauses instead of IN. For example:

  ```
  SELECT myfld FROM mytbl WHERE id = 1
      AND (num = 48 or num = 49 or num=50)
  ```

  (8.3)

- We improved *xf*ODBC so that it now creates a temporary index for an inner join in most cases (regardless of other optimizations) if no join segment is usable as a key and no temporary index has already been created for the table. (8.3)
- (Windows) We improved the *xf*ODBC driver so that ODBC-enabled applications can now change the row set size after a cursor has been created. This improves performance when using SQL Server linked servers. (8.3)

- We added a new flag, "Excluded by ODBC," to S/DE Repository. This flag enables you to omit keys from a system catalog. If you set this flag for a key definition in your repository, **dbcreate** will ignore that definition, so the resulting system catalog won't have an index that corresponds to the key. (8.1.7)
- By default, *xf*ODBC now optimizes SELECT statements that have one or more OR clauses if keys are available to optimize each side of each OR clause. *xf*ODBC also includes a new option, SET OPTION MERGESIZE, that either turns this feature off (when set to 0) or sets the maximum number of rows (which can be from 100 to 65535) allowed in result sets for statements optimized with this new feature. By default, SET OPTION MERGESIZE is set to 1000. (8.1.7)
- We enhanced the *xf*ODBC optimizer to reorder the tables in joins if better key selection will result or if no key would otherwise be used (i.e., in joins that would result in a table scan for the joined table). (8.1.7)
- We enhanced the *xf*ODBC optimizer to make better key selections for BETWEEN clauses. (8.1.7)
- We improved the temporary index *xf*ODBC creates if any field in a join is not part of an index segment. Previously, the temporary index would consist of only the first column specified for the join. Now, however, when *xf*ODBC creates a temporary index, it includes all of the join columns. You can turn this feature off by setting SET OPTION TMPINDEX to OFF. (See SET OPTION in Appendix B of the *xfODBC User's Guide*.) If TMPINDEX is set to OFF, *xf*ODBC will use any column that is a key segment and that is part of the join criteria to optimize the join—unless the column is a user-defined field or, on OpenVMS, an integer field. (8.1.5)
- (Windows) *xf*ODBC can now use up to 99 keys to optimize a SQL statement. Previously, it could use only 16. (8.1.5)
- We added a new logical, TRIM_HOME, and two new files, **trim.ini** and **trim.msg**, for system catalog caching. TRIM_HOME is set automatically and is used to locate the **trim.ini** file, which specifies the amount of space for each shared memory segment used by the cache. The **trim.msg** file enables **syngenload** to print error text rather than just error numbers. (8.1.3)
- We added support for caching system catalogs on Windows NT/2000/XP, UNIX, and OpenVMS systems. For information, see "System Catalog Caching" in the "Configuring Data Access" chapter of the *xfODBC User's Guide*. (8.1)
- We improved *xf*ODBC's data access optimization to use a tag in any segment of a key. In previous versions, a tag had to be the first segment of a key to be used for optimization. (8.1)

Routines for user-defined data types

- (OpenVMS) We changed the name of **xfodbcusr.exe** to **xfodbcusr_so.exe**, and we changed the XFODBCUSR logical to XFODBCUSR_SO. If you have created your own **xfodbcusr.exe** file for *xf*ODBC, you must rename it to **xfodbcusr_so.exe** and change your XFODBCUSR logical name to XFODBCUSR_SO. If you have not created your own **xfodbcusr.exe**, you don't need to make any changes. (8.1.7)

SQL enhancements

- *xf*ODBC now allows you to use a table name or an owner and table name to qualify an asterisk in a column list. For example both of the following are supported:

  `SELECT` *table_name*`.*` `FROM` *table_name*

  `SELECT` *owner_name*`.`*table_name*`.*` `FROM` *table_name*

  (8.1.7)

- We improved SET OPTION PLAN logging so that it now includes column names in key descriptions. (8.1.7)
- We improved SET OPTION PLAN so that it now logs information on subqueries when it's set to ON. (8.1.5)
- We added a new option, PLAN, to the SET OPTION command. PLAN lists the indexes used for a query. For more information, see SET OPTION in Appendix B of the *xfODBC User's Guide*. (8.1)

SQL OpenNet

For new SQL OpenNet features, see "SQL OpenNet" on page 9-6.

*xf*ODBC driver

- (Windows) We updated the reserved keyword list for the *xf*ODBC driver so that cooperating applications can quote fields that have names that are reserved keywords. (8.3)
- (Windows) The *xf*ODBC driver now passes and describes strings as SQL_VARCHAR by default. In previous versions, the *xf*ODBC driver described strings as SQL_CHAR by default, even though it passed them as SQL_VARCHAR. (You can use the VORTEX_ODBC_CHAR environment variable to revert to the previous behavior.) (8.1.7e)
- *xf*ODBC now supports ODBC connections through ADO.NET. To use ADO.NET, you'll need the following products in the following or higher versions:

  Synergy/DE 8.1.7
  .NET Framework 1.1
  MDAC 2.8
  Visual Studio .NET 2003

  You'll also need the patch for Microsoft Support issue 833050 (for Windows 2000/XP) or 833742 (for Windows 2003). To get one of these patches, call Microsoft Support and provide them with the issue ID (833050 or 833742) for your problem. (8.1.7)

- (Windows) The *xf*ODBC driver now supports multi-threading for connections made without SQL OpenNet. This includes Microsoft Data Transformation Services (MSDTS) connections and IIS pooled connections. (8.1.3)
- (Windows) **Vtx4.dll**, the Synergy database driver, is now thread safe. When using the ODBC API to call SQLDriverConnect to open a connection from a heavily threaded environment, be sure to use SQL_DRIVER_NOPROMPT. This prevents a login box from popping up, which causes **vtx4.dll** to hang. (8.1)

Miscellaneous

- *xf*ODBC now supports the Developer and Professional editions of Crystal Reports XI, along with the Developer and Professional editions of Crystal Reports 8, 9, and 10. (Other versions may return invalid data or cause reports to hang.) (8.3)
- (Windows) We updated *xf*ODBC to support Microsoft Visual Studio 2005 (including the ADO visual designers for this product) and SQL Server 2005. Note that you must use Connectivity Series 8.3 to use these products, and note that we do not recommend installing *xf*ODBC to run in a shared environment when you are running Visual Studio 2005 on the client machine for *xf*ODBC development. However, if you do have this configuration, you must change your .NET security permissions on the client machine to permit access to the assembly **xfODBCSpecialization.dll**, located in the SynergyDE\connect directory on the shared machine. This enables *xf*ODBC to work with the wizards in Visual Studio 2005. (8.3)
- (Windows) The **dltest.exe** program is now included in the *xf*ODBC Client installation. (8.3)
- We added an example program, **exam_saveviews.dbl**, to the Connectivity Series distribution (in the synsqlx directory). This SQL Connection program retrieves view information from a system catalog and uses that information to create SQL statements that can be used to re-create the views after the system catalog is regenerated. (8.3)
- (OpenVMS) We changed the VTXIPC logical to VTXIPC_SO. (8.1.7)
- *xf*ODBC is now certified for Office 2003 and XP with Jet Service Pack 6 or higher. (8.1.3)

Version 7

This section briefly describes new features in *xf*ODBC version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break your code when you update to *xf*ODBC version 7.

- We changed the way SODBC_NONULL is set. Formerly, if you set SODBC_NONULL to any value, it affected the way all decimal fields with zeros or nulls were interpreted. Now, if you set it to 2, it affects only primary key segments. If you set it to 3, it affects all key segments. If you set it to any other value, it affects all fields (the original behavior). (7.5)
- *xf*ODBC does *not* use the following environment variables, which were used by *xf*ODBC Driver or *xf*ODBC Connect.

  DAI_AUDIT_FILE
  DAI_AUDIT_LEVEL
  SODBC_NOROLL
  SODBC_TOKEN
  XFODBC_NEWVAR
  XFODBC_TOKEN

  (7.3)
- With *xf*ODBC Driver, there were two ways to enter a network string (@[*port_no*:]*host_name*!*dbdriver*): You could enter it in the "Appended to connect string" field of the xfODBC Setup window as you created a DSN, or users would enter it as they logged in. For *xf*ODBC, however, you specify this information with the following fields of the xfODBC Setup window (the window used to configure DSNs): Vortex driver, Host OS, and Host.

  In addition, note that the *dbdriver* name has changed for UNIX. It is now **VTX4**, but you don't need to specify it. It's automatically specified when you choose Net in the Vortex driver field and Unix in the Host OS field. (7.3)

- (UNIX) You must set the LD_LIBRARY_PATH environment variable (LIBPATH on AIX, SHLIB_PATH on HP-UX) before using SQL Connection or *xf*ODBC. This environment variable specifies the location of dynamically-loaded shared object files that are used by both *xf*ODBC and SQL Connection. To set this environment variable, run the **setsde** script, which sets up your Synergy environment, or use the following syntax to set it manually:

  ```
  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:new_path;export LD_LIBRARY_PATH
  ```

  where *new_path* is the path that contains the **gds0.so** file. For example:

  ```
  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr2/synergyde/connect;export
  LD_LIBRARY_PATH
  ```

  Note that you must set this before starting **vtxnetd** as an OpenNet server and before using SQL Connection to access Synergy ISAM files. Also note that the LD_LIBRARY_PATH environment variable (LIBPATH on AIX, SHLIB_PATH on HP-UX) operates the same way as the UNIX PATH environment variable: when it's specified, the operating system looks for the first match in the search path. In addition, don't add the setuid bit (+s) to **vtxnetd** or VTX4. If you do, these environment variables won't be used and you'll get DLLLOAD errors. (7.3)

- Make sure the SODBC_NOROLL environment variable is *not* set. This environment variable is obsolete, but it can affect the way the system catalog is generated. If it's set when the system catalog is generated, all dates with two-digit years will be interpreted as dates in the current century. (7.3)
- We changed the way multi-dimensional arrays are named in *xf*ODBC. (7.3)
- The characters "sdms:" can no longer be used in a connect string for *xf*ODBC. (7.3)
- The default formats for date/time, dates, and times have changed. The format for date/time fields is *YYYY-MM-DD HH*:*MI*:*SS*. The format for date fields is *YYYY-MM-DD*. The format for time fields is *HH*:*MI*:*SS* (where *HH* is 0-23). (7.3)
- The format for DSNs has changed. All *xf*ODBC Driver and *xf*ODBC Connect DSNs must be replaced with *xf*ODBC DSNs. (7.3)

New features

SQL OpenNet

For new SQL OpenNet features, see "SQL OpenNet" on page 9-15.

Sample database and tutorial

- We created a new sample database that correctly works with **fcompare** and includes more examples of features such as foreign keys. This database also includes relations to tagged files. (7.3.3)
- The sample database is now more than just a sample—it's part of a tutorial that guides you through the tasks involved in developing and maintaining a system catalog. (7.3)

SQL enhancements

- (Windows) The system catalog functions (SQLTables, SQLColumns, SQLPrimaryKeys, SQLForeignKeys, etc.) now return column names that meet ODBC specification requirements. (In previous versions, *xf*ODBC returned column names that didn't adhere to the ODBC specifications.) (7.5.1e)
- (Windows) We added the TRUNC scalar function which removes either the fractional portion of a number or the time portion of a date/time value. (7.5.1c)
- (Windows) SQLGetStmtOption(SQL_MAX_LENGTH) now returns 0 (it previously returned 255), and the driver attempts to return all char and varchar data. You can use this function to set the maximum length for char and varchar columns. The length can be up to 65,535. (Exceeding 65,535 causes a SQL_SUCCESS_WITH_INFO error "Option value changed" and adjusts the value to 65,535.) Setting this value does not cause data truncation. (7.5.1c)
- (Windows) The SQL_CONNCURRENCY statement now defaults to SQL_CONCUR_READ_ONLY as specified by the ODBC API specification. (It formerly defaulted to SQL_CONCUR_LOCK.) (7.5.1c)
- (Windows) In previous versions, SQLRowCount returned 4,096 for all SELECT statements. We corrected this to return -1: "the actual row count for SELECT statements is unknown." (7.5.1c)
- (Windows) SQLGetInfo(SQL_KEYWORDS) now returns a comma-separated list of keywords unique to the *xf*ODBC driver. (Formerly, it returned SQL_SUCCESS and an empty result string.) (7.5.1c)
- *xf*ODBC now supports derived tables when accessing a Synergy database. Derived tables are created by SELECT commands that are used in place of table-references in FROM clauses. This feature improves performance for SQL92 outer joins by reducing the size of the joined tables. (7.5)
- (Windows) We added support for the SQLSetStmtAttribute ("SQL_ATTR_RETRIEVE_DATA") option. This improves our ADO support. (7.5)

- ODBC Level 2.5 API functions are now fully supported. This includes support for SQLDescribeParam, SQLMoreResults, SQLNativeSQL, SQLParamOptions, SQLSetScrollOptions, SQLForeignKeys, and SQLPrimaryKeys. (7.3)
- The SQL_BIGINT data type is now supported. (7.3)
- *xf*ODBC now supports SQL92 INNER JOIN, LEFT [OUTER] JOIN, RIGHT [OUTER] JOIN, and FULL [OUTER] JOIN. In addition, joins do not need to be enclosed in the ODBC escape clause. For example, both of the following are acceptable. (7.3)

  ```
  {oj orders RIGHT JOIN vendors ON or_vendor = vend_key}
  ```

  ```
  orders RIGHT OUTER JOIN vendors ON or_vendor = vend_key
  ```

- *xf*ODBC now supports UPDATE, DELETE, and INSERT for tables with unqualified names. (7.3)
- *xf*ODBC supports the ORDER BY clause—even if the field specified in the clause is not in the select list. This enables Microsoft Access to create queries for the selected fields. (7.3)
- The UNION operator is now supported. UNION enables you to create a single result set from multiple SELECT statements. (7.3)
- The CREATE VIEW statement is now supported. CREATE VIEW enables you to create a view from one or more tables or existing views. (7.3)
- The CREATE SYNONYM statement is now supported. CREATE SYNONYM enables you to create an alias for a table or view. (7.3)
- The FOR UPDATE OF clause is now supported. FOR UPDATE OF enables you to lock rows that match a statement's selection criteria. (7.3)
- *xf*ODBC supports many additional scalar functions. (7.3)
- Column and table aliases are now supported. (7.3)
- *xf*ODBC conforms to the SQL92 standard by uppercasing table identifiers (table name and column name) by default. (7.3)
- SQLPrimaryKeys() returns the first unique key defined for a table. If no unique key is found, no rows are returned. (7.3)

Log-in window and DSN configuration window

- We added a new field, Env. variables, to the xfODBC Setup window. This field enables you to set environment variable definitions for environment variables used by the *xf*ODBC driver on the client. This is the only place these environment variables can be set for services such as web servers that use the *xf*ODBC driver. (7.5)
- The "Connect file" text box, a field in the xfODBC Setup dialog box, has been expanded. (7.3.3)

- We made several improvements to the log-in and DSN configuration windows. The log-in window (xfODBC Info) is the window end users see when they select an *xf*ODBC DSN from their ODBC-enabled applications. The DSN configuration window (xfODBC Setup) is available from the ODBC Data Source Administrator.
  - You can now store user name and password information in the DSN configuration window.
  - You no longer need to specify a remote driver name in the login window or the DSN configuration window.
  - Users are no longer required to enter "sdms:" in the log-in window or the DSN configuration window.
  - The DSN configuration window has separate fields for port number (Port), remote machine name (Host), and operating system for the remote machine (Host OS).
  - We added the Total and In Memory options to the DSN configuration window. These options enable you to control the size of work files created for queries with ORDER BY clauses.

  (7.3)

Routines for user fields

- *xf*ODBC enables you to create routines that manipulate data stored in user fields. You can create routines that manipulate data as it's read from or written to a database. (7.3)

Installation

- (Windows) The Connectivity Series installation now checks for and, if necessary, installs MDAC version 2.7. (7.5.1c)
- (UNIX) The **startnet** file is no longer created by the installation. It is, however, included in the distribution. The **setodbc** script file has been modified to run **startnet**. (7.3)

Support for Synergy data types

- We added an environment variable, VORTEX_ODBC_TIME, to enable you to more easily handle time columns retrieved by applications that use ADO.NET. Setting VORTEX_ODBC_TIME=11 instructs the *xf*ODBC driver to describe time columns as SQL_TIMESTAMP, a data type that ADO.NET can use. (8.3)
- We added a new TIMESTAMP data type (specified with ^CLASS^="YYYYMMDDHHMISS"), which uses a 14-character field of the same format. (7.3.3)

- *xf*ODBC supports the following:
  - Synergy time fields.
  - Additional Synergy date fields. This includes support for a new environment variable, SYNBASEDATE.
  - Synergy **i8** integer fields.
  - Synergy user-defined fields.

  (7.3)
- *xf*ODBC now translates Synergy date fields to date values, rather than datetime values. (7.3)

Support for groups

- By default, if a field is part of a group in the repository, the column name created in the system catalog for the field will include the group name as a prefix. We added an environment variable, SODBC_NOGROUPNAME, That enables you to omit these group names from the column names in the generated system catalog. (7.5)
- *xf*ODBC supports S/DE Repository groups (both explicit and implicit). (7.3)

Support for tags

- *xf*ODBC supports 10 tags per file, which is the maximum number supported by Repository. (7.3)

*xf*ODBC Database Administrator (DBA)

- When you compare the system catalog to the repository files by selecting Compare > Compare to Files, the DBA utility now uses the **-i** option (rather than **-v**) as it spawns the **fcompare** utility. (7.5)
- We improved the user interface for the DBA program. For example, the DBA program now has online help, and you can now use the Group ID drilldown button in the User window to select a group ID for a user. (7.3)
- The DBA Generate function now handles comment lines in the connect file and trailing slashes at the end of an RPSDAT specification. (7.3)

Catalog-generation utility (dbcreate)

We made the following changes to **dbcreate**, the catalog-generation utility. (This utility replaces **sodbccnv**, the *xf*ODBC Driver catalog-generation utility.)

- If SODBC_NOGROUPNAME is set to any value and your repository has a group that is an array and contains an arrayed field, **dbcreate** now adds GR#*n* (where *n* is an element number for the group array) to the beginning of the field name to create the column names. (Formerly, **dbcreate** added #*n*.) *xf*ODBC and the SQL92 standard require the first character of a column to be an alphabetical character. (7.5.1a)
- There is a new set of command line options for the **dbcreate** utility. (7.3)
- **dbcreate** can now use values in the ODBC table name field of Repository. (7.3)

Error messages and logging

- (OpenVMS) If an "out of memory" error or an error 17 (bad file specification) occurs when you access a database, the SDMS log file now reports the additional system error code. (7.3.3)
- We updated **dbcreate** reporting and logging options. (7.3)
- We added a new environment variable, GENESIS_INITSQL, that enables you to set SQL options at start-up. (7.3)
- *xf*ODBC has a new utility that enables you to generate the error message file (**sql.msg**) from a text file you can edit (**sql.txt**). In addition, there's a new environment variable, GENESIS_MSG_FILE, that enables you to specify the location of the error message file. (7.3)

Optimization enhancements

*xf*ODBC includes the following optimization enhancements.

- Temporary indices are now available for pre-revision 4 ISAM files. (7.3.3)
- We improved optimization for WHERE clauses that specify columns that are in more than one segmented key. For example, if a WHERE clause specifies field1 and field3 and there are two segmented keys, key0 (with field1 and field3) and key1 (with field1, field2, and field3), key0 will now be used. (7.3.3)
- *xf*ODBC can use overlaid keys to improve performance when overlaid fields are referenced. (7.3)
- We improved optimization for relative file joins. For example, the *xf*ODBC driver is now able to position on a record number, enabling *xf*ODBC to directly access a row, rather than evaluating the record number of every row in the table. (7.3)
- We improved IN operator performance. For IN operators, *xf*ODBC now uses keys when they're available. (7.3)
- If an index is defined as unique (no duplicates), the *xf*ODBC driver now does a single read. It does *not* perform a subsequent read for the following key. (7.3)

- The scrolling and page down features in Microsoft Access are now smoother in linked tables. (7.3)
- We optimized performance for the LIKE keyword, including improved string comparisons. (7.3)
- The driver now supports query time-outs for applications that support this feature. (7.3)
- (Windows) If you use the close button to quit an application, the server process is terminated. (7.3)
- *xf*ODBC now optimizes ORDER BY clauses with both ascending and descending order. (7.3)
- *xf*ODBC now uses tags to optimize queries—even if the tag is part of the index. (7.3)
- If *xf*ODBC isn't able to use a key when processing a table join, *xf*ODBC now creates a temporary index for ISAM files to optimize data access. Note that this feature is available only for revision 4 files. Use **ipar** or **fconvert** to view the revision of your files. (7.3)
- If a key segment is an overlay field, *xf*ODBC now creates an alternate index that consists of all the fields that make up the overlay. (7.3)

Y2K support

- Y2K support has been enhanced for **d6** data types. The SYNCENTURY environment variable defines a "sliding window" for the default century. (7.3)

Miscellaneous

- (OpenVMS) **Startnet.com** has been improved to allow 10 concurrent ODBC connections. This is dependent on the number of files open and your particular operating system parameters. Hence, we have documented the parameters to change (within **startnet.com**) if you still encounter concurrent connection limits. (7.5.1d)
- We added a new environment variable, VORTEX_ODBC_CHAR, that enables you to describe SQL_VARCHAR strings as SQL_VARCHAR strings, a requirement for Microsoft Data Transformation Services (DTS). If VORTEX_ODBC_CHAR is not set, which is the default, or if it's set to 1, the *xf*ODBC driver passes SQL_VARCHAR strings but describes them as SQL_CHAR strings. (7.5)
- We added more ways to set SODBC_NONULL. Formerly, if you set SODBC_NONULL to any value, it affected the way all decimal fields with zeros or nulls were interpreted. Now, by setting it to specific values, it affects one of the following: only primary key segments, all key segments, or all fields (the original behavior). (7.5)
- Multi-threaded applications such as Cognos Impromptu now work with local data sources. (7.5)
- *xf*ODBC is now certified for Office 2000. (7.3.3a)

- We added the SODBC_NONULL environment variable which causes alpha and decimal fields that contain null values to be returned as empty strings or zeros rather than null. If SODBC_NONULL is set to any value, the *xf*ODBC driver will return an empty string for an alpha field that's set to null, and it will return zero for a decimal field that's set to null. Date fields that contain null are returned as null, even if SODBC_NONULL is set. (7.3.3)
- (Windows) The TEMP environment variable now determines the location of the sort work file used in ORDER BY, GROUP BY, and DISTINCT clauses. (7.3.3)
- (OpenVMS) **sys$scratch** is now used on OpenVMS for the sort work file. (7.3.3)
- You can now use DBQ=*connect_filename* to specify a connect file in a file DSN. (7.3.3)

  For example:

  ```
  DBQ=sodbc_sa
  ```

  You can use DBQ= in a connect string in an application (ADO, VB, C++, etc.). For example:

  ```
  DRIVER=xfodbc;UID=DBADMIN;PWD=MANAGER;DBQ=sodbc_sa;
  ```

- (Windows) Previously, *xf*ODBC would allow only eight connections. We increased the limit to 1024 connections. (7.3.3)

9

SQL Connection

Version 9

This section briefly describes new features in SQL Connection version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to SQL Connection version 9.

SQL Connection

- To support multi-threading with the .NET Framework, we updated the error message buffer for SQL Connection so that it is now channel specific. In versions prior to 9.5.1a, %SSC_GETMSG ignored the channel number argument and instead returned the error message for the most-recently executed SQL statement. Note that there is a case where this modification could break your code: if %SSC_GETMSG returned the error message for the channel you expected in previous versions, though you had inadvertently passed a different channel number, %SSC_GETEMSG will now return the error message for the incorrect channel number. (9.5.1a)

New features

Installation

- (Windows 2000) The Connectivity Series installation has been updated to install MDAC 2.8 with Service Pack 1 on systems that do not have at least MDAC 2.7 with Service Pack 1. (9.1)
- (Windows) The Connectivity Series installation now installs an **opennet_base.srv** file and no longer overwrites the **opennet.srv** file. If no **opennet.srv** file exists, the installation creates one by copying **opennet_base.srv**. Otherwise, it leaves **opennet.srv** as is. (In all cases, however, the installation installs **opennet_base.srv**, overwriting an existing copy if one exists.) This enables you to make changes to **opennet.srv** without the danger of losing those changes when you install a new version of Connectivity Series. Additionally, you can refer to the **opennet_base.srv** file whenever you want to see the settings as distributed by the installation. (9.1)

SQL Connection

- (Windows, UNIX) We added support for MySQL (version 5.1 and higher) on AIX, and we dropped support for MySQL 5.0 versions (which are no longer supported by Oracle) on Windows and Linux platforms. The minimum MySQL version for SQL Connection on Windows, Linux, and AIX is now 5.1.

  NOTE: If you upgrade to MySQL 5.1, the installation will not remove the **libmysql.dll** library for your previous installation. And if that library is still in your path, you may get a "Client version mismatch" error. If you get this error, remove the old **libmysql.dll** library, or make sure it's not in your path. (9.5.1b)

- To support multi-threading with the .NET Framework, we updated the error message buffer for SQL Connection so that it is now channel specific. In versions prior to 9.5.1a, %SSC_GETMSG ignored the channel number argument and instead returned the error message for the most-recently executed SQL statement. Note that there is a case where this modification could break your code: if %SSC_GETMSG returned the error message for the channel you expected in previous versions, though you had inadvertently passed a different channel number, %SSC_GETEMSG will now return the error message for the incorrect channel number. (9.5.1a)
- (Windows) We increased the number of channels that can be open concurrently to 100. (9.5)
- To meet HIPAA compliance, we updated VORTEX API logging and VORTEX host logging so that connect strings no longer appear in log files. We also updated SQL Connection logging (SSQLLOG) so that passwords are masked with asterisks. (9.3)
- (Windows) SQL Connection now supports SQL Server 2008 when using the VTX12_SQLNATIVE driver. (9.1.3)
- We added support for the System.String data type. You can now use string variables in non-array-based %SSC_OPEN, %SSC_DEFINE, %SSC_BIND, and %SSC_MOVE operations. (9.1.3)
- We improved SQL Connection's ability to get and put large binary column (BLOB) and large character column (CLOB) data. To do this, we replaced %SSC_BLOB with a new routine, %SSC_LARGECOL, and we replaced the SSQL_BLOBCOL option for %SSC_OPEN with SSQL_LARGECOL. We also added the SSQL_LARGECOL option to %SSC_EXECUTE, and we made the *option* argument for %SSC_EXECUTE optional. (The %SSC_BLOB function and the SSQL_BLOBCOL option for %SSC_OPEN have not been removed, just deprecated.) (9.1.3)
- We added a new option, MULTI, to VORTEX_API_LOGOPTS and VORTEX_HOST_LOGOPTS. This option instructs Vortex API logging or Vortex host logging to create a separate log file for each concurrently open database channel. Additionally, MULTI is automatically set for VORTEX_HOST_LOGOPTS when using **vtxnetd** on Windows, which solves one aspect of a problem that occurs when the FULL option for Vortex host logging is used with multiple concurrently open database channels. (9.1.3)

- We added a new option, SSQL_EXBINARY, for the *type* argument to %SSC_EXECIO. This option instructs %SSC_EXECIO to retain binary zeros in data retrieved from or sent to binary columns. (If you don't use this option, %SSC_EXECIO converts binary zeros to spaces.) (9.1.3)
- We improved SQL Connection logging (the logging you get with the SSQLLOG environment variable) so that it now logs warnings and returns the correct information for SQL Server. Additionally, we fixed %SSC_GETEMSG so that it now returns the correct information for SQL Server. (9.1.3)
- We added support for DISTINCT in subqueries. (9.1.3)
- (Linux) We added support for MySQL (version 5.0.24 and higher). (9.1)
- We combined and updated several of the sample programs for SQL Connection:
  - We replaced **exam_sqlsrvs.dbl**, **exam_oras.dbl**, **exam_oras1.dbl**, and **exam_sqlsrvbulks.dbl** with **exam_create_table.dbl**. This new sample program creates a table and inserts some rows of data. If you remove the comment character (;) at the beginning of the ".define MULTI_ROW_INSERT" line, it uses a bulk insert.
  - We replaced **exam_sqlsrv.dbl** and **exam_ora.dbl** with **exam_fetch.dbl**. This example contains a simple query that retrieves rows inserted by **exam_create_table.dbl**.
  - We replaced **exam_sqlsrv1.dbl** and **exam_ora1.dbl** with **exam_multirow_fetch.dbl**. This program contains a multi-row query that retrieves rows inserted by **exam_create_table.dbl**.
  - We replaced **exam_sqlsrv2.dbl**, **exam_ora2.dbl**, and **exam_ora3.dbl** with **exam_fetch_update.dbl**. This program contains a simple update that retrieves rows inserted by **exam_create_table.dbl** and then updates the data using %SSC_SQLLINK. It also uses %SSC_REBIND.

  (9.1)

Version 8

This section briefly describes new features in SQL Connection version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to SQL Connection version 8.

SQL OpenNet

- (Windows) We changed the default SQL OpenNet server on Windows to **vtxnetd**, a multi-threaded server. This may break your code if you use SQL Server. See "SQL OpenNet" on page 9-6. (8.3)

SQL Connection

- (Windows) We changed the default database driver for SQL Server to VTX12_ODBC. See "SQL Connection" on page 9-9. (8.3)
- (OpenVMS) We moved to a shared image model (similar to the **.so** and **.dll** files on Windows and UNIX) for the database drivers on OpenVMS. If you are using Oracle or Oracle RDB natively, you must rebuild your database driver using **build_ssql_db.com.** If you don't, your application will fail. (8.1.7)
- (UNIX, OpenVMS) We replaced both the CONNECTEXE logical (OpenVMS) and the CONNECT environment variable (UNIX) with the CONNECTDIR environment variable. Now all platforms—Windows, UNIX, and OpenVMS—use the CONNECTDIR environment variable. (8.1)
- Because Connectivity Series now encrypts user and password information passed in connect strings, we now distribute a **net.ini** file (with an encryption setting) in the connect\synodbc\lib directory. If you've created your own **net.ini** file, add your settings to the new file. Note that your file will be overwritten if it's in the connect\synodbc\lib directory. (8.1)

New features

Installation

- (Windows) The Synergy/DE installation (Connectivity Series component) and Synergy/DE Client installation now install the current MDAC version only if the target system does not have MDAC 2.71 Service Pack 1 or higher already installed and the system does not have SQL Server Cluster installed. The currently distributed MDAC version is 2.8. On Windows NT, 2000, and XP (with no service packs), the user will be prompted to reboot. If the target system already has the minimum required MDAC version for Synergy/DE, but you would like to upgrade to version 2.8, the **mdac_typ.exe** file is available on the Annual License Maintenance Downloads page and the Synergy/DE software CD. (8.1.7)
- (OpenVMS) We added the following system-wide logicals, which are used internally by Connectivity Series: VTX0_SO, VTX1_SO, VTX2_SO, VTX4_SO, SSQLRTL, GDS0_SO. (8.1.7)
- For local or shared installations of Synergy/DE, the installation/upgrade of MDAC is now associated with the Connectivity Series component, rather than with Core Components. (8.1.5)
- (Windows) The Synergy/DE installation and Synergy/DE Client installation now check for and install MDAC version 2.71 Service Pack 1 if necessary. The exceptions to this are on systems where SQL Server Cluster is installed, and on XP and 2003, where MDAC is part of the operating system and can only be updated via an operating system service pack. (8.1.5)
- (UNIX, OpenVMS) We added the CONNECTDIR environment variable, which replaces CONNECTEXE on OpenVMS and CONNECT on UNIX. (8.1)

SQL OpenNet

- (Windows) We reduced the memory used by **vtxnetd** to 512K to enable it to support more concurrent users. Previously, the Windows thread stack limit of 1 MB prevented **vtxnetd** from supporting more than 1000 users.

  To enable you to reduce the memory used by **vtxnetd** even further, we added a new option to **vtxnetd**: -**s***n*, where *n* is a number in kilobytes. If a particular device driver supports a smaller thread allocation than 512K, you can use **-s***n* to further reduce page file and virtual memory usage. For example:

  ```
  vtxnetd -s256
  ```

  (8.3.1d)
- To help track down memory leaks, we improved SQL Connection logging (generated when SSQLLOG is set to 1) to include information on errors generated by databases and to indicate when cursors are not correctly explicitly closed when implicitly closed with an %SSC_RELEASE call or when the program shuts down. (8.3.1d)

- (Windows) We changed the default SQL OpenNet server on Windows to **vtxnetd**, a multi-threaded server. This may break your code if you use VTX12_2000 or if you rely on child processes rather than threads to service requests to the SQL OpenNet service. If you use one of these, either change database drivers if possible or use **vtxnet2** by changing the daemon start-up command line in **opennet.srv**. (See "Customizing the opennet.srv file" in the "Configuring Connectivity Series" chapter of the *Installation Configuration Guide* for information.) We recommend that you switch to VTX12_ODBC. (8.3)
- (Windows) The descriptions for the **vtxnet2** and **vtxnetd** utilities are now unique, which assists with automatic Windows XP Service Pack 2 firewall unblocking. (8.1.7a)
- We added a new option (**-x**) to the **synxfpng** program. This option enables you to use **synxfpng** to ping an SQL OpenNet server (rather than an *xf*Server server). Now you can take advantage of **synxfpng**'s verbose logging, which tells you which call failed when **synxfpng** is unable to reach the server (information that isn't available with **vtxping**). Note that on UNIX, you must install *xf*Server/*xf*ServerPlus to use **synxfpng**. (8.1.7)
- (Windows) We added a new environment variable, VORTEX_HOST_HIDEGPF, which prevents an SQL OpenNet server from shutting down if a thread fails. The Connectivity Series installation automatically sets this environment variable in the **opennet.srv** file. (8.1.5)
- We added a new environment variable, VORTEX_HOST_SYSLOG, which instructs the SQL OpenNet server to generate messages for the event log on Windows, **syslog** on UNIX, and the operator console on OpenVMS when an attempt to connect to the SQL OpenNet server causes fatal errors. This environment variable is set in the **opennet.srv** file (on Windows), in the **startnet** file (on UNIX), or in **CONNECT_STARTUP.COM** (on OpenVMS) by the Connectivity Series installation. (8.1.5)
- (Windows) We added an environment variable that enables you to control the poll interval for the **SynSql** service. By setting the OPENNET_POLL_TIME environment variable in the **opennet.srv** file, you can determine how often the **SynSql** service (the **sqld** program) will check to see if SQL OpenNet daemons (**vtxnetd** or **vtxnet2**) are still running. (8.1.5)
- (Windows) We made the following improvements to the **SynSql** service:
  - It now uses an IP address for polling a **vtxnetd** daemon for failure. (Previously, **SynSql** used the local host name.) This may improve reliability on systems where there are transient domain name system (DNS) problems.
  - It now supplies more debugging information when a poll of **vtxnetd/vtxnet2** fails. This includes information on transient errors such as DNS failures and out-of socket resource errors.
  - It now records any poll failure in the event log before it shuts down.

  (8.1.5)
- (Windows) SQL OpenNet now supports around 200 concurrent connections when using **vtxnet2**. (Previously it supported no more than 108.) The limit depends on the database, database driver, and other factors. For additional information, see Microsoft Knowledge Base article 126962. (8.1.5)

- Connectivity Series now encrypts user and password information passed in connect strings through SQL OpenNet. To support this, we added the **-kn** option to the **vtxnetd** and **vtxnet2** programs. In addition, we now set the VORTEX_HOME environment variable to the connect\synodbc directory, and we add the **net.ini** file to the connect\synodbc\lib directory. The **net.ini** file contains an encryption setting for the client. For information, see "The vtxnetd and vtxnet2 programs" in the "Configuring Connectivity Series" chapter of the *Installation Configuration Guide*. (8.1)
- (Windows) We now distribute a **net_base.ini** file. This file is copied to **net.ini** if a file of that name does not already exist. (The **net.ini** file enables you to specify an encryption key for the client, set defaults for connect strings, time-outs, and an error option.) Because **net.ini** is "copied" and not "installed", it will not be removed on an uninstall or upgrade, preserving your customizations. When we have modifications that need to be made to your **net.ini** file, those changes will be reflected in **net_base.ini**. At that time we will document that the file has changed, and you will be instructed to make the appropriate changes to your **net.ini** file. (8.1)
- (Windows) We added a new logging option to the **vtxnetd** and **vtxnet2** programs. By adding an **L** after the **-a** option, you can now instruct **vtxnetd** or **vtxnet2** to send errors and informational messages about the authentication to the Windows event log. See "The vtxnetd and vtxnet2 programs" in the "Configuring Connectivity Series" chapter of the *Installation Configuration Guide*. (8.1)
- (Windows) The **opennet.srv** file is no longer created by the installation. It is now a distributed file that uses the CONNECTDIR environment variable to specify paths. (8.1)
- (Windows) We added an option (**-l**) to the **sqld** program. This option logs more detailed information about the **SynSql** start-up to the Windows event log. For information, see "The sqld program" in the "Configuring Connectivity Series" chapter of the *Installation Configuration Guide*. (8.1)
- (Windows) You can now instruct SQL OpenNet to authenticate a domain name when using the **-a** option for **vtxnetd** or **vtxnet2**. (8.1)
- (Windows) SQL OpenNet is now stopped and unregistered when you uninstall Synergy/DE. To retain the registered service in the future, you should upgrade (to the same directory) rather than uninstall/re-install. (8.1)

SQL Connection

- (OpenVMS) We updated SQL Connection to support Oracle 10g on Itanium and, as a result, added the **TDB0_9** file to the Connectivity Series distribution. (8.3.1d)
- (OpenVMS) The SYNERGYDE$ROOT logical is now defined as /SYSTEM/EXECUTIVE. (8.3.1d)
- (Windows, UNIX) We implemented connection caching within programs and across program chains. To do this, we added the following to SQL Connection:
  - SSQL_CACHE_CHAIN, an %SSC_CMD option that instructs %SSC_RELEASE to cache database connections and preserve the cache when chaining to other programs.
  - SSQL_CACHE_CONNECTION, another %SSC_CMD option that instructs %SSC_RELEASE to cache database connections. With this option, however, cached connections are closed when a program chains.
  - *Force_release*, an optional %SSC_RELEASE argument that can be set to override the SSQL_CACHE_CHAIN and SSQL_CACHE_CONNECTION options.

  Note that the SSQL_CACHE_CHAIN and SSQL_CACHE_CONNECTION options will cause errors on OpenVMS. (8.3.1c)
- (Windows) We added support for MySQL (version 5.0.24 and higher). (8.3.1c)
- (Windows) To enable SQL Connection to use shared memory access with SQL Server 2005, we added a new database driver, VTX12_SQLNATIVE, for SQL Server Native Client. (8.3.1c)
- %SSC_EXECUTE can now accept an *ncount* value that is less than the number of array elements originally bound with %SSC_OPEN. (8.3.1b)
- (Windows) For compatibility with VTX12_2000, we set the default SQL Server timeout for VTX12_ODBC to 60 seconds. (8.3.1b)
- (Windows) We improved the performance of VTX12_ODBC by using arrayed fetches to fill the SQL OpenNet prefetch buffer. We've seen as much as a four-fold improvement in performance with this change. (8.3.1a)
- We removed cursor and column limits from SQL Connection. You are now limited only by the accessed database. (8.3)
- The default number of cursors allocated by %SSC_INIT has been raised to 128. (8.3)

- (Windows) We changed the default database driver for SQL Server to VTX12_ODBC, an ODBC-based driver that has many advantages over the previous default driver, VTX12_2000. (VTX12_2000 is based on DB-Library.) In previous versions, a connect string that specified VTX12 (for an SQL OpenNet connection) or sqlserver (for a direct connection to a SQL Server client) instructed SQL Connection by default to use VTX12_2000, the latest DB-Library-based database driver. With Synergy/DE 8.3, however, a connect string that specifies VTX12 or sqlserver will by default instruct SQL Connection to use VTX12_ODBC, our ODBC-based driver for SQL Server. Because some features work differently for ODBC-based drivers than they do for DB-Library based drivers, this modification may break your code.

  Although we recommend updating your program to work with VTX12_ODBC so that it will be compatible with future versions of SQL Server (including SQL Server 2005), you can still use VTX12_2000 and our other DB-Library-based drivers by using **vtxnet2** and selecting a different driver on the Connectivity Series tab of the Synergy Configuration Program. The driver you choose there will be the driver that is used when a connect string specifies VTX12 or sqlserver. Additionally, when using SQL OpenNet, you can use VTX12_2000 by specifying VTX12_2000 in the connect string (because we now distribute **VTX12_2000.exe**).

  For more information, see "Notes on VTX12_2000, VTX12_ODBC, and VTX12_SQLNATIVE (SQL Server)" and "Using your program with different drivers and databases" in the "Creating SQL Connection Programs" chapter of the *SQL Connection Reference Manual*. Additionally, see SQL Server documentation for requirements for ODBC-based drivers. (8.3)

- We improved the %SSC_SQLLINK function so that you can now call %SSC_MOVE and %SSC_EXECUTE multiple times for the same set of %SSC_OPEN and %SSC_SQLLINK statements, which enables you to repeatedly fetch and update rows using the same cursor. (8.3)
- We added several new options to %SSC_OPEN and %SSC_CMD to support scrolling cursors. %SSC_OPEN now supports the following:
  - SSQL_SCROLL, which creates the default type of scrolling cursor for the database. Supported for VTX0, VTX11, and VTX12_ODBC.
  - SSQL_SCROLL_READ_ONLY, which creates a static (insensitive) scrolling cursor. Supported for VTX0, VTX11, and VTX12_ODBC.
  - SSQL_SCROLL_DYNAMIC, which creates a dynamic (sensitive) scrolling cursor. Supported for VTX11 and VTX12_ODBC.
  - SSQL_SCROLL_KEYSET, which creates a keyset-driven scrolling cursor. Supported for VTX11 and VTX12_ODBC.

  %SSC_CMD now supports SSQL_CMD_SCROLL, which determines which row in a scrolling cursor result set will be retrieved with the next fetch. (8.3)

- (Windows) We added a new option, log2, to **vtxnetd** and **vtxnet2**. Like the log option, log2 creates a log file (**tcm_*pid*.log**) for error information. The log2 option, however, creates smaller log files because it does not record **vtxping** events. These events often bloat **tcm_*pid*.log** files because the **SynSql** service uses **vtxping** to periodically poll **vtxnetd** or **vtxnet2** to verify that it is still running. (8.3)
- Vortex API logging and Vortex Host logging now list SUCCESS_WITH_INFO messages when using VTX12_ODBC and VTX11. If more than one such message is returned by a single ODBC API call, only the first message is logged. (8.3)
- We updated SQL Connection to support Oracle, SQL Server, and ODBC date/time formats that include microsecond precision. To do this we added the UUUUUU mask to %SSC_OPTION. (8.3)
- (Windows) The %SSC_CMD time-out option SSQL_TIMEOUT is now supported for VTX11, the ODBC database driver, and VTX12_ODBC, the ODBC database driver for SQL Server. (8.1.7d)
- We enhanced the data type conversion functions for SQL Connection to provide more descriptive text for errors. (Previously these functions returned only OUTOFMEM for errors.) (8.1.7a)
- SQL Connection now supports Oracle 10. (8.1.7)
- SQL Connection now supports bulk inserts for SQL Server when using either VTX12_ODBC or VTX12. (Previously, bulk inserts were supported only for VTX12.) To use bulk inserts for SQL Server, use SSQL_SQL_BULK_INSERT, a renamed %SSC_CMD option. (This option was formerly SSQL_SYB_BATCH_STATEMENTS.) (8.1.7)
- We updated SQL Connection so that the SSQLCONN environment variable is no longer necessary. (It is now ignored.) The maximum number of concurrent connections is now always 7. In addition, channel numbers (from 1 to 7) can be assigned in any order. (Previously, channel numbers had to be assigned consecutively starting with 1.) (8.1.7)
- On Windows, we renamed the SQL Server example programs: the **exam_syb*.dbl** programs are now **exam_sqlsrv*.dbl**, and the **stp_syb*.dbl** example programs are now **stp_sqlsrv*.dbl**. On UNIX and OpenVMS, we no longer distribute the **exam_syb*.dbl** or **stp_syb*.dbl** example programs. (8.1.7)
- (Windows) We added another example program, **exam_sqlsrvbulks.dbl**, to the Connectivity Series distribution. This program creates a table and inserts rows of data using a bulk insert, which is more efficient than a standard insert. (8.1.7)
- (OpenVMS) We have now moved to a shared image model (similar to the **.so** and **.dll** files on Windows and UNIX) for the database drivers on OpenVMS. If you are using Oracle or Oracle RDB natively, you must rebuild your database driver using **build_ssql_db.com.** If you don't, your application will fail. (8.1.7)

- (OpenVMS) We now distribute the SQL Connection shared image. (You no longer need to build it.) It is in the CONNECTDIR directory, and **CONNECT_STARTUP.COM** now sets the logical SSQLRTL to point to it. If there's an existing **ssqlrtl.exe** in SYS$SHARE, the Connectivity Series installation will rename it to **ssqlrtl_old.exe**. If you run BUILD_SSQL_STAND (from a previous distribution—we no longer distribute it), it will fail. (8.1.7)
- (Windows, UNIX) An error is now generated when exiting a routine built with the **/qcheck** compiler option in either of the following cases:
  - if SQL Connection is in use, a stack variable is bound or defined, and the cursor is not closed
  - if SQL Connection is in use, a **^m** variable is bound or defined, and the dynamic memory handle is deleted

  (8.1.7)
- We improved the logging that takes place when the SSQLLOG environment variable is set. This logging now records calls to %SSC_RELEASE, %SSC_CONNECT, %SSC_SQLLINK, as well as all calls to %SSC_MOVE, even those that don't return any rows. (Previously SQL Connection would log an %SSC_MOVE call only if it returned one or more rows.) (8.1.5d)
- %SSC_GETDBID now returns SSQL_DID_NONE if the channel (passed as the *dbchannel* argument) has been initialized with %SSC_INIT but is currently connected to a database. In previous versions, calling %SSC_GETDBID in this situation would cause a "channel not open" error. (8.1.5d)
- We added six new %SSC_CMD options:
  - SSQL_CURSOR_TYPE—Sets the ODBC cursor type.
  - SSQL_DYNAMIC_CURSORS—Enables server-side cursors on the database.
  - SSQL_KEEP_OPEN—Prevents the runtime from closing connections in a program chain.
  - SSQL_NEW_BLOBS—Instructs Oracle to use BLOB or CLOB data rather than LONG RAW or LONG data.
  - SSQL_RO_CURSOR—Instructs the database to use read-only cursors.
  - SSQL_TXN_ISOLEVEL—Specifies the ODBC cursor isolation level.

  See %SSC_CMD in the "Database Functions" chapter of the *SQL Connection Reference Manual* for more information. (8.1.5)
- (Windows) We added a new database driver for SQL Server: VTX12_ODBC. This driver uses ODBC rather than DB-Library calls, supports more than 254 bytes for CHAR and VARCHAR data types, supports row locking for SELECT transactions, and is faster with reads than VTX12. See "Building Connect Strings" in the "Creating SQL Connection Programs" chapter of the *SQL Connection Reference Manual* for information on using VTX12_ODBC, which requires DSNs (ODBC data source names), requires different syntax for connect strings and stored procedures, and returns different errors than VTX12. (8.1.5)

- We added the **stp_odbc.dbl** example program to the Connectivity Series distribution. This example program contains a stored procedure that's designed to work with the VTX12_ODBC database driver. In addition, we corrected the **exam_syb2.dbl** example program to use the (UPDLOCK ROWLOCK) hint, and we removed the **stp_ora1.dbl** example file from the distribution. (8.1.5)
- We changed the SSQL_SYB_BATCH_OVERRIDE option for %SSC_CMD to SSQL_SYB_BATCH_STATEMENT. We didn't, however, change the way the option works—it still enables or disables batch operations for Sybase and SQL Server. And we didn't remove the old option, SSQL_SYB_BATCH_OVERRIDE. Code that sets this option will still work. (8.1.5)
- (Windows) We added a new database driver, VTX8, for OLE DB access. (8.1)
- You can now use the CONNECTDIR environment variable in the .INCLUDE statement for **ssql.def**. We've moved the **ssql.def** file from the synsqlx directory to the connect directory, which is the directory CONNECTDIR is set to. (8.1)
- The SQL Connection example programs now include the **ssql.def** file from CONNECTDIR. (8.1)
- The SQL Connection example program **exam_syb2.dbl** now includes an example of a positioned row update. (8.1)
- The SQL Connection example programs **exam_syb.dbl** and **exam_syb2.dbl** now use a unique index and timestamp column to correctly create and use a table for positioned update. (8.1)
- We added the following defines for Oracle parser language types: OCI_V8_SYNTAX, OCI_V7_SYNTAX, and OCI_NTV_SYNTAX. (8.1)
- We added a new option, PLAN, to the SET OPTION command. PLAN lists the indexes used for a query. For more information on SET OPTION PLAN, see SET OPTION in Appendix B of the *xfODBC User's Guide*. (8.1)
- (Windows) To support the shared **msvcrt** file and reduce resource use, we now use **vtxapits.dll** rather than **vtxapi32.dll**. (8.1)

Version 7

This section briefly describes new features in SQL Connection version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to SQL Connection version 7.

- (OpenVMS) When connecting to a database on an OpenVMS server, you must now specify the correct driver name for the database. For example, you must use !VTX0 for an Oracle database. In previous versions, the *service* argument (now called *driver_name*) was merely a placeholder for OpenVMS. (7.5)
- (UNIX) You no longer have to execute **sqlunixbld/makessqlsrv** to add client support. ISAM*xf* and client support are pre-built. (7.3)
- (UNIX) Formerly, connect strings that included a UNIX SQL OpenNet server looked like the following:

  ```
  net:SCOTT/TIGER@localhost!vtxhost.ora
  ```

  For version 7.3 onwards, these connect strings have a new format. Because drivers now consist of a program and an associated shared object, connect strings must now have the driver's program name. (For example, the driver for Oracle consists of **VTX0**, which is the program, and **VTX0.so**, which is the associated shared object.) The following is an example of the new connect string syntax:

  ```
  net:SCOTT/TIGER@localhost!VTX0
  ```

  This instructs the **vtxned** program to find the program (**VTX0**, in the above example) and run it. The program then loads the shared object of the same name.

  If you don't want to change your connect strings for this new format, you can rename the driver program and shared object. For example, for Oracle, you could rename **VTX0** to **vtxhost.ora** and **VTX0.so** to **vtxhost.ora.so**. (7.3)
- (UNIX) You must set the LD_LIBRARY_PATH environment variable (LIBPATH on AIX, SHLIB_PATH on HP-UX) before using SQL Connection or *xf*ODBC with this release. This environment variable specifies the location of dynamically-loaded shared object files that are used by both *xf*ODBC and SQL Connection. To set this environment variable, run the **setsde** script, which sets up your Synergy environment, or use the following syntax to set it manually:

  ```
  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr2/synergyde/connect;export
  LD_LIBRARY_PATH
  ```

  Note that you must set this before starting **vtxnetd** as an OpenNet server and before using SQL Connection to access Synergy ISAM files. Also note that the LD_LIBRARY_PATH (LIBPATH on AIX, SHLIB_PATH on HP-UX) environment variable operates the same way as

the UNIX PATH environment variable: when it's specified, the operating system looks for the first match in the search path. In addition, don't add the setuid bit (+s) to the Synergy runtime (**dbr**). If you do, these environment variables won't be used. (7.3)

- When retrieving date values from a database into a numeric field, %SSC_MOVE and %SSC_EXECIO now return a Julian date value that's compatible with the Synergy Language functions %NDATE and %JPERIOD. For example, database date fields were formerly returned as *YYYYMMDD* decimal dates when used with a "YYYYMMDD" date/time mask set through %SSC_OPTION. To disable the new behavior, use the new %SSC_CMD option SSQL_OLD_ZONEDDATE and the %SSC_EXECIO option SSQL_OUTDATE. However, if you do use SSQL_OLD_ZONEDDATE, expect increased CPU time and increased memory usage. As an alternative, you can overlay your decimal field with an alpha field and pass the alpha field in place of the decimal field in %SSC_MOVE and %SSC_EXECIO. To disable the new behavior, the SSQL_OLD_ZONEDDATE option must be set before using %SSC_OPEN. (7.1)
- The default time base for date-to-numeric conversions has changed to –1721378. You can use %SSC_OPTION to change this time base. (7.1)
- We changed the default format for dates that are returned into defined alpha variables. It is now *DD-MON-YYYY*. To change the format, use %SSC_OPTION. (7.1)

New features

SQL OpenNet

- (Windows) SQL OpenNet can now be installed on a machine configured as a license client. Previously it could be installed only on a machine that was a license server. To enable this feature, the **SynLM** service must be re-registered (via Synergy/DE uninstall-reinstall or by running **synd -x** followed by **synd -r** using the current version of **synd.exe**). A reboot may be necessary. (7.5)

SQL Connection

- SSQLLOG logging now lists the connect string when a connection fails, and it indicates whether a call to %SSC_OPEN reused a cursor or created a new cursor. (7.5.1f)
- We added a new %SSC_CMD option, SSQL_ODBC_AUTOCOMMIT, that starts the ODBC driver or Synergy Database driver in autocommit mode. (7.5.1f)
- (Windows) We added SSC_EXECIO and SSC_BLOB support to **vtx11.dll**. This support is dependant on support for these functions in the underlying database. (7.5.1e)
- %SSC_OPEN logging has been enhanced to indicate the specific action taken: cursor re-use, close/open, initial open, and open of a closed cursor. (7.5.1e)
- **Vtx12** has is now non-threaded to avoid DB-Library connection problems. (7.5.1e)
- You must now supply both a user name and a password when connecting to SQL Server. (7.5.1e)

- (OpenVMS) We improved **startnet.com** to allow 10 concurrent ODBC connections. This is dependent on the number of files open and your particular operating system parameters, so we have documented the parameters (within **startnet.com**) you'll need to change if you still encounter concurrent connection limits. (7.5.1d)
- (Windows) The Connectivity Series installation now checks for and installs MDAC version 2.7 if necessary. (7.5.1c)
- We improved performance for the first fetch on a previously opened cursor when connected to Oracle. (7.5)
- You can now pass a negative value for *numvars*, an argument to %SSC_BIND and %SSC_DEFINE. A negative value instructs the routines to replace rather than append the bind/define variables. (7.5)
- We added a new routine %SSC_BIND, which enables you to bind variables in the same way you define variables with %SSC_DEFINE. This function enables you to use more than 255 variables in an INSERT statement. (7.5)
- We added an %SSC_OPEN option, SSQL_ONECOL, to override the default pre-fetch on %SSC_MOVE. (7.1)
- %SSC_DESCSQL now describes decimal or implied-decimal data types. (7.1)
- We added a new function, %SSC_REBIND, for improved performance on repeat operations. (7.1)
- We added a new function, %SSC_INDICATOR, to enable you to find out the truncation/null status of fetched rows. (7.1)
- %SSC_EXECIO no longer limits the data types that can be used. (7.1)
- Array arguments are now supported for multi-row operations on %SSC_MOVE, %SSC_OPEN, and %SSC_EXECIO. (7.1)
- SQL Connection now allows 252 concurrent open cursors. (7.1)
- SQL Connection now allows you to define up to 1024 variables or columns. (7.1)
- We added an %SSC_CMD option, SSQL_KEEP_OPEN, to keep database connection open across a program chain for Windows and UNIX systems. (7.1)
- All arguments other than *dbchannel* are now optional for %SSC_INIT. (7.1)
- The *mode* argument (formerly *rdopt)* of %SSC_COMMIT and %SSC_ROLLBACK is now optional. (7.1)
- The *ncount* argument of %SSC_MOVE is now optional and defaults to 1. (7.1)
- We added a new optional argument, *dbcursors*, to %SSC_INIT to specify the number of physical database cursors. (7.1)

Version 6

This section briefly describes new features in SQL Connection version 6 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

Arguments

- We added *row_count* as an optional argument to the %SSC_EXECUTE and %SSC_MOVE functions. (6.1)
- We added *row_count* and *errno* as optional arguments to the %SSC_GETEMSG function. (6.1)
- We added a *compute_flg* argument to %SSC_MOVE for multi-row fetches and compute-row fetches. (6.1)

Environment variables

- We added two new environment variables: SSQLLOG, which logs cursor tracking information. (6.3)

Functions

- To support new two-digit type codes, the *var_type* variable returned from %SSC_DESCSQL has been increased from **d1** to **d2**. However, to maintain support for existing calls to %SSC_DESCSQL, SQL Connection checks the size of the passed record and loads the record data appropriately. (6.3)
- The %SSC_CANCEL function no longer performs any action. (6.1.1g/6.1.2b)
- An error is now generated by %SSC_MOVE if the number of columns in the select list does not match the number of columns defined with %SSC_DEFINE or %SSC_STRDEF. (6.1.1)
- We added a new function, %SSC_BLOB, that puts or gets a binary large object or a character large object. (6.1)
- We added a new function, %SSC_SCLOSE, that performs a soft close on an open cursor. This function allows reuse of a physical cursor within SQL Connection without reopening a new cursor. (6.1)

Miscellaneous

- We now default to version 7 behavior for Oracle drivers. (6.1)
- The following license requirements have changed: (6.1)

  The UNIX client license requirement has been removed. The application code "SQLX" is no longer required for SQL Connection stand-alone or client applications. The SQL Connection database driver stand-alone license configuration is enforced with LD*xx* application codes, where *xx* is a two-digit database ID. For example, SQL Connection for Oracle requires the LD00 code to be configured on the stand-alone machine. (See **ssql.def** for database ID numbers.)

  The SQL Connection database driver server license configuration is enforced with DB*xx* application codes, where *xx* is a two-digit database ID. For example, SQL Connection for Oracle requires the DB00 code to be configured on the server. (See **ssql.def** for database ID numbers.) Additionally, a server requires an ONET license for the number of concurrent connections.
- We added a *warn* argument to %SSC_MOVE which returns true (non-zero) if any warning occurred on the move. (6.1)

10

*xf*Server

Version 9

This section briefly describes new features in *xf*Server version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- (Windows) The **setruser** utility now displays additional messages when both global and local RUSER values are used. Removing a local RUSER with **-d** will display an informational message when a global RUSER still exists. Likewise, removing a global RUSER with **-gd** will display a message when a local RUSER still exists. In addition, when setting RUSER, a warning message will be displayed when an RUSER is already set in the opposite (global or local) environment. (For example, **setruser -g** will display "Warning: This RUSER will be hidden by local RUSER *user*" if a local RUSER already exists.) (9.5.3)
- (UNIX) Rsynd was enhanced to use the location specified in the RSFILPATH environment variable (if set) for the location of files created without a path specified. (9.5.3)
- (OpenVMS) We made several enhancements to the **servstat** program. The servers are numbered in the list, making it easier to select the one you want to view. In addition, the list now distinguishes between *xf*Server and *xf*ServerPlus. The display status shows additional information when encryption is enabled: whether master or slave, the cipher level (low/medium/high), and the certificate file being used. (9.5.3)
- (Windows, UNIX) We added a buffering feature, which improves performance of sequential WRITES and PUTS calls. Records waiting to be written to the server are held in a buffer on the client and then written all at once, which improves file loading performance. This feature is supported for relative and sequential files that are opened in output or append mode (WRITES) and for stream files that are opened in output or append mode (WRITES or PUTS). Buffering is turned on by setting the SCSPREFETCH environment variable (on the client) to the size of the buffer. (Setting SCSPREFETCH now turns on both prefetching and buffering.) (9.5.1a)
- *xf*Server now supports the sending and receiving of encrypted network packets between client and server. (Clients must be version 9.3 or higher.) We added three new command line options to **rsynd** to support encryption: **-encrypt**, **-cert**, and **-cipher** (/ENCRYPT, /CERTIFICATE, and /CIPHER on OpenVMS). On Windows, you can also specify encryption settings from the Synergy Configuration Program. (9.3)
- We improved the performance of the ISAM operations STORE, WRITE, and DELETE to *xf*Server by 30 to 40 percent when **synbackup** is in use. (9.1.5a)

Version 8

This section briefly describes new features in *xf*Server version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*Server version 8.

- (Windows) *xf*Server using Windows authentication security now allows domain users when using an NT domain under NTLM security. This requires explicitly registering the *xf*Server service manually. To activate SSPI with RUSER, as with "rsynd -r -s" register as follows: "rsynd -r -SSPI=+NTLM". Restricted mode requires the following: "rsynd -r -SSPI=NTLM". (8.3.1c)
- *xf*Server supports clients running Synergy/DE version 6.1 and higher. Previously we supported version 5.7.9 and higher. (8.3)
- We no longer support *xf*Server data access on the *xf*ServerPlus port. If you were running both *xf*Server and *xf*ServerPlus on the same port, you must reconfigure one of these services to run on a different port. The **rsynd -b** option (/ENABLE_DATA on OpenVMS) will have no effect. If you use this option and logging is turned on, a warning message will be logged. (8.3)
- (Windows) If RUSER is set in both the registry and the environment, the setting in the environment now takes precedence over the setting in the registry. This change especially benefits users who need to configure *xf*ServerPlus to access data remotely using *xf*Server running in secure mode (which requires that RUSER be set in the environment) and also need to have RUSER set in the registry for other *xf*Server users. (8.3)
- (Windows) Because of the potential for data corruption, a remote file specification is no longer permitted to reference another remote file specification when the first file specification is on Windows. (8.3)
- (Windows 2000/XP) We changed the default installation directory for *xf*Server to C:\Program Files\Synergex\Synergyde. The new directory will be used for new installations. If you are upgrading, the installation will default to your current directory. (8.1)
- If you have an existing *xf*Server installation and change the directory when installing a newer version, you will need to remove your *xf*Server services and then re-add them. See the "Configuring xfServer" chapter of the *Installation Configuration Guide* for instructions. (8.1)

New features

- (Windows) *xf*Server now supports Windows authentication security, which uses the Windows operating system to authenticate Windows clients. Windows authentication can be used by itself (referred to as "Restricted" mode) or in combination with existing RUSER security in Secure mode. See "Understanding xfServer security" in the "Configuring *xf*Server" chapter of the *Installation Configuration Guide*. (8.3)
- (Windows) We added a new option, **-sspi**, to **rsynd** to support Windows authentication. (8.3)
- (Windows) To support Windows authentication, the RUSER registry setting (environment variable) now recognizes "SSPI" as a special value. Setting RUSER to SSPI (and not specifying a password) indicates to *xf*Server that it should use Windows authentication rather than RUSER to authenticate the client. (8.3)
- (Windows) The **synckusr** utility has been updated to recognize an RUSER value of SSPI without a password as Windows authentication mode. It also now checks an RUSER setting in the environment before one in the registry. In addition, when RUSER is not set, the first line of the output will be "RUSER not set. *Username* from GetUserName system call (implied RUSER)". See "The synckusr Utility" in the "Configuring *xf*Server" chapter of the *Installation Configuration Guide*. (8.3)
- (Windows, UNIX) **Rsynd** now supports the **-u** option for *xf*Server. When used with *xf*Server, this option enables you to specify a default user account and password that will be used as the persona for all clients when *xf*Server is run in non-secure mode. (8.3)
- (OpenVMS) We added a new option, /DEFAULT_USER, to **rsynd**. This option enables you to specify a default user account that will be used as the persona for all clients when running *xf*Server in non-secure mode. (8.3)
- (Windows) We added a new **-g** option to **setruser**, which enables you to set RUSER at a global level in the registry. When **-g** is specified, **setruser** prompts for the user name and password and then returns the encoded string and sets RUSER to this value in the registry under HKEY_LOCAL_MACHINE\Software\Synergex. See "The setruser Utility" in the "Configuring *xf*Server" chapter of the *Installation Configuration Guide*. (8.3)
- (Windows) When the *xf*Server service is stopped, all connections to *xf*Server will now be shut down cleanly, and on the server a "Connection terminated on request" message will be logged in the application event log. (8.3)
- (Windows) You can close a specific *xf*Server connection with the new **synxfmon -k** option. See "The Monitor Utility for Windows" in the "General Utilities" chapter of *Synergy Language Tools*. (8.3)
- (Windows) We added a verbose logging option for *xf*Server event logging. You can turn on verbose logging by selecting the "Verbose logging" checkbox on the xfServer Information dialog in the Synergy Configuration Program. Regular event logging now logs only user connections; verbose logging logs user connections plus any errors that occur. (8.3)
- (Windows) *xf*Server errors that are the result of licensing failures are now logged to the application event log. (8.3)

- (OpenVMS) The installation has been modified to allow either one or two servers to be configured at install time: one *xf*ServerPlus and/or one *xf*Server. (8.3)
- We added a prefetch feature, which improves the performance of sequential READS calls. Prefetched records are stored in a buffer on the client. You can use this feature with files of any type that are open in input mode and with relative or ISAM files that are open for update and use the LOCK:Q_NO_TLOCK option. To turn on prefetching, set the SCSPREFETCH environment variable (on the client) to the size of the buffer. See "Prefetching and Buffering Records" in the "Configuring *xf*Server" chapter of the *Installation Configuration Guide*. (8.3)
- (UNIX) We increased the internal file cache limit for the Monitor utility to 2,048. (8.1.7d)
- (UNIX) We added support to **rsynd** to allow individual **synrc** files for multiple **rsynd** daemons that are running on different ports. When **rsynd** is started, it will first read the **/etc/synrc** file and then read the **/etc/synrc.*port_number*** file if it exists. The settings in the **/etc/synrc.*port_number*** file will take precedence over the settings in the **/etc/synrc** file. (8.1.7a)
- (Windows) *xf*Server and related utilities (such as **setruser.exe** and **synckusr.exe**) now handle large Windows DOMAIN\USER combined names up to 255 bytes. (8.1.7a)
- (UNIX) *xf*Server will now strip all leading white-space characters (not just blanks) when reading the **/etc/synrc** file and the user's **.synrc** file. (8.1.7a)
- (OpenVMS) We now install **synxfpng** with Synergy Language (in [dbl.bin]) instead of with *xf*Server/*xf*ServerPlus (in [server]). In addition, the installation now creates a verb for **synxfpng** so that you no longer have to create a symbol for it. (8.1.7)
- We now document the **-v** option for **synxfmon**. This option enables you to view additional operational information, as well as the user name and remote port number of the machine being reported on. See "The Monitor Utility for Windows" in the "General Utilities" chapter of *Synergy Language Tools* for details. (8.1.7)
- (Windows) We made some slight performance improvements in *xf*Server. (8.1.5)
- (Windows) We reduced the memory footprint of **rsynd** by delay loading the spooling DLLs and certain related system DLLs. This may also reduce desktop heap contention. (8.1.5)
- (OpenVMS) We added a new qualifier to **rsynd**, /SHUTDOWN=ALL, which enables you to stop **rsynd** and terminate all existing connections. (8.1.5)
- (OpenVMS) *xf*Server now allows more than 256 channels to be opened. (8.1.5)
- (OpenVMS) The **servstat** program can now process command line options if it is set up as a foreign symbol. (8.1.5)
- We enhanced the help message that displays when you run **rsynd** with the **-h** option. It now includes descriptions of options and multiple usage examples. (8.1.3)
- (Windows) We made a small improvement in performance when closing sockets. (8.1.3)

- (OpenVMS) We now distribute the **synxfpng** utility on OpenVMS. This utility can be used to debug any TCP/IP connection, even if *xf*Server is not involved. To use it, set up a symbol for it:

  ```
  $ synxfpng:==$synergyde$root:[server]synxfpng.exe
  ```

  (8.1.3)

- (Windows) We added a new environment variable, RSFILPATH, which defines the default directory on the server for files that do not have a path specification. If RSFILPATH is not set, the default directory on Windows NT is C:\Winnt\Profiles\All Users\Documents; on Windows 2000/XP, it is C:\Documents and Settings\All Users\Documents. (8.1)
- (Windows) We added a new option, **-s**, to the **synckusr** utility to enable you to look up the default file directory on a specified server. This is the directory that a file without a complete path will be placed in. (8.1)
- (Windows) **Rsynd** now recognizes user-specific environment settings set under HKEY_CURRENT_USER\SOFTWARE\Synergex\Synergy xfServer\Synrc. Environment variables set here override those set for a specific instance of *xf*Server and for all instances of *xf*Server. See "Defining environment variables manually" in the Windows section of the "Configuring xfServer" chapter of the *Installation Configuration Guide* for details. (8.1)
- (Windows) We added a new registry setting, ENABLEUSERHIVE, which must be set when you use user-specific environment variables (see above). See "Defining environment variables manually" in the Windows section of the "Configuring xfServer" chapter of the *Installation Configuration Guide* for details. (8.1)
- (Windows) We changed the secondary location for the temporary file that is created when you use the SORT statement with *xf*Server. Now, if TMP or TEMP is not set in the Windows registry, the file is placed in C:\Winnt\Profiles\All Users\Documents on Windows NT and in C:\Documents and Settings\All Users\Documents on Windows 2000/XP. (8.1)
- (Windows) All *xf*Server services are now stopped and unregistered when you uninstall Synergy/DE. In the future, to retain your registered services, you should upgrade (to the same directory) rather than uninstall/re-install. (8.1)
- (UNIX) We added a new option (**-P**) to the monitor utility. The command "query -P *nnnn option*" (where *nnnn* is a port number and *option* is a query option) will query the *xf*Server monitor on the specified port. This enables you to monitor an *xf*Server running on a port other than the default (2330). (8.1)

For more information on *xf*Server, see the "Configuring xfServer" chapter of the *Installation Configuration Guide* and the *xf*Server release notes, **REL_SRV.TXT**.

Version 7

This section briefly describes new features in *xf*Server version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- (Windows) We added a new **rsynd** option, **-rs**, which enables you to register and then start **rsynd** in one step. (7.5.1e)
- (Windows) All *xf*Server services are now shut down when you uninstall Synergy/DE. You no longer have to stop them manually. (7.5.1e)
- The SCSCOMPR environment variable (which is equivalent to the Compress Data Packets checkbox in the Synergy Configuration Program), can now be set on only the server if desired. (It no longer has to be set on both the server and the client.) (7.5.1a)
- (Windows) *xf*Server can now be installed on a machine configured as a license client. Previously it could only be installed on a machine that was a license server. To enable this feature, the SynLM service must be re-registered (via Synergy/DE uninstall-reinstall or by running **synd -x** followed by **synd -r** using the current version of **synd.exe**). A reboot may be necessary. (7.5)
- (OpenVMS) We added a new **rsynd** option, **/LOG_LEVEL**, which can be used to control the logging level and whether messages are sent to the console. (7.5)
- (Windows) We added a Monitor utility (**synxfmon.exe**) that tells you which files are open, who opened them, and whether those files are locked. See "The Monitor Utility for Windows" in the "General Utilities" chapter of *Synergy Language Tools* for more information. (7.5)
- When *xf*ServerPlus is enabled (**-w** or **/XFPL_ENABLE**), the *xf*Server data access functionality on that same port is now disabled by default. To enable it, use the new **rsynd -b** or **/DATA_ENABLE** option. (7.5)
- (Windows) We added the new Synergy Configuration Program to the Start menu. To run it, select Synergy/DE > Utilities > Synergy Configuration Program. Use this new program to configure *xf*Server. (7.3)
- The default port for *xf*ServerPlus has been changed to 2356 so that it no longer conflicts with the default port for *xf*Server, which remains 2330. (7.3)
- (Windows) Logging is disabled by default. (7.3)
- We enhanced the **synckusr** utility and added complete documentation for it. (7.3)
- (OpenVMS) We added a command-line interface to **rsynd**. (7.1)

11

*xf*ServerPlus

Version 9

This section briefly describes new features in *xf*ServerPlus version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*ServerPlus version 9.

- Version 9.1.5 and earlier Synergy Method Catalog files must be converted for use with *xf*ServerPlus 9.3. See the release notes file (**REL_XFPL.TXT**) for conversion instructions. (9.3)

New features

*xf*ServerPlus

- The *xf*ServerPlus log file now displays milliseconds in the date and time stamp for the packet received, packet returned, and the user function calls. (9.5.3)
- The format YYYYMMDDHHMISSUUUUUU is now supported for both parameters and return values when coercing a decimal data type to a DateTime. (9.5.3)
- We added support for coercing decimal data types to decimals and nullable decimals for parameters and return values. (9.5.3)
- (Windows, UNIX) We added a new command line switch, **-text**, to **rsynd**. It takes as a parameter a text string, which is added to the end of the command line when *xf*ServerPlus is started. If the text string contains a single "%s", it is replaced with the IP address of the client. (9.5.3)
- (OpenVMS) We made several enhancements to the **servstat** program. The servers are numbered in the list, making it easier to select the one you want to view. In addition, the list now distinguishes between *xf*Server and *xf*ServerPlus. The display status shows additional information when encryption is enabled: whether master or slave, the cipher level (low/medium/high), and the certificate file being used. (9.5.3)
- Attribute support (**dbl2xml**) now supports group arguments .INCLUDEd from the repository. A group argument is processed as a single alpha field, with the size taken from the repository. (9.5)

- (Windows, UNIX) We added some new error reporting by *xf*ServerPlus to handle problems with ELBs. Previously, these conditions would simply result in a "routine not found" error. The new errors are more specific:
  - 2020, Old ELB file format detected - relink.
  - 2021, Bad ELB detected.
  - 2022, ELB file built with opposite 'endian'.
  - 2023, ELB file built with opposite bit size.

  (9.5)
- The 32-bit SMC/ELB utility now supports both 32-bit and 64-bit ELBs. (9.5)
- (OpenVMS) **Rsynd** has been enhanced to allow it to be restarted while a client still has a connection open on the specified port. (9.5)
- You can now add attributes and documentation comments to your Synergy code and run the **dbl2xml** utility to create an XML file, which can then be imported into the SMC. This feature enables you to populate the SMC without using the MDU to perform data entry, and should make it easier to keep your code and SMC in sync. (9.3)
- We added support for encryption of network packets sent and received between *xf*ServerPlus and the .NET and Synergy clients. (Clients must be version 9.3 or higher.) We added three new command line options to **rsynd** to support encryption: **-encrypt**, **-cert**, and **-cipher** (/ENCRYPT, /CERTIFICATE, and /CIPHER on OpenVMS). On Windows, you can also specify encryption settings from the Synergy Configuration Program. (9.3)
- For a .NET client, *xf*ServerPlus supports the passing of enumerations (defined in the repository) as parameters and method return types, as well as enumerations that occur as fields within a structure parameter. (9.3)
- We changed the file format for the Synergy Method Catalog (SMC) files to accommodate new fields. Pre-8.3 SMC files must be converted for use with *xf*ServerPlus 9.1.3, but 8.3 and 9.1 SMC files can be used without conversion. See the "SMC Conversion Utility (smccnvrt)" section of **REL_XFPL.TXT** for details on converting SMC files. (9.1.3)
- The SMC/ELB Comparison utility (**smc_elb.exe**) supports both version 8.3/9.1 format (rev3) and version 9.1.3 format (rev4) SMC files. (9.1.3)
- The **genxml** utility supports both version 8.3/9.1 format (rev3) and version 9.1.3 format (rev4) SMC files. (9.1.3)
- The **genxml** utility now checks structure sizes in the SMC against the corresponding structures in the repository and issues a warning if there are discrepancies. (9.1.3)
- We added support for a Synergy System.String data type as either a parameter or a return value when using an *xf*NetLink .NET client. (9.1.3)
- We added support for passing a Synergy System.Collections.ArrayList class as a parameter when using an *xf*NetLink .NET client. (9.1.3)

Method Definition Utility

- We added “decimal” and “nullable decimal” to the list of coerced types available for parameters and return values when the type of the parameter or return value is decimal. (9.5.3)
- We added a new DateTime Format, YYYYMMDDHHMISSUUUUUU, to the drop-down lists for the Format field in both the Method Definition and the Parameter Definition window. (This field used to be named “Date format”; we shortened it to “Format” to make room for this new format.) (9.5.3)
- We now support an update option for importing. You can use the **mdu -u** command line option or the “Update interfaces” option in the MDU GUI to update existing interfaces and add new interfaces. We removed the “Add new interfaces” option from the GUI because you can select “Update interfaces” instead and then choose only the new interfaces you want to add. (9.3)
- We added an Encryption checkbox to the Method Definition window that is used to indicate that a method requires encryption when slave encryption is enabled. Encryption is supported only for .NET and Synergy clients. (9.3)
- The MDU now supports selecting enumerations defined in the repository as method return values and as parameters. (9.3)
- We added support for $ in filenames. Since this is permitted in Synergy routine names, it was an inconvenience for customers who had to rename routines throughout an application. (9.3)
- The Verify utility now verifies enumerations in the repository as well as structures. It will generate warnings when return types or parameter types contain an enumeration that is not found in the current repository. (9.3)
- We added a **-v** option to **mdu.dbr** so that you can run the Verify Catalog utility from the command line. The existing **-l** option, which is used to specify the logfile name, was modified to be used for the verify log (in addition to the import log). (9.1.3)
- We added a new field to the Method Definition window called “Coerced type”, which enables you to specify a non-default data type for the return value on the client. This field is used only by *xf*NetLink .NET. (9.1.3)
- We added a new field to the Parameter Definition window called “Coerced type”, which enables you to specify a non-default data type for the parameter on the client. This field is used only by *xf*NetLink .NET. (9.1.3)
- We added a new field to both the Method Definition window and the Parameter Definition window called “DateTime format”, which is used to indicate the desired format when the coerced type is DateTime or Nullable DateTime. This field is used only by *xf*NetLink .NET. (9.1.3)
- We added a new field to the Parameter Definition window called “ArrayList”, which is used to indicate that the parameter is a Synergy System.Collections.ArrayList class. This field is used only by *xf*NetLink .NET. (9.1.3)
- We added a new field to the Parameter Definition window called “DataTable”, which is used to indicate that a structure parameter defined as a structure collection or ArrayList should be created as a DataTable in the C# class. This field is used only by *xf*NetLink .NET. (9.1.3)

- The “Collection” field on the Parameter Definition window was renamed “Structure collection”. (9.1.3)
- On the Parameters list window, the columns for Ary (array) and Col (collection) were combined into a single column named CollType (collection type), which will display array, arylist (for ArrayList), or str col (for structure collection). (9.1.3)
- The MDU import and export utilities were updated to handle the new fields and data types introduced in 9.1.3. (9.1.3)
- The **smccnvrt** utility now converts SMC files from any previous version to 9.1.3 format. See the “SMC Conversion Utility (smccnvrt)” section of **REL_XFPL.TXT** for details on converting SMC files. (9.1.3)

Version 8

This section briefly describes new features in *xf*ServerPlus version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*ServerPlus version 8.

*xf*ServerPlus

- Pre-8.3 Synergy Method Catalog files must be converted for use with *xf*ServerPlus 8.3. Refer to the *xf*ServerPlus release notes file (**REL_XFPL.TXT**) for conversion instructions. (8.3)
- We no longer support *xf*Server data access on the *xf*ServerPlus port. If you were running both *xf*Server and *xf*ServerPlus on the same port, you must reconfigure one of these services to run on a different port. The **rsynd -b** option (/ENABLE_DATA on OpenVMS) will have no effect. If you use this option and logging is turned on, a warning message will be logged. (8.3)
- (Windows) If RUSER is set in both the registry and the environment, the setting in the environment now takes precedence over the setting in the registry. This change especially benefits users who need to configure *xf*ServerPlus to access data remotely using *xf*Server running in secure mode (which requires that RUSER be set in the environment) and also need to have RUSER set in the registry for other *xf*Server users. (8.3)
- (UNIX) To improve *xf*ServerPlus security, **rsynd** now verifies that the password specified with **-u** has been encoded using **setruser**. A clear text password is no longer permitted. (8.3)
- (UNIX) If no password is specified with the **-u** option, the user starting *xf*ServerPlus must be root (uid=0) or signed on as the user specified with **-u**. This is the documented behavior, but it was not previously being enforced. (8.3)
- (OpenVMS) The account used to run *xf*ServerPlus sessions must have the SHARE privilege. This requirement is a result of the changes we made to the way sockets are used to establish communication. (8.3)
- (Windows 2000/XP) We changed the default installation directory to C:\Program Files\Synergex\Synergyde. The new directory will be used for new installations. If you are upgrading, the installation will default to your current directory. (8.1)
- If you have an existing *xf*ServerPlus installation and change the directory when installing a newer version, you will need to remove your *xf*ServerPlus services and then re-add them. See the "Configuring and Running xfServerPlus" chapter of *Developing Distributed Synergy Applications* for instructions. (8.1)

Method Definition Utility (MDU)

- Pre-8.3 Synergy Method Catalog files must be converted for use with *xf*ServerPlus 8.3. Refer to the *xf*ServerPlus release notes file (**REL_XFPL.TXT**) for conversion instructions. (8.3)

New features

*xf*ServerPlus

- The logging entry for function return value is now included in the log when XFPL_FUNC_INFO is set to ALL. Previously, it was included only when debug logging was turned on. (8.3.1d)
- (Windows, UNIX) *xf*ServerPlus now supports separate log files for each session. This is the default behavior. You can specify a single log file by setting XFPL_SINGLELOGFILE to ON in the **xfpl.ini** file. The primary advantage to multiple log files is improved performance. See "Setting Options for the xfServerPlus Log" in the "Configuring and Running *xf*ServerPlus" chapter of the *Developing Distributed Synergy Applications* manual. (8.3)
- We changed some logging defaults. XFPL_LOG is now set to OFF (i.e., *xf*ServerPlus logging is off by default), and XFPL_SESS_INFO is set to NONE. These are both the programmatic defaults and the values in the distributed **xfpl.ini** file. (8.3)
- (UNIX) *xf*ServerPlus (**xfpl.dbr**) now logs errors to the system log (**syslog**), regardless of whether *xf*ServerPlus logging is turned on. Errors found in the **xfpl.ini** file are also logged to **syslog**. If you need to log only errors, you can turn off the *xf*ServerPlus log and just refer to the system log. This will improve performance. (8.3)
- (OpenVMS) *xf*ServerPlus (**xfpl.dbr**) now sends errors to the operator console, regardless of whether *xf*ServerPlus logging is turned on. Errors found in the **xfpl.ini** file are also sent to the operator console. If you need to see only errors, you can turn off the *xf*ServerPlus log and just refer to the operator console. This will improve performance. (8.3)
- (Windows) Errors found in the **xfpl.ini** file are now written to the Windows application event log. (8.3)
- The distributed method catalog includes method descriptions (in the new "Method desc" field). These descriptions are also in the **defaultsmc.xml** file. (8.3)
- We changed the way that *xf*NetLink and *xf*ServerPlus use sockets to establish communication. Version 8.3 *xf*NetLink clients will connect to *xf*ServerPlus using the new method; pre-8.3 clients and clients running in debug mode will continue to use the old method. This change was made to prevent problems that occurred when there were multiple firewalls in a WAN. (8.3)
- We added a new data type packet code, BI, to support "binary (handle)" type parameters, which can be used by *xf*NetLink Java and *xf*NetLink .NET. (8.3)
- (Windows XP, 2003) **Rsynd** now validates that the username/password is a valid account on the local machine or domain before registering the service. Previously, only the username was checked, and the password was not validated until the first call from the client was made. (8.3)

- (Windows) By default, *xf*ServerPlus now uses a single desktop for *all* dbs processes started by *all* instances of *xf*ServerPlus (when XFPL_DBR is not set). Previously, each dbs used its own desktop. This could cause processes to fail to start due to desktop heap limitations.

  Optionally, you can set XFPL_UNIQUE_DESKTOP, which causes *each* instance of *xf*ServerPlus to use a desktop for all of its dbs processes. You may need to do this because there is currently a bug in Windows Server 2003 SP1 and in Windows XP SP2 that limits the number of processes per desktop to 40. Synergex is working with Microsoft to resolve this issue. Note that this bug does not exist on Windows Server 2000. See the release notes file, **REL_XFPL.TXT**, for details on setting XFPL_UNIQUE_DESKTOP. (8.3)

- (Windows) The **genxml** utility is now also distributed with *xf*Series so that the MDU's Export Methods feature is available even when Professional Series is not installed. (8.3)
- (OpenVMS) The installation has been modified to allow either one or two servers to be configured at install time: one *xf*ServerPlus and/or one *xf*Server. (8.3)
- The **genxml** utility now maintains the case of parameter names when it writes the XML file. Previously, parameter names were changed to lowercase, which caused a "duplicate parameter" error in the Method Definition Utility when a routine had two parameters that differed only in case, and an export followed by an import was performed. (8.3)
- (Windows) **Rsynd** now supports user principal name (UPN) format (*user_name@domain*), in addition to the down-level format (*DOMAIN\user_name*), when specifying the user name with the **-u** option. (8.3)
- (Windows) The service runtime (**dbs.exe**) now logs traceback information from fatal runtime errors to the Windows application event log. (8.1.7d)
- (Windows) *xf*ServerPlus (**xfpl.dbr**) now logs all errors to the Windows application event log, regardless of whether logging is turned on. If you want error-only logging, you can turn off the *xf*ServerPlus log and just refer to the event log. This will improve performance. (8.1.7d)
- We added a 120 second time-out to the receive of the acknowledgement from the client after the first packet is sent from *xf*ServerPlus. The same time-out value is used for regular and debug sessions; it is not configurable. (8.1.7d)
- (UNIX) We added support to **rsynd** to allow individual **synrc** files for multiple **rsynd** daemons that are running on different ports. When **rsynd** is started, it will first read the **/etc/synrc** file and then read the **/etc/synrc.*port_number*** file if it exists. The settings in the **/etc/synrc.*port_number*** file will take precedence over the settings in the **/etc/synrc** file. This means that you can, for example, now set XFPL_SMCPATH and XFPL_INIPATH separately for each instance of *xf*ServerPlus. (8.1.7a)

- If *xf*ServerPlus encounters an error while reading the **xfpl.ini** file, it will output the error to the *xf*ServerPlus log file (**xfpl.log**), and the remote connection will be terminated. If an error is encountered, the remainder of the **xfpl.ini** file will continue to be processed so that all errors within the file can be logged at one time. If logging is not turned on, the connection will simply terminate. Previously, no errors were output to the log file and the remote connection would continue. (8.1.7a)
- (Windows) *xf*ServerPlus now handles large Windows DOMAIN\USER combined names up to 255 bytes. (8.1.7a)
- (Windows, UNIX) The translation value for XFPL_LOGICAL settings in the **xfpl.ini** file can now be a maximum of 1,024 characters instead of 256 characters. An error will be output to the log file if the translation value exceeds 1,024 characters, and the connection shall be terminated. (8.1.7a)
- (OpenVMS) We added a new option (11) to the **servstat** program to close and reopen a new version of the current *xf*ServerPlus logfile. This enables you to examine the logfile without shutting down **rsynd**. See "The servstat Program" in the "General Utilities" chapter of *Synergy Language Tools*. (8.1.7a)
- (OpenVMS) The **xfpl** sessions waiting for a client connection no longer die after 10 minutes. They will remain in the free pool until **rsynd** is shut down. (8.1.7a)
- (OpenVMS) We now install **synxfpng** with Synergy Language (in [dbl.bin]) instead of with *xf*Server/*xf*ServerPlus (in [server]). In addition, the installation now creates a verb for **synxfpng** so that you no longer have to create a symbol for it. (8.1.7)
- We added support for collection parameters (ArrayLists). *xf*ServerPlus can now send a variable length array of structures to either an *xf*NetLink Java or *xf*NetLink .NET client. For details, see "Returning a Collection of Structures" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual. (8.1.7)
- We significantly improved *xf*ServerPlus performance by reducing the overhead required to encode packets, decode packets, and call Synergy routines. These changes will improve the scalability of *xf*ServerPlus. (8.1.5)
- We added a compression option to *xf*ServerPlus. You can turn compression on by setting XFPL_COMPRESS in the **xfpl.ini** file. This will compress both sent and received data containing repeated zeroes or spaces. For details, see "Configuring Compression" in the "Configuring and Running *xf*ServerPlus" chapter of the *Developing Distributed Synergy Applications* manual. (8.1.5)
- (Windows, UNIX) We added a new method for debugging your remote Synergy routines, which enables you to connect to the *xf*ServerPlus machine via Telnet and run a debugger session in the Telnet window. To support this, we added a new option, **-rd**, to **rsynd**. For details, see "Debugging Your Remote Synergy Routines" in the "Configuring and Running *xf*ServerPlus" chapter of the *Developing Distributed Synergy Applications* manual. (8.1.5)

- When XFPL_FUNC_INFO is set to CRITICAL or ALL in the **xfpl.ini** file, the log now includes the time the packet was received by *xf*ServerPlus and the time the packet was returned to the client. Previously, it logged the time that parsing of the packet started; this has been removed since it is essentially the same as the time the packet was received. (8.1.5)
- We reduced the memory footprint of **rsynd** by delay loading the spooling DLLs and certain related system DLLs. This may also reduce desktop heap contention. (8.1.5)
- (OpenVMS) The **servstat** program has been enhanced. It now displays *xf*ServerPlus session information, has an option to purge all sessions in the free pool and start new sessions, and can process command line options if it is set up as a foreign symbol. (8.1.5)
- (Windows) We made a small improvement in performance when closing sockets. (8.1.3)
- (OpenVMS) We now distribute the **synxfpng** utility on OpenVMS. This utility can be used to debug any TCP/IP connection, even if *xf*Server is not involved. To use it, set up a symbol for it:

  ```
  $ synxfpng:==$synergyde$root:[server]synxfpng.exe
  ```

  (8.1.3)
- (Windows) All *xf*ServerPlus services are now stopped and unregistered when you uninstall Synergy/DE. In the future, to retain your registered services, you should upgrade (to the same directory) rather than uninstall/re-install. (8.1)
- *xf*ServerPlus now supports passing parameters larger than 64K in size. See "Handling Variable-Length and Large Data" and "Passing Arrays Larger Than 64K" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual for more information. (8.1)
- We have improved performance, especially when passing structure and array parameters. (8.1)
- *xf*ServerPlus now supports using European formatting for implied decimals if flag 1 is set in the FLAGS routine. You must call the FLAGS routine in each Synergy method that you want to use European formatting. This is because *xf*ServerPlus resets this flag after every call so that data passed across the wire is formatted properly. (8.1)
- The **genxml** program now includes a check to ensure that the sum of the field sizes in a structure match the structure size. (8.1)

For more information on *xf*ServerPlus, see the *Developing Distributed Synergy Applications* manual and *xf*ServerPlus release notes, **REL_XFPL.TXT**.

Method Definition Utility (MDU)

- We added a new field to the Method Definition window called “Method name”. The method name is used by *xf*NetLink Java (with class wrappers), *xf*NetLink COM, and *xf*NetLink .NET to invoke the Synergy routine within your client code. Previously, the Synergy routine name was used for this purpose. Adding a method name field gives you much-needed flexibility, as it allows the method name on the client to differ from the routine name on the server.

  The new method name field is particularly useful when doing pooling for COM and .NET clients because the pooling support methods must be named with specific names. Previously, this meant that the Synergy routines they represented also had to be named with those specific names. This was no problem if your application required only one method of each type. However, if, for example, your application required two “Initialize” methods, you had to create separate ELBs on the Synergy side to hold each of the routines since they had the same name. Now, the Synergy routines can be named anything you like; it is only the method names that need to adhere to the naming conventions. See “Understanding Routine Name, Method Name, and Method ID” in “Defining Your Synergy Methods” chapter of the *Developing Distributed Synergy Applications* manual for more information. (8.3)

- You can toggle between displaying the new method name and the method ID on the Methods list window by selecting Functions > Toggle View or pressing CTRL+V. The toggle setting is respected when you select “by Method” from the Sort menu, do a find on the Methods list, select methods to import, and do a find on the Select Methods (to import) list. (8.3)
- We added support for a parameter name that is different than the structure name. This enables you to have two or more parameters that use the same structure within a method. (8.3)
- We added three new fields for entering descriptive comments about the method: Method desc and Return desc on the Method Definition window and Description on the Parameter Definition window. If you are creating a Java jar file or .NET assembly, the information in these fields will be inserted into the generated class files as documentation comments. (8.3)
- You no longer are required to enter the maximum and required number of parameters on the Method Definition window. Those two fields are now display-only. After you enter parameters for a method, the MDU will calculate the maximum number of parameters and the number of required parameters and display them. (8.3)
- The Interface name field and the Parameter name field were increased from 31 to 50 characters. (8.3)
- You can now change the method ID by selecting the “Modify” function, and then accessing the Method ID dialog from the Method Definition window. Consequently, we removed the “Rename” option. (8.3)
- (UNIX, OpenVMS) In order to make room for the new fields on the Method Definition window, we changed the ELB name field from displaying 255 characters to displaying only the first 51 characters of the ELB name. To enter more than 51 characters, select F10 (or Functions > Edit ELB Name) when the cursor is in the ELB name field. A dialog will display in which you can enter a longer value. (8.3)

- On the Parameter Definition window, we changed the names of buttons and menu entries to make the process more intuitive. When you are entering parameters for the first time, enter data for the first parameter, and then select the Next button (or Functions > Next Parameter). The parameter data will be saved and the window will redisplay so that you can add another parameter. When you are done adding the last parameter, select Done. (8.3)
- We removed the option to change the format for the date. The format for the "Date last updated" field on the Method Definition window is now always MM/DD/YYYY. (8.3)
- We removed the Reports feature, as it has not been updated to handle the new fields added in version 8.3. (8.3)
- User-specific settings for the MDU are now stored in the following files: **synuser.ini** (Windows), **$HOME/mdu.ini** (UNIX), **SYS$LOGIN:mdu.ini** (OpenVMS). Previously, the MDU was not able to store preferences by user. (8.3)
- The MDU now attempts to open the repository files on start-up. (Previously, the MDU did not attempt to open the repository files until you defined a structure parameter.) If you explicitly specify a repository on the command line (with **-r** or with **-m** and **-t**) when starting the MDU, and it cannot be found or either of the files cannot be opened, the MDU will generate an error and terminate. If you start the MDU from the Windows Start menu or do not explicitly specify a repository when starting from the command line, the MDU will follow the documented logic to find a repository. If one cannot be found, the program will start anyway and not give an error; if you check the Repository Location option, the fields will be blank. (8.3)
- If a method catalog is not specified at start-up, and the default method catalog (in DBLDIR) could not be opened, the program will prompt you to select a valid catalog, instead of exiting. (8.1.7a)
- We added a new function, Verify Catalog, which can be used to update the structure sizes in your method catalog after making changes to your repository. (8.1.7)
- We added a new checkbox, Collection, to the Parameter Definition screen to support the collection parameters now available for *xf*NetLink Java and *xf*NetLink .NET clients. (8.1.7)
- We added additional validations and a logfile to the Import Methods function. If you run the import from the command line, you can specify the log with the **-l** option. If you run the import from within the utility, you'll be prompted for the logfile name. Most validations that are normally done by the MDU when you enter data are now also done during import. You can ensure that your SMC definitions are valid by exporting and then immediately importing the entire catalog. (8.1.7)
- You can now create an SMC on import from the command line. If you specify an SMC path on the command line and there are no SMC files in that location, the SMC files will be created from the XML file specified with the **-i** option. (8.1.7)
- The MDU now validates the command line options and presents an error if an invalid option or an invalid combination of options is used. (8.1.7)
- We added a **-?** option to the MDU command line options to display usage information. (8.1.7)

- We made a number of miscellaneous improvements to the MDU user interface and online help. For example, we added a new menu named "Utilities" and moved the Import Methods and Export Methods functions to it. The new Verify Catalog function is also on that menu. (8.1.7)
- We added command line options for exporting (**-e**) and importing (**-i**) Synergy Method Catalog (SMC) definitions. These options enable you to export definitions to an XML file and import definitions from an XML file without displaying the MDU user interface. (8.1)
- We added a new data type (handle) to the Parameter Definition screen. Select handle when you want to pass a non-array parameter greater than 64K in size. See "Handling Variable-Length and Large Data" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual for more information. (8.1)
- In the Import Methods function, the MDU now supports XML filenames longer than 51 characters. (8.1)

For more information on the Method Definition Utility, see the "Defining Your Synergy Methods" chapter of the *Developing Distributed Synergy Applications* manual and the *xf*ServerPlus release notes, **REL_XFPL.TXT**.

Version 7

This section briefly describes new features in *xf*ServerPlus version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*ServerPlus version 7.

- (Windows) The service runtime (**dbs.exe**), which is used by *xf*ServerPlus, now interprets **synergy.ini** when SFWINIPATH is set. (7.5)
- The default port for *xf*ServerPlus has been changed from 2330 to 2356 so that it does not conflict with the default port for *xf*Server. If you were relying on a default port of 2330, you will now need to explicitly specify it. (7.3)

New features

*xf*ServerPlus

- (Windows) We added a new **rsynd** option, **-rs**, which enables you to register and then start **rsynd** in one step (when used with the **-w** and **-u** options). (7.5.1e)
- (Windows) *xf*ServerPlus can now be installed on a machine configured as a license client. Previously it could only be installed on a machine that was a license server. To enable this feature, the SynLM service must be re-registered (via Synergy/DE uninstall-reinstall or by running **synd -x** followed by **synd -r** using the current version of **synd.exe**). A reboot may be necessary. (7.5)
- (OpenVMS) We added a new **rsynd** option, /LOG_LEVEL, which can be used to control the logging level and whether messages are sent to the console. (7.5)
- (Windows, UNIX) When *xf*ServerPlus is started with the XFPL_DBR environment variable set, the log file now includes the line “XFPL_DBR logical is set”. (7.5)
- When a COM client connects to *xf*ServerPlus, it will be properly identified as such in the *xf*ServerPlus log file. (7.5)
- You can now pass arrays of structures that total over 64K (65,535). (7.5)
- We added a new utility, **gensyn**, which will generate test skeletons from the definitions of your Synergy routines in the SMC. This will make it easier to test and debug your modularized Synergy code. (7.5)
- When *xf*ServerPlus is enabled, the *xf*Server data access functionality on that same port is now disabled by default. To enable it, use the new **rsynd -b** or **/DATA_ENABLE** option. (7.5)

- We added support for automatic caching of the SMC records within *xf*ServerPlus. This improves performance by enabling *xf*ServerPlus to return a previously read record instead of accessing the SMC files again. (7.5)
- We made several changes to *xf*ServerPlus which have optimized it to run faster. (7.5)
- We added a new routine to the *xf*ServerPlus API, XFPL_REGCLEANUP. This routine enables you to register a clean-up method with *xf*ServerPlus, which will run when socket communication with the client is unexpectedly lost. (7.5)
- Version 7.3 makes it easier to call Synergy logic from Active Server Pages by implementing COM component creation, structure support, and more robust error handling in *xf*ServerPlus. (7.3)
- (Windows) We added the new Synergy Configuration Program to the Start menu. To run it, select Synergy/DE > Utilities > Synergy Configuration Program. Use this new program to configure *xf*ServerPlus. (7.3)
- We added additional error messages to assist you in troubleshooting. (7.3)
- We added the XFPL_BASECHAN setting in the **xfpl.ini** file to specify a starting point for the range of channels used by *xf*ServerPlus. (7.1)

Method Definition Utility (MDU)

- We added an error message to help prevent name collisions when working in a COM environment. If two routines are named the same (case insensitive) and they are assigned to the same interface, you'll get a message warning you of a potential routine name collision. The message does not prevent you from continuing. (7.5)
- When a record is in use by another user, the MDU will now display a "record locked" error. (7.5)
- When starting the MDU, you can now specify that the repository files reside on a remote machine that is running *xf*Server by specifying the node name in the path. You can do this with the **-r** option, or when using the **-m** and **-t** options. (7.5)
- We removed the tabsets in the user interface and updated the menus accordingly. For example, there is no longer a Search tab; instead you select the Find option from the Functions menu. (7.5)
- We added the ability to delete an interface and all the methods associated with it. (7.5)
- When you create a new set of SMC files using the Main > Catalog Location option, the files will no longer be completely blank. Instead, they'll include the default methods XFPL_LOG, XFPL_REGCLEANUP, and the test methods. (7.5)
- The list of methods now includes the interface name, and it can be sorted by interface name, method ID, routine name, or ELB. (7.5)
- We added online help to the utility. (7.5)

- We added an import/export feature, which enables you to export method definitions as an XML file and then import them into a different catalog. This feature can also be used to import the default methods (XFPL_LOG, XFPL_REGCLEANUP, and the methods required for the test programs) from the **defaultsmc.xml** file, which is included in your distribution. (7.5)
- We added the ability to search for a structure name when entering a structure parameter. (7.5)
- We made many small improvements to the user interface and the general functioning of the utility, such as rearranging some menu entries and making shortcut keys consistent. (7.5)
- (Windows) We added a Browse button to the Catalog Location dialog. (7.5)
- (Windows) The MDU can now be launched from Workbench. (7.3)
- We added support for selecting structures as parameters. (7.3)
- We added MDU reports. (7.1)

12

*xf*NetLink Synergy

Version 9

This section briefly describes new features in *xf*NetLink Synergy version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- We added support for encryption of network packets sent and received between *xf*ServerPlus and *xf*NetLink Synergy. (9.3)
- To support encryption, we added an optional argument to %RXSUBR and RX_SETRMTFNC. For methods requiring slave encryption, "/encrypt" is added to the end of the method ID argument. (9.3)

Version 8

This section briefly describes new features in *xf*NetLink Synergy version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- We changed the way that *xf*NetLink and *xf*ServerPlus use sockets to establish communication. Version 8.3 *xf*NetLink clients will connect to *xf*ServerPlus using the new method; pre-8.3 clients and clients running in debug mode will continue to use the old method. This change was made to prevent problems that occurred when there were multiple firewalls in a WAN. (8.3)
- You will no longer receive an error when calling a Synergy method that has a Collection parameter from an *xf*NetLink Synergy client. This means you can use the same method definitions for *xf*NetLink Synergy and *xf*NetLink Java or .NET. On the *xf*NetLink Synergy side, write your code as though you were passing variable length data. That is, create a memory area on the client and pass the memory handle (as the parameter that is defined as a collection in the MDU) to *xf*ServerPlus using the RCB_xxx routines. For more information, see "Returning a Collection of Structures" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual. (8.1.7a)
- (Windows) A new define, RX_OS_WIN, has been created in **rxapi.def**. It is a duplicate of RX_OS_NT. This value is returned by %RX_RMT_OS and %RX_RMT_SYSINFO. (8.1.5)
- *xf*NetLink Synergy now supports passing parameters larger than 64K when you use the RCB_*xxx* routines. See the next two entries below for more details. See "Handling Variable-Length and Large Data" and "Passing Arrays Larger Than 64K" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual for more information on this feature. (8.1)
- We added a new option, D_TYPE_HVA, to the *type* argument in the RCB_SETARG and RCB_INSARG routines. This option enables you to pass a non-array parameter larger than 64K when using the RCB_*xxx* routines remotely with *xf*NetLink Synergy. (8.1)
- We added a new option, D_TYPE_HND, to the *type* argument in the RCB_SETARG and RCB_INSARG routines. This option enables you to pass an array larger than 64K when using the RCB_*xxx* routines remotely with *xf*NetLink Synergy or when using the RCB_*xxx* routines locally. It can also be used to pass a non-array parameter larger than 64K as a memory handle when using the RCB_*xxx* routines locally. (8.1)
- An implicit external function for RC_API has been added to the Synergy compiler. This means that you no longer need to declare RC_API as an external function when using the Synergy routine call block API. (8.1)

- (64-bit systems) The Synergy routine call block API no longer limits arguments to 65,535 bytes. (8.1)
- As documented, the port argument in the RX_START_REMOTE function now defaults to 2356 instead of 2330. (8.1)

For more information on *xf*NetLink Synergy, see "Part II: xfNetLink Synergy Edition" in the *Developing Distributed Synergy Applications* manual and the Synergy Language release notes, **REL_DBL.TXT**.

Version 7

This section briefly describes new features in *xf*NetLink Synergy version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- We added a new routine, %RX_CONTINUE, which allows a remote routine to continue processing after an %RXSUBR call has timed out. (7.5)
- We added support for a Synergy client. (7.1)

13

*xf*NetLink Java

Version 9

This section briefly describes new features in *xf*NetLink Java version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*NetLink Java Edition version 9.

- If the **genjava -c** option (which determines JRE compatibility) is not specified, the default is now 1.5 instead of 1.2, as it was in 9.5.1a. The value specified with the **-c** option affects not only what new features are available, but also a number of default type mappings. This change may break your code if you regenerate classes for an existing application. (9.5.3)
- We added support for User fields in repository structures passed as parameters where the subtype is Date and the User data field contains either ^CLASS^=YYYYMMDDHHMISS or ^CLASS^=YYYYMMDDHHMISSUUUUUU. Such fields will map to the Java Calendar class when classes are generated with JRE v1.5 compatibility. Previously, all User fields were converted to Strings; consequently, this change may break your code if you regenerate classes for an existing application. (9.5.3)
- We added support for passing Boolean fields within structures when classes are generated with JRE v1.5 compatibility. Previously, Boolean fields were converted to bytes; consequently, this change may break your code if you regenerate classes for an existing application. (9.5.3)
- Binary fields within structures are now mapped to byte arrays when classes are generated with JRE v1.5 compatibility. Previously, binary fields were converted to Strings; consequently this change may break your code if you regenerate classes for an existing application. (9.5.3)
- The following methods have been deprecated: getxfMinport(), getxfMaxport(), setxfMinport(), setxfMaxport(). These methods should be used only when running in debug mode (using the debugInit() and debugStart() methods) through a firewall, requiring a valid range of ports. They cannot be used when making standard, non-debug connections. (9.5.3)
- We added a new option to **genjava**, **-c** *version*, which indicates the Java compatibility of the generated classes. Valid values for *version* are 1.2 and 1.5. Specifying **-c 1.2** means the classes will be generated as they were in *xf*NetLink Java version 9.5.1 and earlier and will be compatible with JRE 1.2 through 1.4. Specifying **-c 1.5** means the classes will be generated so that they are compatible with JRE 1.5. In addition, default type mappings change, ArrayLists use generics, and various new *xf*NetLink Java features released in this patch will be available. The default is **-c 1.2**. This change may break your code if you regenerate classes for an existing application with the **-c 1.5** option. (9.5.1a)

- We have changed the default type mapping for decimal and implied-decimal parameters and return values when **genjava** is run with the **-c 1.5** option. See Appendix B in the *Developing Distributed Synergy Applications* manual for details. This change may break your code if you regenerate classes for an existing application with the **-c 1.5** option. (9.5.1a)
- Date and time fields in repository structures now map to the Calendar class instead of the Date class when **genjava** is run with the **-c 1.5** option. This change may break your code if you regenerate classes for an existing application with the **-c 1.5** option. (9.5.1a)

New features

- *xf*NetLink Java now supports passing enumerations (defined in the repository) as parameters and method return types, as well as enumerations that occur as fields within a structure parameter. (9.5.3)
- We added support for type coercion for return values and parameters for decimal, implied-decimal, and integer data types. This feature enables you to specify a non-default data type (when defining data in the MDU or attributing your code), which will be used as the Java data type in the generated classes. Type coercion is also supported for fields in repository structures. (9.5.3)
- We added support for passing return values of type System.String. (9.5.3)
- We added support for the Synergy System.Collections.ArrayList class as a parameter. (9.5.3)
- We added two utility methods to the generated classes, getPoolName() and getSynergyWebProxy(). The getPoolName() method returns the pool ID that was used in the call to the usePool() method. The getSynergyWebProxy() method returns the internal instance of the SynergyWebProxy. (9.5.1a)
- We added two additional getInstance() methods to the SWPManager class to enable you to use a non-default pooling properties file. One passes the full path and filename of the pooling properties file as a String; the other passes the pooling properties file as a java.util.Properties object. (9.5.1a)
- We added a new option to **genjava**, **-t** *version*, which indicates the version of the JRE that the JAR file should be built to target. Valid values for *version* are 1.5 and 1.6. If **-t** is not specified, the default is the version of Java installed on the machine where the JAR file is built. (9.5.1a)
- We added a new option to **genjava**, **-ro**, which causes fields flagged in Repository as read-only to be generated as read-only properties in structure classes. These properties will have “get” methods but not “set” methods. (9.5.1a)
- *xf*NetLink Java now supports passing parameters of type System.String. (9.5.1a)

- We added support for encryption of network packets sent and received between *xf*ServerPlus and *xf*NetLink Java. See "Using Encryption" in the "Configuring and Running *xf*ServerPlus" chapter of the *Developing Distributed Synergy Applications* manual for details on this feature. (9.5.1a)
- We added a new utility, **genCert.java**, to assist you in setting up your *xf*NetLink Java machine for encryption. (9.5.1a)

Version 8

This section briefly describes new features in *xf*NetLink Java version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*NetLink Java Edition version 8.

- We updated *xf*NetLink Java to use the latest Xerces XML parser. You must change your Java classpath to include **xercesImpl.jar** and **xml-apis.jar** instead of **xerces.jar**. For complete information on setting the classpath, see "Setting the Classpath" in the "Creating Java Class Wrappers" chapter of the *Developing Distributed Synergy Applications* manual. (8.1.7)
- (Windows 98/Me/NT, UNIX, OpenVMS) We changed the name of the default *xf*NetLink Java installation directory from C:\xfNetLnk to C:\xfNLJava. The new default will be used on new installations. If you are upgrading to 8.1 (Windows only), the installation will default to your current *xf*NetLink Java directory and no change to your classpath is required. However, if you choose to use the new default installation directory on an upgrade, you must reset your classpath. (8.1)
- (Windows 2000/XP) We changed the default *xf*NetLink Java installation directory from C:\xfNetLnk to C:\Program Files\Synergex\xfNLJava. The new default will be used on new installations. If you are upgrading to 8.1, the installation will default to your current *xf*NetLink Java directory and no change to your classpath is required. However, if you choose to use the new default installation directory on an upgrade, you must reset your classpath. (8.1)
- The **genxml** program now includes a check to ensure that the sum of the field sizes in a structure match the structure size. This fixes a problem in which arrays of unfilled groups were not getting their field values set properly. This change could break your code if you attempted to compensate for this problem in your client code. (8.1)
- (Windows) The syntax for wrapped installations has changed. If you wrap the *xf*NetLink Java installation, please contact Synergex for an updated copy of the documentation and a description of the changes. (8.1)

New features

- We added support for a method name separate from the routine name. See "Method Definition Utility (MDU)" on page 11-11. (8.3)
- We added support for a parameter name that is different than the structure name. This enables you to have two or more parameters that use the same structure within a method. The class will still be named with the structure name; you will see the parameter name when viewing the method signature in a class browser or when using a feature such as IntelliSense. (8.3)
- You can now include the comments needed for Javadoc when you define routines in the MDU. The method, return value, and parameter descriptions entered in the MDU will be inserted in the generated Java classes as comments. (Previously, you had to edit the generated **.java** files to add comments, and the edits were lost if you had to regenerate the component.) (8.3)
- For structures passed as parameters, the field description in the repository is used as the Javadoc comment for the property methods in the generated class file. The description text from the repository is preceded with either "Sets" or "Gets". (There's a "set" method and a "get" method associated with each property.) If a field does not have a description in the repository, "Sets the property value" and "Gets the property value" are used as the Javadoc comments. (8.3)
- We changed the order in which the Javadoc comments are written for parameters because J2SE 5.0 (JDK 1.5) was giving a warning when the previous order was used. The information is now in this order, following the @param tag: parameter name, Java data type, direction parameter is passed. (8.3)
- We added support for passing parameters with a binary data type, such as JPEG images. On the Java side, the parameter is handled as an ArrayList. On the *xf*ServerPlus side, the data is received as a memory handle. Note that this does not affect how we support binary fields in structures; they are still mapped to strings. See "Passing Binary Data" in the "Calling Synergy Routines from Java" chapter of the *Developing Distributed Synergy Applications* manual. (8.3)
- We added a new class, synByteArray, to the **xfnljav.jar** file. This class is needed to support binary parameter types. (8.3)
- If all ports in the range are in use when the *xf*NetLink Java client attempts to connect to *xf*ServerPlus, the client will now wait 1 second and then retry. After a total of five attempts, the client will report a "no available ports in range" error. (8.3)
- We changed the way that *xf*NetLink and *xf*ServerPlus use sockets to establish communication. Version 8.3 *xf*NetLink clients will connect to *xf*ServerPlus using the new method; pre-8.3 clients and clients running in debug mode will continue to use the old method. This change was made to prevent problems that occurred when there were multiple firewalls in a WAN. (8.3)
- The **genxml** utility now maintains the case of parameter names when it writes the XML file. Previously, parameter names were changed to lowercase, which caused a "duplicate parameter" error in the Method Definition Utility when a routine had two parameters that differed only in case, and an export followed by an import was performed. (8.3)

- The **genxml** utility now supports two consecutive backslashes in a filename, as might appear in a UNC path. (8.3)
- If a Synergy method sends a zero date to *xf*NetLink Java as part of a structure, a Date object with a date of 01/01/0001 is created. (Previously, zero dates were being created using a date in 1970, which could easily be a "real" date.) Your client application will need to test for this date to know that a zero date was sent from Synergy. If *xf*NetLink Java sends a date of 01/01/0001 to a Synergy method, a zero date is created. (8.1.7a)
- If minport and maxport are defined, the *xf*NetLink client now makes the initial connection to *xf*ServerPlus on a port within that range. Previously, the initial connection to *xf*ServerPlus was made on a random port above 1024, which caused difficulties for users accessing *xf*ServerPlus through a firewall. Now, the random port is used only when minport and maxport are not set. (Note that the port number range is still used for the listening port, as it was previously.) Note the following:
  - If you are currently using minport/maxport, you may need to expand the range of ports so that there are enough ports available for both listening and the initial connection.
  - If your *xf*NetLink client is running on Windows, you may need to decrease the TcpTimedWaitDelay setting in the registry to ensure that ports are released promptly. This value is set in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters. See Microsoft KB article 314053 (for XP) or 120642 (for 2000/NT) for more information on this registry setting.

  (8.1.7a)
- We added support for a custom package name to **genjava**. You can specify a package name from the command line with the **-p** option or from the Component Information dialog in Workbench. Maximum length is 101 characters. (8.1.7)
- We added support for collection parameters (ArrayLists). The Java client can now receive a variable length array of structures from *xf*ServerPlus. (8.1.7)
- We optimized structure handling to significantly improve performance when passing arrays of structures. (8.1.5)
- We added support for Java connection pooling. You can now create a pool of connections to *xf*ServerPlus that are ready to use when a client makes a request. For details, see the "xfNetLink Java Edition" section of the *Developing Distributed Synergy Applications* manual. (8.1.3)
- We added a new exception class, xfPoolException, which signals errors that occur at the Java connection pool level. (8.1.3)
- We added a new exception class, xfJCWException, which serves as a wrapper for the other exceptions in **xfnljav.jar** when your code uses Java class wrappers. (8.1.3)

- *xf*NetLink Java now supports passing parameters larger than 64K. The **genjava** and **genxml** utilities were modified to support this enhancement. See "Handling Variable-Length and Large Data" and "Passing Arrays Larger Than 64K" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual for more information. (8.1)
- After you run Build > Generate Class Wrappers in Workbench to create Java class wrappers, the **.java** files are added to the Java Component project. The JAR file is no longer added to the project. This enables you to edit the class files as needed before building the JAR file. (8.1)
- We added a **readme.txt** file to the xfNLJava\Examples directory that describes how to run the sample Java programs. (8.1)

For more information on *xf*NetLink Java, see "Part III: xfNetLink Java Edition" in the *Developing Distributed Synergy Applications* manual and the *xf*NetLink Java release notes, **REL_XFNJ.TXT**.

Version 7

This section briefly describes new features in *xf*NetLink Java version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- *xf*NetLink Java now includes tools that enable you to create Java class wrappers from your SMC and Repository definitions. You can then build the class wrappers into a JAR file, which can be referenced as a Javabean in a JSP page. The class wrapper tools can be accessed from Workbench or run from the command line. (7.5)
- We added support for passing structures (defined in the Repository) as parameters to *xf*NetLink routine calls. (7.5)
- We added support for passing more than 20 parameters. (7.3)
- Classes are now distributed in a **.jar** file. (7.1)
- We added a configurable time-out value. (7.1)

14

*xf*NetLink COM

Version 9

This section briefly describes new features in *xf*NetLink COM version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

New features

- The Synergy Type Library Configuration utility online help is now compiled HTML Help rather than WinHelp. (9.1)

Version 8

This section briefly describes new features in *xf*NetLink COM version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*NetLink COM Edition version 8.

- (Windows 2000/XP) We changed the *xf*NetLink COM default installation directory to C:\Program Files\Synergex\xfNLCOM. The new default will be used on new installations. If you are upgrading, the installation will default to your current *xf*NetLink COM directory. (8.1)
- If you have an existing *xf*NetLink COM installation and change the directory when installing a newer version, you will need to unregister and reregister your Synergy TLBs. See the *Developing Distributed Synergy Applications* manual for instructions on unregistering and registering TLBs. (8.1)
- We no longer distribute **xfnlcom.dll**, the original (version 7.1) implementation for *xf*NetLink COM. This DLL was rendered unnecessary when we implemented the type library generation tools and the object-oriented API in version 7.3. Code written with version 7.1 will not run in version 8.1. (8.1)
- If you are upgrading from 7.3 to 8.1, any Synergy type libraries that you registered with 7.3 will need to be unregistered and re-registered. This is required because **xfrerouter.dll**, which is the implementation for Synergy TLBs, is now installed in the xfNLCOM directory. (Previously it was installed in synergyde\xfutil.) See "Part IV: xfNetLink COM Edition" of the *Developing Distributed Synergy Applications* manual for instructions on unregistering and registering TLBs. (8.1)
- If you are upgrading from 7.3 to 8.1 and want to pool *xf*NetLink COM components, you will need to add an entry for the XFPL_REGCLEANUP routine to your SMC. (A new installation will include this routine in the default SMC.) To do this, use the import methods feature in the MDU to import XFPL_REGCLEANUP from the **defaultsmc.xml** file. For details on importing methods, see the "Defining Your Synergy Methods" chapter of the *Developing Distributed Synergy Applications* manual. (8.1)
- The **genxml** program now includes a check to ensure that the sum of the field sizes in a structure match the structure size. This fixes a problem in which arrays of unfilled groups were not getting their field values set properly. This change could break your code if you attempted to compensate for this problem in your client code. (8.1)
- The syntax for wrapped installations has changed. If you wrap the *xf*NetLink COM installation, please contact Synergex for an updated copy of the documentation and a description of the changes. (8.1)

New features

- We added support for a method name separate from the routine name. See "Method Definition Utility (MDU)" on page 11-11 for details. (8.3)
- We added support for a parameter name that is different than the structure name. This enables you to have two or more parameters that use the same structure within a method. The class will still be named with the structure name; you will see the parameter name when viewing the method signature in a class browser or when using a feature such as IntelliSense. (8.3)
- If all ports in the range are in use when the *xf*NetLink COM client attempts to connect to *xf*ServerPlus, the client will now wait 1 second and then retry. After a total of five attempts, the client will report a "no more available ports" error. (8.3)
- We changed the way that *xf*NetLink and *xf*ServerPlus use sockets to establish communication. Version 8.3 *xf*NetLink clients will connect to *xf*ServerPlus using the new method; pre-8.3 clients and clients running in debug mode will continue to use the old method. This change was made to prevent problems that occurred when there were multiple firewalls in a WAN. (8.3)
- The **genxml** utility now maintains the case of parameter names when it writes the XML file. Previously, parameter names were changed to lowercase, which caused a "duplicate parameter" error in the Method Definition Utility when a routine had two parameters that differed only in case, and an export followed by an import was performed. (8.3)
- The **genxml** utility now supports two consecutive backslashes in a filename, as might appear in a UNC path. (8.3)
- All errors that occur within the *xf*NetLink COM client are now logged to the Windows application event log, even if client-side logging is turned off. (8.1.7d)
- If minport and maxport are defined, the *xf*NetLink client now makes the initial connection to *xf*ServerPlus on a port within that range. Previously, the initial connection to *xf*ServerPlus was made on a random port above 1024, which caused difficulties for users accessing *xf*ServerPlus through a firewall. Now, the random port is used only when minport and maxport are not set. (Note that the port number range is still used for the listening port, as it was previously.) Note the following:
  - If you are currently using minport/maxport, you may need to expand the range of ports so that there are enough ports available for both listening and the initial connection.
  - You may need to decrease the TcpTimedWaitDelay setting in the Windows registry to ensure that ports are released promptly. This value is set in HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Services\Tcpip\Parameters. See Microsoft KB article 314053 (for XP) or 120642 (for 2000/NT) for more information on this registry setting.

  (8.1.7a)

- You will no longer receive an error when generating a type library that includes a method with a collection parameter. (Collection parameters are not supported on *xf*NetLink COM.) Instead, you will get a warning message and **gentlb** will continue, but the method will not be included in the type library. This feature enables you to use the same method definitions with an *xf*NetLink COM client and an *xf*NetLink Java or .NET client. See "Returning a Collection of Structures" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual. (8.1.7a)
- We optimized structure handling to significantly improve performance when passing arrays of structures. (8.1.5)
- *xf*NetLink COM now supports passing parameters larger than 64K. The **gentlb** and **genxml** utilities were modified to support this enhancement. See "Handling Variable-Length and Large Data" and "Passing Arrays Larger Than 64K" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual for more information. (8.1)
- The connect time-out value is now also used for the request for session time-out, which measures how long *xf*NetLink will wait to receive an acknowledgment from the connection monitor in *xf*ServerPlus. (8.1)

For more information on *xf*NetLink COM, see "Part IV: xfNetLink COM Edition" in the *Developing Distributed Synergy Applications* manual and the *xf*NetLink COM release notes, **REL_XFNC.TXT**.

Version 7

This section briefly describes new features in *xf*NetLink COM version 7 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*NetLink COM Edition version 7.

- If you are upgrading from 7.3 to 7.5, any Synergy type libraries that you registered with 7.3 will need to be unregistered and re-registered. This is required because **xfrerouter.dll**, which is the implementation for Synergy TLBs, is now installed in the xfNLCOM directory. (Previously it was installed in synergyde\xfutil.) See "Part IV: xfNetLink COM Edition" of the *Developing Distributed Synergy Applications* manual for instructions on unregistering and registering TLBs. (7.5)
- We have changed the way we handle duplicate interfaces in *xf*NetLink COM. When a type library contains two interfaces with the same name (case insensitive), we append a number, starting with one and incrementing, to the end of each duplicate interface. Previously, we appended underscores. (7.5)

New features

- *xf*NetLink COM now supports European formatting in implied decimals. (7.5.1f)
- **Genxml** now handles overlay fields in groups (i.e., substructures) in the same manner as it does overlay fields in structures. (7.5)
- We changed the XML parser used by *xf*NetLink COM to the Xerces parser. It can handle larger XML files. (7.5)
- The performance for passing large arrays of structures has been dramatically improved. (7.5)
- We made several improvements to the Synergy Type Library Configuration Utility: it now displays interfaces more quickly, especially on a system with a great many type libraries; it can now handle reading over 500 type libraries and displaying all the interfaces; you can now set xfhost to "localhost" rather than having to enter the machine name. (7.5)
- The COM type library generation tools are now installed with *xf*NetLink COM instead of with Professional Series Development Environment. This will make deployment easier. (7.5)
- *xf*NetLink COM is no longer dependent on the Microsoft Java Virtual Machine. As a result of this change, there is no longer a restriction on running both *xf*NetLink COM and *xf*NetLink Java on the same machine. (7.5)

- The **gensxml** utility was re-written in Synergy Language so that it can be run on all Synergy-supported platforms. It is now named **genxml**. This enables you to create a Synergy XML file from code on a UNIX or OpenVMS system; the file can be then copied to a Windows system to make a type library. (7.5)
- **Genxml** has a new **-n** option that enables you to use the value in the Alternate name field (formerly called the ODBC name field) in Repository instead of the value in the Name field when passing structures as parameters. This feature can also be accessed when you generate type libraries from Workbench. (7.5)
- We added support for the Repository "Excluded by web" flag. It enables you to exclude certain fields from your type library when passing structures as parameters. (7.5)
- *xf*NetLink COM components can now be pooled using COM+ pooling on Windows 2000/XP. (7.5)
- Network packet information can now be listed within the *xf*NetLink COM log file by setting an option in the Synergy TLB Configuration utility. (7.5)
- We added support for a connect time-out for both regular and debug connections. You can set this value in the Synergy TLB Configuration utility. (7.5)
- *xf*NetLink COM was wrapped to make it easier to use. Using *xf*NetLink COM and tools in Professional Series Development Environment, you can now create a type library of Synergy methods that can be used by any COM-compliant client. (7.3)
- There is a new utility, the Synergy Type Library Configuration utility, to assist you in configuring a client machine for use with *xf*NetLink COM. (7.3)
- We added an *xf*NetLink COM test program, **xfTestchm**. (7.3)
- We added support for passing more than 20 parameters and for passing structures (defined in the Repository) as parameters. (7.3)
- We added the ability to run an *xf*ServerPlus debug session from a COM client. (7.3)
- We added support for a COM-compliant client. (7.1)

15

*xf*NetLink .NET

Version 9

This section briefly describes new features in *xf*NetLink .NET version 9 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*NetLink .NET Edition version 9.

- We added a new option to **gencs**, **-w**, which causes the assembly to be generated to use WCF contracts. This enables the assembly to be hostable. In addition, the **-w** option changes the way ArrayList and structure collection parameters are handled: they are generated as List<T> parameters (where T is the data type of the elements) instead of ArrayLists. Consequently, in your client code, you must use a List<T> class for a parameter that is defined as an ArrayList or structure collection in the SMC. This change may break your code if you regenerate classes for an existing application with the **-w** option.

  The **gencs -w** option is currently not available from the Component Information dialog in Workbench. However, there is a workaround for Workbench users: From within a .NET Component project, select Project > Project Properties. Go to the Tools tab and select Generate C# Classes from the list of Tools. At the end of the command line, add a space followed by "-w". (9.5.1a)

- Enumeration member names now have an initial cap (e.g., Green) instead of being all caps (e.g., GREEN) in the generated code. (9.3.1a)
- (64-bit) After installing 9.3, you must rerun the batch file (or select Build > Build Assembly in Workbench) to rebuild the assembly, and then rebuild your application in Visual Studio to use the rebuilt assembly. (9.3)
- A field in a structure that is marked as a binary data type in the repository will now be converted to a byte array in the generated C# code. Formerly, it was converted to a string. To continue to use the old behavior, run **gencs** with the **-nb** option. This change may break your code if you regenerate classes for an existing application. (9.1.5b.)

New features

- We added a new option to **gencs**, **-p**, which causes generated structure classes to include the INotifyPropertyChanged class. Because this feature applies only when generating properties, it is not valid when used in combination with the **-g** option (which generates structure members as public fields). (9.5.3)
- *xf*NetLink .NET now supports User fields in repository structures passed as parameters, where the subtype is Date and the User data field contains ^CLASS^=YYYYMMDDHHMISSUUUUUU. Such fields map to a DateTime data type in the generated classes. (9.5.3)

- The format YYYYMMDDHHMISSUUUUUU is now supported for both parameters and return values when coercing a decimal data type to a DateTime. You can specify this format when defining data in the MDU or when attributing your code. (9.5.3)
- We added support for coercing decimal data types to decimals and nullable decimals for parameters and return values. (Coercing decimal to decimal is not as redundant as it might seem, because by default, decimals are converted to int or long depending on size.) (9.5.3)
- In the *xf*NetLink .NET Configuration utility, we removed the minport and maxport options from the Key drop-down list in the Add Class Setting dialog because these settings have been deprecated. If you open an existing config file that contains these settings, when you attempt to modify them a message will display, telling you to remove them. (9.5.3)
- The Synergy .NET Configuration utility has been renamed the *xf*NetLink .NET Configuration utility to avoid confusion with the support for .NET in Synergy Language. You'll see the new name on the Start menu, in the application and help windows, and on the Synergy/DE > Utilities menu in Workbench. (9.5.1)
- We added a new option to gencs, **-nr**, which causes output parameters to be generated as "out" rather than "ref" in the C# code. This option is also available in the Component Information dialog in Workbench. (9.5)
- We now support Visual Studio 2010 and Framework version 4.0. (9.3.1b)
- The "Original" property (new in 9.3.1) is now generated only when you use the **gencs -r** option. (9.3.1b)
- The version number of the **xfnlnet.dll** and **gencs.exe** files distributed with 32-bit and 64-bit *xf*NetLink .NET are now the same. This will make it easier to develop in one environment and deploy in another. (9.3)
- We added support for encryption of network packets sent and received between *xf*ServerPlus and *xf*NetLink .NET. (9.3)
- *xf*NetLink .NET now supports passing enumerations (defined in the repository) as parameters and method return types, as well as enumerations that occur as fields within a structure parameter. (9.3)
- We added support for a Boolean data type within a structure. (9.3)
- We added support for a nullable decimal coerced type for implied-decimal and decimal data types in repository structure fields. We added support for a decimal coerced type for decimal data types in repository structure fields. (If you don't specify decimal as the coerced type, decimal are converted to int or long, depending on size.) (9.3)
- We added support for an "Original" property within structure classes. This property will hold a duplicate instance of the class, which can be compared to the class in its current state to see if anything has changed. (9.3)
- The TableName property in a DataTable class is now set to just the structure name. Previously, it was set to SynDataTable plus the structure name. This change facilitates connecting to a data source. (9.1.5a)

- The columns in a DataTable are now initialized when the table is instantiated. Previously, the column names were initialized when the call was made. This change enables you to see the column names in Visual Studio at design time. (9.1.5a)
- We no longer overwrite the **AssemblyInfo.cs** file when classes are regenerated. (9.1.3)
- *xf*NetLink .NET now supports Visual Studio 2008. The minimum supported version is 2005. (9.1.3)
- *xf*NetLink .NET version 9.1.3 is built with version 2.0 of the .NET Framework, which is now the minimum supported Framework version. (9.1.3)
- The Repository read-only flag is now honored when structure members are generated as properties. In the C# class, the property will have a "get" method but no "set" method. (9.1.3)
- We added support for type coercion for return values and parameters for decimal, implied decimal, and integer data types. This feature enables you to select a non-default data type (when defining data in the MDU), which will be used as the C# data type in the generated classes. Type coercion is also supported for fields in repository structures. (9.1.3)
- We added a new Clone() method to the structure classes in the generated assembly. Clone() returns an exact copy of the structure class on which it is called. (9.1.3)
- We added a new Equals() method to the structure classes in the generated assembly. Equals() tests whether data in two structure classes is the same and returns true or false. (9.1.3)
- We added a Changed property to the structure classes in the generated assembly. The Changed property is included only when structure members are generated as properties. It returns a Boolean value of true if the value of any property's data field in the structure class has been changed by calling the "set" method. If no values have changed, it returns false. (9.1.3)
- We added support for the Synergy System.String data type as both a parameter and a return value. (9.1.3)
- We added support for the Synergy System.Collections.ArrayList class as a parameter. (9.1.3)
- We added support for DataTables. You can define a structure parameter that is an ArrayList or structure collection as a DataTable in the MDU; it will be created as a DataTable in the generated assembly. (9.1.3)
- We added a new environment variable, SYNCSCOPT, which can be set to any valid C# compiler option that you want added to the command line when the assembly is built. SYNCSCOPT can be used when building from the command line or from Workbench. (9.1.3)
- If protocol logging is enabled, the log now includes a date/time stamp for each packet sent and received. (9.1.3)

Version 8

This section briefly describes new features in *xf*NetLink .NET version 8 and the changes in this version that may break your code. Items are listed with the most recent first, and the number at the end of each item indicates the revision in which the feature was implemented.

Features and fixes that may break your existing code

The following changes may break code when you update to *xf*NetLink .NET Edition version 8.

- If your Synergy assembly includes methods that pass structures as parameters, you must add a reference to **xfnlnet.dll** in your Visual Studio .NET project. (8.3)
- If you wrap the *xf*NetLink .NET installation, please contact Synergex for a current copy of the documentation. (8.1)
- If you have created an application using a Synergy assembly built with version 7.5.3, you must rerun the batch file (created by **gencs**) to rebuild the assembly, and then rebuild your application in Visual Studio to use the rebuilt assembly. (8.1)

New features

- When a call time-out occurs, the exception error message now includes the exception type, method name, and the call time-out value in seconds, e.g., System.IO.IOException processing method “myMethod” Socket timeout error: 10060, Call timeout = 12 seconds. (8.3.1d)
- We added the ability to generate C# classes and build a Synergy .NET assembly from within Workbench using a .NET Component project. See “Creating an Assembly in Workbench” in the “Creating Synergy .NET Assemblies” chapter of the *Developing Distributed Synergy Applications* manual. (8.3)
- We added support for a method name separate from the routine name. See “Method Definition Utility (MDU)” on page 11-11 for details. (8.3)
- We added support for a parameter name that is different than the structure name. This enables you to have two or more parameters that use the same structure within a method. The class will still be named with the structure name; you will see the parameter name when viewing the method signature in a class browser or when using a feature such as IntelliSense. (8.3)
- You can now include the comments needed for API documentation when you define routines in the MDU. The method, return value, and parameter descriptions entered in the MDU will be inserted in the generated C# classes as comments. (Previously, you had to edit the generated C# files to add comments, and the edits were lost if you had to regenerate the component.) (8.3)
- For structures passed as parameters, the field description in the repository is included as the XML documentation comment for the property (or field) in the generated class file. If a field does not have a description in the repository, the comment “***Field To Do***” is inserted in the generated file. (8.3)

- We added support for passing parameters with a binary data type, such as JPEG images. On the .NET side, the parameter is handled as a byte array. On the *xf*ServerPlus side, the data is received as a memory handle. Note that this does not affect how we support binary fields in structures; they are still mapped to strings. See "Passing Binary Data" in the "Calling Synergy Routines from .NET" chapter of the *Developing Distributed Synergy Applications* manual. (8.3)
- We added several new options to the **gencs** utility: **-g**, **-i**, and **-x**. The **-g** option enables you to generate structure members as public fields rather than as properties with private fields, which is still the default. The **-i** option enables you to create multiple copies of the same class from a specified interface name. The **-x** option enables you to specify the version of the .NET Framework that you want to generate the assembly for. If you have only one version of the .NET Framework installed on your machine, you do not need to use this option. If you have both versions 1.1 and 2 installed, the default version is 1.1. Use the **-x** option (-x 2) to specify that you want to build for version 2 instead. For more information on these new options, see "The gencs Utility" in the "Creating Synergy .NET Assemblies" chapter of the *Developing Distributed Synergy Applications* manual. (8.3)
- If a method definition has a parameter marked as optional, **gencs** will now give a warning that it is converting the optional parameter to a required parameter. (8.3)
- The **genxml** utility now maintains the case of parameter names when it writes the XML file. Previously, parameter names were changed to lowercase, which caused a "duplicate parameter" error in the Method Definition Utility when a routine had two parameters that differed only in case, and an export followed by an import was performed. (8.3)
- The **genxml** utility now supports two consecutive backslashes in a filename, as might appear in a UNC path. (8.3)
- *xf*NetLink .NET now generates a C# interface for each generated class. This enables you to more easily call methods on a dynamically instantiated object. See "Understanding the Generated Classes" in the "Creating Synergy .NET Assemblies" chapter of the *Developing Distributed Synergy Applications* manual. (8.3)
- By default, client-side logging will now use multiple log files, one for each class that instantiates a connection to *xf*ServerPlus. Previously, all connections used the same log file. To revert to a single log file, use the Synergy .NET Configuration Utility to set "single log file" to "on" in the application configuration file. See "Creating and Editing Configuration Files" in the "Calling Synergy Routines from .NET" chapter of the *Developing Distributed Synergy Applications* manual. (8.3)
- We now support event logging for non-privileged users, and errors that occur when writing to the event log are ignored. (8.3)
- We now support the installation of *xf*NetLink .NET on a system that has Framework version 2 installed. (8.3)

- If you use your own key file and you are using version 2 or higher of the .NET Framework, the information regarding the key file and delayed signing will be written to the batch file rather than the **AssemblyInfo.cs** file when you generate C# classes. This is because Microsoft has changed the way this feature is implemented in the upcoming Framework version 2. See "Editing Information in AssemblyInfo.cs" in the "Creating Synergy .NET Assemblies" chapter of the *Developing Distributed Synergy Applications* manual. (8.3)
- We changed the way that *xf*NetLink and *xf*ServerPlus use sockets to establish communication. Version 8.3 *xf*NetLink clients will connect to *xf*ServerPlus using the new method; pre-8.3 clients and clients running in debug mode will continue to use the old method. This change was made to prevent problems that occurred when there were multiple firewalls in a WAN. (8.3)
- If all ports in the range are in use when the *xf*NetLink .NET client attempts to connect to *xf*ServerPlus, the client will now wait 1 second and then retry. After a total of five attempts, the client will report a "no available ports in range" error. (8.3)
- We improved performance when calling methods that pass structures or arrays of structures. (8.3)
- The Synergy .NET Configuration utility can be launched from within Workbench. Go to Synergy/DE > Utilities and select "Synergy .NET Configuration". (8.3)
- The Synergy .NET Configuration utility has been updated to support the new "single log file" key/value pair and the 50-character interface name. (8.3)
- The recently used files list in the Synergy .NET Configuration utility now holds and displays up to 10 files and can store the recently used files from multiple users. (8.3)
- Arrays of strings are now initialized to null. *xf*NetLink .NET will now check to see if all elements are initialized before trying to use them. If an element is equal to null, a blank string is assigned. (8.1.7e)
- Date strings that are zeroes or spaces are now handled as a zero date, which means they are assigned the default date (01/01/0001). (8.1.7e)
- All errors that occur within the *xf*NetLink .NET client are now logged to the Windows application event log, even if client-side logging is turned off. (8.1.7d)
- Event logging is now supported for non-privileged users. (8.1.7e)
- We added a linger setting to the TCP client instances to allow time to send a close-down message before closing the socket. (8.1.7d)
- When using pooling, if there is a socket error (which terminates the connection) before the Deactivate() or Cleanup() methods are called, they will not be called. (8.1.7d)
- If a Synergy method sends a zero DateTime to *xf*NetLink .NET as part of a structure, a .NET default date of 01/01/0001 is created. (.NET has no concept of a zero date.) Your client application will need to test for this date to know that a zero date was sent from Synergy. If *xf*NetLink .NET sends a date of 01/01/0001 to a Synergy method, a zero date is created. (8.1.7a)

- We added two new options to **gencs** (**-s** and **-t**) to aid in signing assemblies. The **-s** option enables you to specify a key file name, and **-t** enables you to indicate that you want to use delayed signing. You must create the key file using Microsoft's **sn.exe** utility and place it in the desired location before running **gencs**. If **-s** is not passed, *xf*NetLink .NET will generate public and private keys in a strong name key file named with the assembly name. See "The gencs Utility" and "Using Your Own Key File" in the "Creating Synergy .NET Assemblies" chapter of the *Developing Distributed Synergy Applications* manual. (8.1.7a)
- If minport and maxport are defined, the *xf*NetLink client now makes the initial connection to *xf*ServerPlus on a port within that range. Previously, the initial connection to *xf*ServerPlus was made on a random port above 1024, which caused difficulties for users accessing *xf*ServerPlus through a firewall. Now, the random port is used only when minport and maxport are not set. (Note that the port number range is still used for the listening port, as it was previously.) Note the following:
  - You must regenerate your assembly to take advantage of this new feature.
  - If you are currently using minport/maxport, you may need to expand the range of ports so that there are enough ports available for both listening and the initial connection.
  - You may need to decrease the TcpTimedWaitDelay setting in the Windows registry to ensure that ports are released promptly. This value is set in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters. See Microsoft KB article 314053 (for XP) or 120642 (for 2000/NT) for more information on this registry setting.

  (8.1.7a)
- We added two new connect() methods to the generated .NET assembly, which enable you to pass at runtime either the port number and host name, or the port number, host name, min port, and max port. (8.1.7)
- We added support for collection parameters (ArrayLists). The .NET client can now receive a variable length array of structures from *xf*ServerPlus. (8.1.7)
- We changed the fields in a structure to be genuine .NET properties so that they can be bound to .NET controls. (8.1.7)
- We enhanced the messages reported by the *xf*NetLink .NET test program, **xfTestnet.exe**. Now, the success message includes the version of *xf*ServerPlus being used. If there is a failure, the message simply reports the reason for the failure rather than giving a traceback. (8.1.5d)
- *xf*NetLink .NET now supports .NET Framework version 1.2. (8.1.5b)
- We optimized structure handling to significantly improve performance when passing arrays of structures. (8.1.5)
- We added the assembly attribute ApplicationAccessControl, with a value of false, to the generated **AssemblyInfo.cs** file. This is a new attribute added by Microsoft. Without it, if you were using pooling, you would get a permissions error when attempting to start the pool. The error could be eliminated by clearing the 'Enforce component level access checks' option. With the new attribute, you no longer have to clear this option before starting the pool. (8.1.5)

- We now include version 1.1 of the .NET Framework redistributable on the Synergy/DE software CD. (8.1.3)
- *xf*NetLink .NET now supports the .NET Framework version 1.1 C# compiler. (8.1.3)
- *xf*NetLink .NET is now supported on all terminal server machines except NT 4 Terminal Services. (8.1.3)
- *xf*NetLink .NET now supports passing parameters larger than 64K. The **gencs** and **genxml** utilities were modified to support this enhancement. See "Handling Variable-Length and Large Data" and "Passing Arrays Larger Than 64K" in the "Preparing Your Synergy Server Code" chapter of the *Developing Distributed Synergy Applications* manual for more information. (8.1)
- The batch file that builds the assembly now supports spaces in the pathname of the XML file. (8.1)
- The connect time-out value is now also used for the request for session time-out, which measures how long *xf*NetLink will wait to receive an acknowledgment from the connection monitor in *xf*ServerPlus. (8.1)
- When you open an existing .NET config file that does not have an xfnlnet section, you will be prompted to add one, and now the default interface will be created also. (8.1)
- When adding a value for host name, you can now include the following characters: underline (_), dash (-), period (.). This enables you to use an IP address for the host name. (8.1)

For more information on *xf*NetLink .NET, see "Part V: xfNetLink .NET Edition" in the *Developing Distributed Synergy Applications* manual and the *xf*NetLink .NET release notes, **REL_XFNN.TXT**.

Version 7

We added support for a .NET client. See "Part V: xfNetLink .NET Edition" in the *Developing Distributed Synergy Applications* manual for information on *xf*NetLink .NET. (7.5.3)