Transitioning from Spread 3.17.x to Spread 4.0

Version 4.0 of The Spread Toolkit introduces some notable changes over versions 3.17.x. These include:

- Dynamic Configuration
- Configuration Equality Enforcement
- Flush Client API
- Increased Membership Information
- New Membership Information Access Functions
- Support for Unix Process Forking
- New Default and Spread-Core Client Libraries
- StdUtil Data Structures
- New Distribution Directory Layout
- New Win32 Build Files

Dynamic Configuration

Spread prior to version 4 required a configuration file specifying all of the daemons who would **ever** be part of the spread configuration and to make changes to the set of potential daemons required that all of the daemons be shut down and restarted once the configuration file was edited to include the changes. This made adding servers to a pool or changing the name or IP of an existing server require a full restart of the system which can be disruptive.

Version 4 of Spread can now handle changes to the spread.conf daemon configuration without shutting down and restarting all of the daemons. Daemons can be added to segments or removed without existing clients losing their spread connections. Some changes to the spread configuration (such as changing the name of an existing daemon) will cause the clients to receive a view change message showing a temporary change in the membership. See the DynamicConfiguration.txt document in the /docs directory for more details.

Configuration Equality Enforcement

The Spread system has always had a requirement that the spread.conf file, specifying the daemon configuration, specified identical configurations at all nodes. However earlier versions of Spread did not enforce this requirement automatically. Now Spread enforces that all participating daemons are using an identical configuration. See the DynamicConfiguration.txt document for more details.

Flush Client API

Flush Spread was available as an add-on for Spread 3.17.x. In Spread 4, Flush Spread is now included with the main distribution. The classic Spread API has calls that start with SP_, whereas the Flush Spread interface has API calls that begin with FL_. Both the Flush and classic Spread interfaces are exported in the libspread client library.

A user should use either FL or SP calls on any given Spread client connection. Most users should continue to use the classic SP interface. If Flush is used however, two rules should be followed:

- **do not mix or use both FL and SP calls on the same Spread client connection.**
- **do not mix FL and SP clients in the same Spread group.**

Flush Spread is an extension to Spread. Flush Spread and Spread are extremely similar group communication systems in the services that they provide, and in their general interface. The main difference between Flush and Spread interfaces is how they handle view/membership changes. The Spread interface presents a group communication system which conforms to the *Extended Virtual Synchrony* (EVS) semantics whereas; the Flush interface presents a group communications system which conforms to the *Virtual Synchrony* (VS) semantics. In an EVS system, membership changes occur without a client's intervention, whereas in a VS system a client must give its permission before a new view/membership can be installed. For more information on the Flush interface, see the *Flush_or_SpreadAPI.txt* file and the flush man pages included in the docs directory.

Increased Membership Information

In Spread 4, membership messages that occur as a result of a network partition or merge now contain more information about the membership event then was presented by version 3.17.x. Prior to Spread 4, a message with a CAUSED_BY_NETWORK service_type contained a set of the private group names of processes which came with the receiver from the old into the new membership. In Spread 4 however, these membership messages contain a *list of sets* of private group names. Each of these sets contains the list of processes from one of the merging network segments that was previously partitioned. This provides a more complete picture of the network event.

Consider an example of 3 network segments A, B, and C merging together. Each segment contains 2 members 1 & 2 (Ex A1, A2 and B1 , B2) that are in group X. The new membership of the group X is {A1,A2,B1,B2,C1,C2} and the VS sets received in the membership message at each daemon are specified in the table below.

Group X Membership Message Received By:	Spread 3.17.x membership information	Spread 4 membership information
A1	{A1, A2}	{A1, A2}, {B1, B2}, {C1, C2}
B1	{B1, B2}	{A1, A2}, {B1, B2}, {C1, C2}
C1	{C1, C2}	{A1, A2}, {B1, B2}, {C1, C2}

A membership message received by A1 in Spread 3.17.x would only list which members came with A1 into the new membership view. Now in Spread 4, A1 receives that list, as well as the lists of members that came from segment B, and a list of members that came from segment C.

New Membership Information Access Functions
With the increase of the information provided by the membership messages, accessing this information needed a new API. In Spread 3.17.x a developer used the SP_get_gid_offset_memb_mess, SP_get_num_vs_offset_memb_mess, and SP_get_vs_set_offset_memb_mess functions to get the offsets into the message body of where this information resided.

In Spread 4, the message body layout is documented, and the user is free to parse the information themselves. Alternatively they can use the new Membership Information Access Functions below:

- A call to SP_get_memb_info will extract the membership information from the message body into a membership_info structure.
- A call to SP_get_vs_sets_info will extract the multiple vs_sets in a membership body
- A call to SP_get_vs_set_members will copy out the list of private group names from a specific set.

For the specifics on these functions, please see the documentation and man pages.

Support for Unix Process Forking
The client libraries of Spread in versions 3.17.x and 4 are thread-safe. However prior to version 4, it was not safe to use the Spread connection in a child process that was forked from a multithreaded parent process who set up the connection. In Spread 4 however, this is now allowed, so long as only one process is accessing the connection at a time. It is the applications responsibility to ensure this.

The new SP_kill function aides in support of forking by allowing the parent process to close its copy of the Spread connection without signaling the daemon and without shutting down the child process's connection. See the MultiThreadedClients.txt and SP_kill documentation for more information.

New Default and Spread-Core Client Libraries

Spread 3.17.x came with 2 client libraries:

- libspread: Non thread safe client library
- libtspread: Thread safe client library

In Spread 4, there are 3 client libraries distributed with the binary release:

- libspread: Default client library. Exports thread safe SP and FL API calls and includes STDUtil data structures.
- libspread-core: Exports non thread safe SP calls only, and does not include STDUtil data structures.
- libtspread-core: Exports thread safe SP calls only, and does not include STDUtil data structures

For most users, it is sufficient to link with the *libspread* library. It is thread safe and contains both the classic Spread API as well as the new Flush Spread API.

If however the user is using a previous version of the StdUtil library, and are worried about name-space collisions, they can link with either the *libspread-core* or *libtspread-core* libraries depending on their need for thread safety. Be advised however, that the Flush Spread Interface depends on StdUtil and is not included in the *libspread-core* libraries.

StdUtil Data Structures

The data structures used internally by the Spread daemon have been replaced with the data structures from the StdUtil library. StdUtil is an open source cross-platform set of high performance data structures in C.

The StdUtil data structures are already linked into the *libspread* default client library. The user does not, and should not, link with the StdUtil library directly. If you need to link with it directly, use one of the *libspread-core* libraries to avoid namespace collisions.

New Distribution Directory Layout

The directory layout of the Spread distribution has changes from version 3.17.x. In version 4, the layout is the same for both the source and binary distributions. The layout is as follows:

Directory	Description
bin	Compiled executables
buildtools	Tools needed during build process
daemon	Daemon source code
docs	Documentation
examples	Sample program sources
include	Headers to include when linking to client library
java	Java client library code
lib	Compiled libraries
libspread	Client library source
perl	Perl client library code
stdutil	StdUtil data structures code
win32	Microsoft Windows project files

The bin and lib directories will contain platform specific subdirectories (such as win32, Linux, Mac, etc) in the binary distribution.

New Win32 Build Files

The included build files for Windows are only for when you are building Spread from source. They are not needed if you are using the binary distribution.

In Spread 3.17.x, the build files for Spread source on Microsoft Windows were project files for Microsoft's Visual Studio 5.0. The Spread 4 release updates these project files to Microsoft's Visual Studio 2002 .Net. To build on Windows, open up the spread.sln in Visual Studio .Net 2002 or later, and compile the entire solution.

I you have a previous version of Visual Studio, or do not use Visual Studio to build, see the Win32BuildInstructions.pdf document to understand the build requirements.