InfiniLinks Version 1.4.0

Release Notes v1.0

Dagran Healthcare Systems Consulting

This is release version 1.4 of the InfiniLinks software, the last major enhancement to the engine before version 2.0 is released. This release showcases a couple of exciting new features and includes a number of bug fixes. As an added bonus, the request/response mechanisms within each communications client have been rewritten to significantly reduce processor utilization when using TCP/IP-based communications clients.

Installation Notes

Similar to previous versions of the program, the Windows software comes in two InstallShield installation packages, one for the server software and one for the monitor and corresponding configuration tools. Version 1.4 can be installed over previous versions of the software; however, it is not recommended that this release be installed over the preview release of version 1.99. UNIX platform releases are sent as two tar archive files, one including updated binaries, and a second containing updated configuration files.

The following files have been made obsolete in this release and should be manually removed from the *bin* directory when performing an upgrade:

File Name	Description
dhctapi.dll	This file, which contained DHSC's lengthy wrapper around the Windows telephony architecture, is no longer needed. The TAPI wrapper has been moved into *dhccdl.dll*.

Two new, mandatory entries have been added to the [Server] section of the server.ini file. The Windows installation program will automatically add these entries to your existing server.ini. These entries are:

File Name	Description
MaxFileSize	Indicates the maximum size, in MB, that the server's log file can grow to before it is automatically archived.
Overwrite	This is entry is a toggle that indicates whether the server's log file should be overwritten or appended to when the server is started.

Consult the enhancements section for more information regarding these new application settings. These configuration settings will need to be manually added to the server.ini for non-Windows installations. Other changes to existing installations include adding new error messages to the server.ini file and correcting typographical errors in existing error messages.

For UNIX platforms, extract the contents of the binary tar file over the existing files in your INFINILINKS_ROOT/bin directory. Extract the contents of the config tar file into your INFINILINKS_ROOT/config directory. This will overwrite all of your standard initialization files except your server.ini config file. A new file will also be inserted into your config directory called strings.ini Open the strings.ini file and copy and paste the contents of this file over the strings section of the server.ini file. This will update the error messages within the server.ini file.

This is the last time UNIX users will need to manually update the strings section of the server.ini. Changes to the algorithm for handling error messages will eliminate the need to perform these steps after version 2.0 is released.

UNIX users will also need to manually add the `Overwrite` and `MaxFileSize` settings to the server.ini file.

A final warning for users who may have made changes to their interp.ini config file: this process will overwrite any changes you have made to your interp.ini file. It is recommended that you manually add any changes back in to the [ParseValues] section of interp.ini.

Enhancements

Two oft-requested enhancements have been added to this release of the software. The first option allows end-users to determine whether the dhc.log should be overwritten at startup (current behavior) or appended to without affecting the current contents of the log. Additionally, there is also a new setting which allows the user to control the maximum size that the dhc.log file can grow to before the contents of the file are archived. Once the log file is archived, the existing dhc.log file is overwritten with new content.

Server Module

This behavior is controlled via new entries in the `[Sever]` section of the server.ini file. If you are unsure where your server.ini file is located, check the value of the $DHC_SERVER_PATH environment variable (all platforms). The value should contain the full path to your existing master configuration file.

Control of whether the server overwrites or appends to the log file is done via the `Overwrite` entry. This entry has two possible values, 0, which indicates that the server should append to the log file, and 1, which tells the server to overwrite the log file at startup. The default value of this entry is 0. If you plan to fully utilize the new archiving features of the engine, we recommend keeping the value of this entry at 0.

The second flag added to the `[Server]` section is the `MaxFileSize` parameter. This parameter tells the server what the maximum size, in Mb, that the log file may grow to before it is archived. Possible values for the setting are a numeric value between 1 and 1024. A value of 1 indicates that the server may grow to 1 Mb before being archived, while a value of 1024 (the maximum size of the log file) instructs the server to archive the log file when it hits 1 GB.

Once the log file exceeds its maximum size, the server renames the file to be <logfilename.0>. If this file already exists, it then tries to rename the file to <logfilename.1>. It will continue to try and rename the file up to <logfilename.100>. If all of these files exist, the server will then shut itself down.

Both of these entries are required values for the server. The server will not start without these entries being properly configured. Listed below is an example of an optimally configured `[Server]` section:

```
[Server]
ServerPort=8050
DebugPort=9050
ErrorLog=c:\dhsc2\infinilinks\Log\error.log
LogFile=c:\dhsc2\infinilinks\Log\dhc.log
MaxFileSize=1
```

```
Overwrite=1
```

Communications Module

The second major enhancement to the engine applies to communications client. Another oft-requested feature was for each communications client to maintain log files independently of the server and other clients. Now, by adding a new `[Log]` section to the client's initialization file, you may now write CDL log messages to the communications client's log file, as well as have the communications client write out informational messages about its current status.

Below is an example of a `[Log]` section of a communications client's log file:

```
[Log]
FileName=/home/Andrew/test/log/out1.log
MaxSize=5000000
Active=on
Debug=on
```

The `FileName` entry is the path and file name for the log file. The `MaxSize` parameter indicates, in bytes, the maximum size that the file can grow to before it is automatically overwritten. The `Active` flag indicates whether or not the log file is turned on, i.e. a value of 'off' means no data will be written to the log file. This flag is used in cases where the developer would like to stop writing to the log file without needing to recompile the CDL script file. A value of 'on' means all requests to write to the log file will be honored. The `Debug` setting is a toggle that indicates whether communications client debug statements (hard-coded into the application), should be written to the log file in addition to user-defined log messages. These debug statements will assist in diagnosing problems with communications clients that may occur in the future.

To write a message to the log file, you use the CDL builtin LOGMESSAGE. In past editions of the software, the LOGMESSAGE builtin took two arguments, the first argument being the string to write to the server's log file, and the second argument was ignored. Now, changing the second argument to "1" will cause the server to write the message to the communications client's log file instead of the server's log file. A value of "0" or a NULL value will have the client write the message to the server's log file. There is no need to recompile existing CDX files after installing the new binaries.

Bug Fixes

The following bug fixes have been made to the CDL module of the InfiniLinks server:

- **(00001015) DISCONNECT built-in occasionally shuts down comm. client –** This problem affects TCP/IP communications clients acting as a TCP/IP server. Using the command of 'exec DISCONNECT "0"' should cause the communications client to disconnect itself from the endpoint and reset itself for a new communications session. Occasionally, this command would cause the client to shut down completely. The problem occurred because the client would attempt to reopen the port before it had been completely closed by the operating system. The server now checks for that particular status code and continually retries the connection on the port instead of shutting down.
- **(00001014) TCP/IP-based clients utilize excessive CPU resources and process messages slowly –** This problem occurred when two threads of execution poll the server for data and enter into a race condition. Each communications client uses two threads, one thread to service requests sent from

the server, and a second thread to execute protocol events and CDL functions. A problem occurred where the request thread would poll the server socket and find a response for the CDL thread. Instead of handling the situation gracefully, the request thread would lock the socket for an inordinate period of time, preventing the CDL socket from retrieving the server's response. The implementation has been changed to eliminate this race condition.

- **(00001053) (Linux Only) –** A problem cropped up for communications clients under UNIX using TCP/IP. A client connection would occasionally not recognize that the other side of the connection had disconnected and would not attempt to reconnect. This occurred when unread data exists on the TCP/IP port and prevents the client from "seeing" the disconnect. We have rewritten the data buffering mechanisms within the TCP/IP protocol object so that the client will behave identically regardless of the type of operating system that the client is run on. This fix will also increase overall throughput and reduce CPU utilization.

The following bug fixes have been made to the server module:

- **(00001016 / 00001017) Corrupted Queues –** Under currently-unknown circumstances (possibly a result of the flush bug below), a queue header record became corrupted, preventing communications clients from pulling data off the queue, but allowing new messages to be inserted into the same block within the queue. This created a situation where the queue counter would go up, but each new message would overwrite the previous message. We have added a number of checksums and fail-safes to determine when the header record becomes corrupted and to shut down the queue to prevent data loss.
- **(00001018) Bug in Flush Command –** There was a bug in the queue flush command that could potentially lead to queue corruption. When the flush command was executed, the server would clear the queue's header record, but would not free the physical data stored within each queue block. This causes the queue to corrupt the next time it tries to write to that data block, which may not occur for quite a while. The problem has been fixed.
- **(00001062) Duplicate client names –** At startup, the server would not detect when two or more communications clients shared the same name. Fix now throws an error message to the dhc.log at startup and prevents the server from starting.

The following bug fixes have been made to the interpreter module:

- **(00001058) Synchronization Code Bug –** For HL7 messages, if the last field in a message caused an error and triggers synchronization code to execute, the interpreter module failed to recognize the end of segment delimiter and treated the next segment as fields within the preceding segment. A fix was applied to the synchronization code, eliminating the error.

The following bug fixes have been made to the Monitor:

- **Monitor Profiles Removed From Server Explorer –** If you are running the Monitor and are connected to two or more servers simultaneously, shutting down or disconnecting one profile may occasionally cause all server nodes to disappear from the tree view, even though the profile is still connected. The issue has been fixed

The following bug fixes have been made to the MML translation module:

- **(00001042) MDLMake Compiler –** This is not necessarily a bug per se; however, if a user forgot to append a % sign to a token, the compiler would treat the following statement as a literal value:

```
overwrite   “ADT_MSHSEGMENT.SENDING_APPLICATION”
to “%ADT_MSHSEGMENT.SENDING_APPLICATION”
```

Instead of writing the value of the sending application field to the outbound message, the module would write the literal string ADT_MSHSEGMENT.SENDING_APPLICATION to the outbound message. We have modified the compiler to compare all literal strings against the MDL message definitions and throw an error if the string is an exact match against a field definition.

Documentation

No modifications to the help files have been made at this time. A new release of the InfiniLinks Users Guide will be sent out in mid-November.