InfiniLinks Version 2.20

Release Notes

Dagran Healthcare Systems Consulting

Thank you for using InfiniLinks.. Version 2.20 has no deployment restrictions. This release features enhanced support for encryption, as well as support for FTP within communications components.

This release also addresses a number of problems, including server and component lockup issues. Please read this document thoroughly prior to performing an upgrade. Familiarizing oneself with these changes will help to avoid many common pitfalls during the upgrade process.

Installation Notes

There are two options for installing InfiniLinks 2.20 under Windows. The CD includes a full installation package and a patch installation. The patch installation is the *InfiniLinks Maintenance.exe* and may be used to perform an express upgrade for installations running 2.1 and higher. The patch will auto-detect the location of the InfiniLinks installation and seamlessly update the current installation to 2.20. No changes to any configuration files are required when upgrading from 2.10 and higher to 2.2.

Use the *Infinilinks.exe* InstallShield package to install 2.2 on a new machine or upgrade from a version prior to 2.11b. This InstallShield package contains both client and server modules and prompts the user to select the InfiniLinks packages to install. Selecting the Server Tools option installs the server, alert agent, and script compilers. Selecting the Client Tools option installs the InfiniLinks Monitor and Remote Configuration Application (RCA). Selecting the Example Files option installs newly updated MDL files that include new and updated record definitions for X.12 and HL7 version 2.4.

For UNIX and Linux platforms, InfiniLinks continues to ship in two archive files. The two files under UNIX include a binary archive, which includes all InfiniLinks executable and library files, and a configuration archive, which contains updated configuration files.

As a reminder, queues created under InfiniLinks version 2.10 and lower are incompatible with queues created under InfiniLinks 2.11b and higher. There is no need to remove queues when upgrading from 2.11b to 2.20, however.

Changes were made to the binary file format for compiled CDL script files in InfiniLinks version 2.11b. When upgrading from versions 2.10 and lower, these CDL scripts will need to be recompiled. Install the new binaries and then run the *convertcnfg* utility. The program converts the *cdx* files for all communications components configured within the SIF. Alternately, the 2.11b or 2.20 *cdlmake* compiler can be used to manually update existing compiled script files.

No executable programs or dynamic-link libraries have been retired for this release of the program. Do not delete any binary file in the */bin/* directory, regardless of the file's last modified timestamp.

The following binary files have been added in this release:

File Name	Description
dhcftp.dll	This DLL contains DHSC's implementation of the File Transfer Protocol specification. The DLL is used by communications components to communicate with remote FTP servers.

Enhancements

Communications Components

Encryption

InfiniLinks now supports four flavors of DES encryption, along with AES introduced in version 2.1, using the built-in commands in a CDL script. The four new encryption processes are: DES, DES_EDE2, DES_EDE3, and DES_XEX3. Refer to Chapter 3 of the *InfiniLinks Reference Guide* for detailed information regarding the previously mentioned forms of DES is discussed in detail in Chapter 3.

The key piece to remember while using DES encryption is the length of the primary key. The length of the key is determined by the value, 64, 128, or 192, used while creating the keys with *keygen*. Plain vanilla DES requires a key length of 64, while DES_EDE2 requires a key length of 128, and DES_EDE3 and DES_XEX3 require a length of 192 bits. Specifying an invalid key length generates an error specifying the appropriate key length size required.

Component Modifications

The parameters were modified to the section called [Crypt], of the Communications Component Configuration File (3CF). This section is optional; however, the configurations must exist at the time the component starts to enable encryption.

The various entries for this section are:

Entry Name	Description
state	This value determines if the data shall be encrypted/decrypted when the appropriate CDL built-in function is executed.
BlockCipher	Requires one of the following values: AES, DES, DES_EDE2, DES_EDE3, DES_XEX3
CipherMode	This value is required for all defined BlockCiphers. Valid values are CBC, CFB, CTR, ECB, and OFB.
Coding	This setting defines how encrypted data is encoded. Valid values are Base64 or Hex.
KeyCoding	Valid values are None, Base64 or Hex. This setting defines how the keys are encoded so they can be decoded before use.
PrimaryKey	This value must contain the key file used for encrypting/decrypting data.
IVKey	The IV is a 128-bit key that is used in conjunction with the cipher modes for encrypting/decrypting data. This value is not required when the CipherMode is set to ECB.

FTP Protocol

Support for FTP has been added as a protocol for communications components. FTP support allows a communications component to communicate with an FTP server and either upload or download files from the remote machine. Most Linux and UNIX systems include an FTP server as part of the operating system, and Microsoft offers an FTP server solution for its server-grade operating systems. Many third parties also offer products that can be used with DOS or any version of Windows down to version 3.1.

The FTP implementation does not include full support for some of the more esoteric parts of the FTP standard. First, the InfiniLinks FTP client supports only the default FTP *stream* transmission mode. In this state the FTP client or server transmits a file as an unstructured stream of data and then closes the data channel socket to indicate the end of the file. This mode does not support any automatic recovery or error checking algorithms; the entire file must be resent in the event of transmission failure.

Communications support ASCII and binary file transfers. No support for EBCDIC or custom data types is included. It is possible for a communications component to support EBCDIC transmissions, however. Transmit EBCDIC data as binary and use the CDL language builtin commands for translating strings from EBCDIC to ASCII and vice-versa.

FTP communications components operate on a user-defined schedule. For an outbound component, discrete messages are cached on the local machine in a directory. The component's configuration file specifies a schedule for connecting with the FTP server and transferring the cached data. This is referred to as the data-exchange schedule.

There are two types of FTP schedules: batch and simulated real-time. In batch interfaces, transactions are accumulated in a file and held before being processed by an endpoint application. In these cases, a data file is transferred once or twice a day. Batch files are common in financial interfaces where one system will accumulate charges and credit throughout the day and then send the information to a second system after the business office is closed for the day.

With simulated real-time interfaces, messages are placed in individual files and transmitted to the receiving system at frequent intervals, such as every 30 or 60 seconds. The receiving system then has a scanner process that detects the delivery of new files, and processes the transactions in the file. Although this scheme is slower than a true real-time interface, in most cases it is a more than adequate substitute.

To specify a batch mode, set the *ScheduleType* parameter to 'schedule'. Then, specify the times the client should initiate a data exchange with the endpoint application in the *XferTime* parameter. The format for this field is HHMMSS, and multiple transfer times can be strung together using the pipe ("|") character. For example, a value of "060000|180000" would cause the client to initiate a data exchange at 6:00 AM and 6:00 PM daily.

For a simulated real-time interface, set the *ScheduleType* parameter to 'interval'. Then, specifying the amount of time to elapse between data exchanges in the *Interval* parameter. Finally, use the *IntervalType* parameter to indicate the units of measurement for the *Interval* parameter. Possible values are 'seconds', 'minutes', 'hours' and 'days.' If the *Interval* parameter is set to 6 and the *IntervalType* parameter is set to hours, the FTP client will initiate a data exchange every six hours.

#	Entry	Notes
1	HostName	Enter the host name or the IP address for the workstation on which the FTP server resides.
2	PortNumber	Enter the port number that the FTP server "listens" on for inbound connections. The FTP protocol specifies port 21 as the standard FTP port.

3	Mode	Use this parameter to specify the mode for file transfers. Specify 'A' or 'ASCII' to transfer all files as ASCII text. Enter 'B' or 'Binary' to transfer files as binary images.
4	UserName	Enter the user name that the FTP client will log in as. The requirements for the user name vary with the operating system that the FTP server runs under.
5	Password	Enter the password for the User specified in the UserName parameter.
6	FileName	For inbound components, this entry indicates which files to process within a given directory. This can be either a hard-coded file name, or a pattern-match string using any of the pattern matching characters. See the pattern matching section for more information. For an outbound component, this parameter is the name of the file to write data. Either this can be a hard-coded file name such as "filename.dat", or the parameter may contain a place holder that will be replaced with an incrementing sequence number. To include a sequence number in the file name, start the placeholder with the '%' sign. The next character is a number indicating how many characters in length the sequence number should be zero-filled. Finally, place a 'd' in the field to indicate the server should output a number. Take the following example: "file%5d.dat". The first file name created would be "file00001.dat", the second file name is "file00002.dat" and the third file name is "file000003.dat". The component tracks the sequence number internally and resets the sequence number to 0 when it reaches 10000.
7	LocalDir	For inbound components, this value represents the location where files transferred from the FTP server are copied to. The files remain here temporarily until the contents of the files are processed by the CDL script. For outbound components, this property indicates the directory outbound files are written to while waiting for the next scheduled FTP data exchange.
8	ArchiveDir	For inbound components, indicates the directory files are copied to after being processed by the CDL script. For outbound components, indicates the directory where files are archived after being uploaded to the FTP server.
9	ErrorDir	The directory to copy files to if a processing error occurs. Both inbound and outbound components use this parameter.
10	Overwrite	For inbound components, indicates whether a file downloaded from an FTP server should overwrite a file with the same name in the *LocalDir* directory.

		For outbound components, indicates whether a file should overwrite a file with the same name that already exists on the remote server. A value of 'on' indicates the file should not be overwritten, causing the file to be moved to the ErrorDir folder. A value of 'of' instructs the client to overwrite the file.
11	RemoteDir	Indicates the remote directory on the FTP server to open after establishing a connection with the FTP server. For inbound components, indicates the remote directory files are downloaded from, while for outbound components, indicates the directory where files are written.
12	FTPMode	A value of 'P' or 'Passive' or this property indicates that the client will request the FTP server place itself into passive mode. A value of 'N' or 'Native' indicates that the FTP client/server will use the default method for establishing a data channel.
13	ScheduleType	A value of 'interval' for this property indicates that the client will use the *Interval* and *IntervalType* properties to determine the data exchange schedule with the FTP server. A value of 'schedule' instructs the FTP client to use the *XferTime* property to set up its data exchange schedule.
14	IntervalType	Indicates the unit of measurement for the interval property. Valid values are 'seconds', 'minutes', 'hours', and 'days.'
15	XferTime	One or more data exchange times in the format of HHMMSS. Multiple values can be concatenated together using the '|' character. A value of 090100|210000 instructs the client to perform a data exchange at 09:01 AM and 09:00 PM daily.
16	Interval	Indicates the amount of time to elapse before starting a new data exchange with the FTP server.
17	DataPort	For passive FTP communications, indicates the port number the FTP server connects to on the local machine to establish the FTP data channel.

RCA

Conveniently, the "Setup INI File" option of the RCA has been updated to include support for the FTP protocol. **Figure 1.1** shows the new FTP page in action:

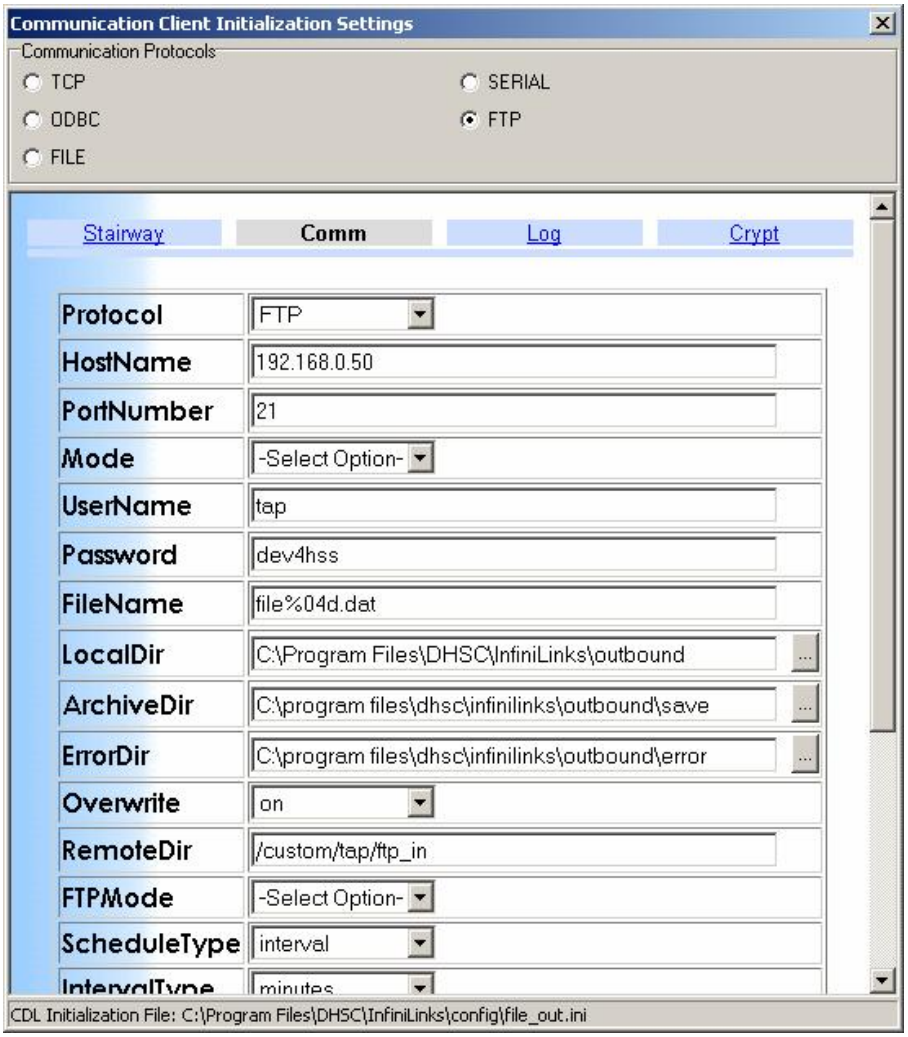

Figure 1.1 -- FTP Set up options

To access the page, bring up the RCA and select the Communications tab. From the drop-down box, select an existing communications component or define a new one. Click on the "Setup INI File" button. The component configuration window appears.

Under "Communications Protocols," select FTP. A list of all the entries for the FTP communications protocol are listed. When all entries are completed, scroll to the bottom of the HTML page and click "Save". The RCA will quickly validate the settings and save the information to the appropriate configuration file.

If errors are detected, the RCA displays a message box informing the user of the errors. Correct the errors and click on the "Save" or "Close" button again.

Documentation

The *InfiniLinks Reference Guide* has been updated for version 2.2. The guide is located in the *docs* directory underneath the root InfiniLinks directory. Chapter 3 has been updated

with new information on the new encryption options. Chapter 4 has also been updated with detailed information on our implementation of the FTP protocol.

Bug Fixes/Resolved Issues

The following issues were resolved in this release of InfiniLinks:

InfiniLinks Server

- **Server port lockup in 2.1 (00001353) –** The server would occasionally lock up when an excessive number of inbound client connection attempts were made in a short period. The server process continued to run, but routers stopped processing data and new inbound client requests were not serviced. The problem occurred because of mutex contention between an inbound communications component and the Log File Manager. The problem only manifests itself when a large number of messages are sent to the log file in a short period of time. The problem has been resolved.
- **Encryption initialization process crashes server (HP-UX only) (00001352) –** Under HP-UX, the encryption initialization process causes the server to exit with a "Segmentation Fault." operating system. The server has been modified to bypass initialization of the encryption libraries if encryption is set to "off" in the Server Initialization File (SIF). The problem is isolated to v11.00 E of the HP-UX and will not be supported under this version of the operating system. Upgrade to v11.00 U of HP-UX to enable encryption support.

Monitor

- **Search returns all records found, regardless of total number (00001100) –** In previous versions of InfiniLinks, performing a search on a queue returned all matching records. If a large number of matches were encountered, it could cause the server. Monitor, or both, to hang or crash. The problem has been resolved. The server now returns a preset number of records, and the user may browse through the search results as if they were browsing through the queue.

RCA

- **RCA allows data entry of restricted characters (00001260) –** Previous versions of the RCA allowed the entry of restricted characters in critical data entry fields. Using these characters would cause the server to reject the server.ini file. The problem has been resolved.

Communications Component

- **Outbound comm. components stop processing data (00001355) –** Under 2.11b, outbound communications components would occasionally stall while processing data. This problem occurred because two threads would poll the socket for data and the second thread would process the data before the first thread could reset itself. The problem will likely only manifest itself on multi-processor systems. The problem has been fixed.
- **Outbound comm. components hang during shut down (00001357) –** Under 2.11b, outbound components would occasionally hang during the shut down process, eating up CPU resources. The only way to stop the process is to manually kill it. The problem occurred because of thread contention during the shut down process. This problem will likely only manifest itself on multi-processor systems.

Known Issues/Errata

InfiniLinks Server

- **Tab characters in SIF cause server to fail to start (00001347) –** Using a TAB character anywhere in the SIF prevents the server from processing the file. Do not use TAB characters in any InfiniLinks initialization files. This issue will be corrected in an interim release.
- **Excessive messages in log file when a queue requires a rebuild (00001348) –** The server will occasionally log a series of messages in the log file regarding inbound queues. The message is "The requested operation could not be completed. Missing error message in initialization file. Error code is: 0". If this problem occurs, stop the queue in question and perform a rebuild operation. This problem will be corrected in a future release
- **(Linux only) Server does not properly execute write lock on server log file –** Under newer versions of Red Hat Linux, the operating system does not lock the log file for writing while the server is running. This means that the log file could inadvertently be overwritten by a user or by invoking a second instance of the server while the first is still running. This issue will be examined later.

MML sub-module

- **Extraneous sub-component delimiters added to translated messages (00001292) –** Using an MDL file with sub-component delimiters and the NOGARBAGE keyword causes the last name to be written to the outbound message as the following: ALANIS\\\\\^. If the sub-component delimiters are present in the source message, the translated field becomes ALANIS\\\\\\\\\\^. This will be fixed in an interim release in future versions.

CDL sub-module

- **Nulls in encrypted strings not handled properly (00001289) –** Due to the storage architecture of the CDL sub-module, CDL scripts do not function properly with encrypted, unencoded data. . The option for no encoding has been removed from the CCCF. This option may be restored in the future if sufficient interest is expressed. .

Client Test Utility

- **Request Timed Out Message when rebuilding a queue (00001349) –** When executing the rebuild command against a large queue, the following message may appear, "Request time out. Continue to wait? (Y/N):" This message appears because the operation takes longer to complete than the time *clientt* waits for a response from the server. Select Y to continue to wait. The command will eventually complete.

Alert Agent

- **Comm. Activity Time out events (00001302) –** Problems may occur when using the Alert Agent to monitor a server in a different time zone for comm. component inactivity. The alert agent compares a timestamp from the server with a timestamp from the local machine to determine if the comm. inactivity safe interval has elapsed, without taking account the time zone for either machine. This may result in invalid Comm. Activity Timeout events generated.
- **Viewing Alerts (00001301) –** Response time may degrade significantly when invoking the events list from THE_CONFIGURATOR or the Alert Agent on a database containing a large number of events. This behavior will be addressed in a future release.

Agent Test Utility

- **Alert Agent crash when using the LIST command (00001350) –** Using the list command to display a large number of events (100+) in the Agent Test Utility may cause the Alert Agent to crash. This will be addressed in a future release.