InfiniLinks Version 2.10

Release Notes

Dagran Healthcare Systems Consulting

Thank you for using InfiniLinks. Version 2.1 release of the InfiniLinks software has no restrictions on use and may be installed in production environments at each customer's discretion. This release features support for encryption, changes to the server configuration utility, and a new alerting agent that monitors one or more servers for error conditions.

Please read this document thoroughly prior to performing an upgrade. Many new features have been added and, in some cases, the behavior of existing functionality has changed. Familiarizing oneself with these changes will help to avoid many common pitfalls during the upgrade process.

Installation Notes

Unlike previous versions of the program, both the client and server portions of InfiniLinks are released as a single InstallShield package under Windows. The InstallShield package now prompts the user to select the InfiniLinks packages to install prior to installation. 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 binary 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 and sample files, including the updated MDL files.

Due to changes in how queue index files are maintained by the server, queues created using previous versions of InfiniLinks are not compatible with version 2.1.When performing an upgrade, it is required that the contents of the queue directory be completely deleted prior to starting the install process. This applies to all platforms.

Changes were made to the binary file format for compiled CDL script files. CDX files are now platform neutral, meaning that a script compiled on Windows may be used under Linux and HP-UX. Unfortunately, these changes necessitate recompiling all existing CDL scripts prior to starting the version 2.1 InfiniLinks server. 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 2.1 *cdlmake* compiler can be used to manually update existing compiled script files.

Version 2.1 introduces a handful of changes to the Server Initialization File (SIF), located in the */config* directory underneath the root installation directory. To aid you in upgrading an existing installation, the *convertcnfg* utility has been revised. The utility will update the SIF and all communications component configuration files to meet version 2.1 requirements. It will also backup and remove your version 1.4 and 2.0 queue files, as they are incompatible with version 2.1.

There have been some additions to the default InfiniLinks directory structure. The following directories have been added to the base InfiniLinks directory:

Directory	**Description**
key	This directory is the default location for storing public and private key files created using the new *keygen* utility. It is recommended,

	but not enforced, that all keys used by communications components be stored here.
Templates	This directory is the default location for storing e-mail templates created using THE_CONFIGURATOR. It is recommended, but not enforced, that all template files be stored here.

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 modification timestamp.

The following binary files have been added in this release:

File Name	**Description**
cryptopp.dll	Contains the generic 3DES and RSA encryption routines used by InfiniLinks and its communications components.
dhccrypt.dll	Contains the interface routines to integrate the generic encryption algorithms in cryptopp.dll into the InfiniLinks object architecture.
dhcgui.exe	THE_CONFIGURATOR. Use this application to create and modify Alert Agent Configuration Files (AACF).
dhcmd5.dll	Contains the generic MD5 hash algorithm used by the Alert Agent to authenticate user names and passwords with an SMTP server.
dhcsink.dll	Contains generic routines for integrating Microsoft's Internet Explorer with InfiniLinks client applications.
dhcsmtp.dll	This DLL contains the Simple Mail Transfer Protocol (SMTP) implementation for sending outbound e-mail messages.
dhcutil.dll	A library containing utility functions used by the InfiniLinks server and client applications.
dhcwatch.exe	The InfiniLinks Alert Agent. Use this application to monitor one or more InfiniLinks servers for errors.
keygen.exe	Use this utility to generate encryption keys for use with an InfiniLinks communications component.
newconfig.dll	Contains routines used by THE_CONFIGURATOR to create and modify Alert Agent Configuration Files (AACF).

Enhancements

Server

Encryption

New to version 2.1, InfiniLinks now supports AES and RSA encryption between InfiniLinks components and to the outside world. InfiniLinks' support for encryption can be broken down into functional categories that are listed below.

Queue Encryption – InfiniLinks can be instructed to encrypt data within InfiniLinks queue files. This type of encryption prevents an intruder from looking at private data in the queues, even if he has access to the physical files that comprise a queue.

Please note that only the raw message is encrypted within the queue. For performance considerations, user-defined fields are not encrypted. For medical institutions, it is not recommended that user-defined fields contain clinical information protected under HIPAA.

Client Tools Encryption – The server and all client tools that connect to the server now support encrypted communications. This allows users to connect securely to an InfiniLinks Server across an insecure medium, such as the Internet. Client tools that support encryption include the Monitor, clientt, RCA, InfiniLinks IDK applications, and the Alert Agent.

No modifications to settings are required within the client application to support encrypted communications. When client tools encryption is enabled within the server, the server sends a single-use key to the client. When the client receives the key, it loads encryption support and begins coding all traffic to the server using the received key. The key sent from the server is randomly generated and is only valid for the current session.

Use caution when enabling encryption support over the Internet, as it involves exposing the port that the InfiniLinks server "listens" on. This has the potential of opening the InfiniLinks server to malicious activity, including Denial of Service attacks and packet spoofing. DHSC always recommends using encrypted communications protocols such as Virtual Private Networks (VPNs) in conjunction with client tools encryption.

Component Communications Encryption – Component encryption behaves similarly to client tools encryption, in that all network traffic between communications component processes and the InfiniLinks Server are encrypted. This is useful when communications components are running on a different workstation than the InfiniLinks server process. The same caveats and recommendations for client tools encryption also apply to component encryption as well.

All forms of server encryption are configured within the SIF. All settings are contained with the [Encrypt] section. An example of the [Encrypt] section is illustrated below:

```
[Encrypt]
Queues=0
CommClients=0
ClientTools=0
```

In this example, encryption in all forms is disabled. A value of '0' for each field disables encryption for that category, while a value of '1' enables it. After changing these settings, the InfiniLinks Server needs to be restarted for the changes to take effect.

Enabling queue encryption on a Server using queues with unencrypted data requires performing another step. Unencrypted queues need to use the *qconv* utility to encrypt the queues. Attempting to start the server with unencrypted queues will result in the server shutting down with an error. Likewise, attempting to start a server with encrypted queues while the queue encryption flag is disabled also causes an error.

<u>Labels</u>
The server has also been enhanced to include "labels." A label is a way to logically group components within the Monitor. To support this, a new [Labels] section has been added to the SIF. Below is an example section:

```
[Labels]
LabelCount=2
Label0=ADT,ADT Components
Label1=DHSC,DHSC Example Label
```

As you can see from the example above, the section is composed of a control entry called LabelCount, and a repeating Label*n* entry. Each Label entry has two parameters. The first parameter is the name of the Label, and the second parameter is a helpful description.

A label can be associated with a queue, router or communications component. To add a component to a label group, add the label name to the end of the component definition. For example, to add the queue In1 group add the string “ADT” to the end of its entry in the [Queues] section. The following example illustrates the modifications that are required:

```
Before: Queue0=In1,C:\Program Files\DHSC\InfiniLinks\Queues\2.1\In1,1,I,10000,START
After: Queue0=In1,C:\Program Files\DHSC\InfiniLinks\Queues\2.1\In1,1,I,10000,START, ADT
```

The [Labels] section is optional and does not need to be present in the SIF for the server to start. However, even if the section is present, it is not required that all InfiniLinks components be associated with a label group. The server will start normally if one or more components do not have a value in the labels property.

Log File

Two new parameters have been added to the [LogFile] section. The first parameter is the Overwrite entry. This is a binary parameter. Setting the parameter to 0 causes the server to create a backup of the DHC log file at startup, if the log file already exists. If the log file is named *dhc.log*, the server attempts to rename the file to *dhc.log.0.* The server increments the number at the end of the file name until it finds a unique name that does not exist.

The second parameter is the MaxFileSize parameter. The value of this parameter is the maximum size of the log file, in MB, that the log file may grow to before it is archived following the process above. Valid values for this entry are any number within the range of 1 (1 MB) to 1024 (1GB).

Other Changes

Two other minor changes were made to the InfiniLinks server. The server now tracks the last time message activity occurred from a communications component. This parameter may be viewed within the Monitor. For inbound components, it indicates that last time a message was received from an endpoint application. For outbound components, it indicates the last time a message was sent to an endpoint application.

The second change occurs when the server detects an error during the startup process. Using the *dhcserv* executable under Windows, the server now displays a message box indicating the error. Under Linux and HP-UX, the application displays the error message on the console that started the process. This allows the user to determine the error that prevented the server from starting without having to open the log file.

qconv Utility

The *qconv* utility has been upgraded to include encryption support. *qconv* now has two modes: the standard mode, which is used to update a queue when queue field definitions have changed, and encryption mode. The standard mode behaves identically to previous versions. Consult the InfiniLinks Reference Guide for more information on using *qconv* in standard mode.

Encryption mode encrypts or decrypts queue files based on the parameters provided. To invoke encryption mode, type in *qconv* followed by one of the following arguments:

Argument	**Description**

-e	*Qconv* encrypts all queues configured in the server.ini file. A backup copy is made of each queue.
-eb	Encrypts all queues configured in the server.ini file. No queue backup files are created.
-d	*Qconv* decrypts all queues configured in the server.ini file. A backup of each queue is created.
-db	Decrypts all queues configured in the server.ini file. No queue backups are created.

DHSC recommends that a backup always be performed when performing an encryption/decryption option. Use the –eb and -db parameters at your own risk.

In encryption mode, *qconv* encrypts or decrypts all queues in the server.ini file. The utility checks the encryption settings in the SIF prior to starting the process. If an attempt is made to encrypt queues with the queue encryption setting toggled off, the utility throws an error and exits. If an encryption or decryption request is made on a queue file that is already encrypted/decrypted, the utility ignores the file and continues.

Communications Components

InfiniLinks now also supports inter-application encryption via new builitn commands in a CDL script. See the section on communications changes later in this chapter for more information.

Component Modifications

In addition to supporting inter-process encryption between the server and component, encryption support has also been added to the CDL scripting language. A new section, called [Crypt], has been added to the Communications Component Configuration File (3CF). This section is optional; however, it must be present at the time the component is started for encryption to be enabled.

The various entries for this section are:

Entry Name	**Description**
State	This value is used to determine if the data shall be encrypted/decrypted when the appropriate CDL builtin function is executed.
BlockCipher	There are currently 2 valid values for this setting, AES or RSA.
CipherMode	This value is only required when the BlockCipher is set to AES. Valid values are CBC, CFB, CTR, ECB, and OFB.
Coding	This setting defines how the 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.
AESKey/RSApub	This value must contain the AES key or RSA public key file used for encrypting/decrypting data. When BlockCipher equals AES, the value must contain the path to the AES key, and when BlockCipher is set to RSA, the value must contain the path and filename to the RSA public key.
IV/RSApriv	The IV is a 128-bit key that is used in conjunction with the AES key for encrypting/decrypting data. This value is not required when

	the CipherMode is set to ECB. This value holds the RSA private key when BlockCipher is set to RSA.

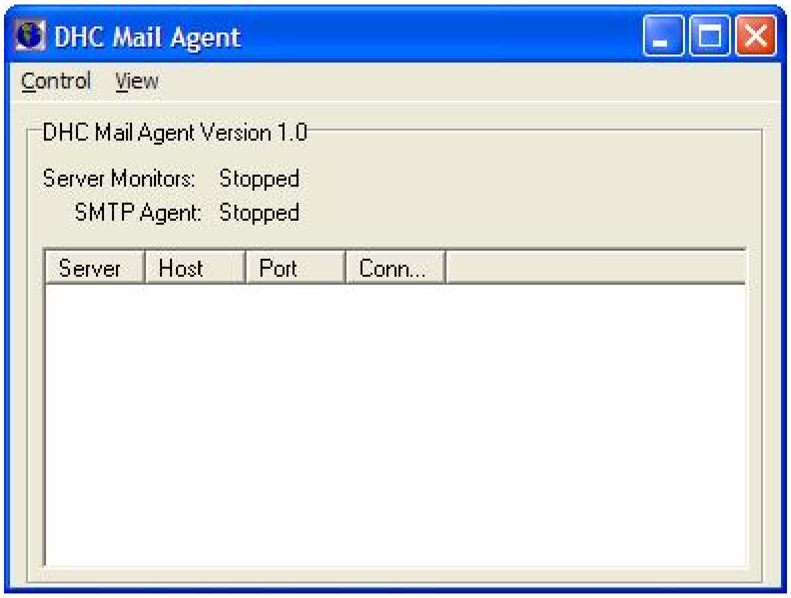
Communications Definition Language (CDL) Enhancements
The user may encrypt or decrypt data within a CDL script via two new built-ins, ENCRYPT and DECRYPT. The ENCRYPT builtin takes two arguments: destination variable and source string. The source string is the unencrypted value that is encrypted according to the parameters set in the [Crypt] section. The encrypted string is then placed in the destination variable.

The DECRYPT builtin operates similarly by taking the same two arguments in the same order. The source string is an encrypted string. The value is decrypted using the key and settings in the [Crypt] section. The unencrypted string is then stuffed into the destination variable.

As with all builtin commands, the destination variable is the first argument and the source string is the second argument.

Mail Agent

New to version 2.1 is the InfiniLinks Mail Agent, a utility designed to monitor one or more running InfiniLinks servers for errors. If an error occurs, the Agent can be configured to generate an outbound e-mail alert to a pre-configured e-mail address. **Figure 1-1** shows the Windows version of the mail agent.



Server Monitors: Stopped
SMTP Agent: Stopped

Figure 1-1 The InfiniLinks Mail Agent

Via the Mail Agent's configuration file, users may set up the Mail Agent to monitor multiple InfiniLinks servers simultaneously. The configuration file contains the name and location of the server to monitor, as well as various interval and threshold values that control when an alert is generated.

The Mail Agent is multi-threaded and each InfiniLinks is monitored in a separate thread. When an error or other important activity is noted by the agent, an *event* is generated. Events contain information about the issue that occurred on the server. Information associated with an event includes the type of event that occurred, the time the event occurred, and its status.

Events are placed into a database maintained by the Mail Agent. All events are considered unprocessed until the e-mail subsystem processes the event. The mail subsystem is responsible for generating an outbound e-mail from the event and sending to one or more recipients via a standard SMTP server. The e-mail subsystem runs in its own thread and can be enabled or disabled independently of the threads that monitor InfiniLinks servers.

E-mail messages are generated from template files. By configuring a template, a user defines who will receive the message and what information will be included in the subject and body of the e-mail. Through tags in the code, a user can mark areas of the email that will be replaced with information from the event generated by the server monitor. Templates are generated by using a template designer and are associated with events in the agent's configuration file.

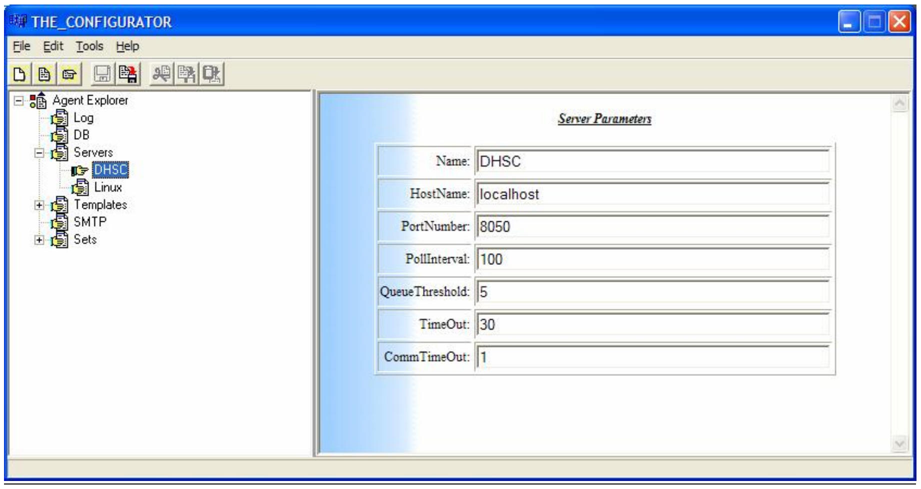
By defining a series of rules within the configuration file, users can define logic for determining who will receive an e-mail or whether an e-mail should be generated from the current event. From these rules, a user may choose to send an alert to one person on a weekday, but configure a separate rule to send alerts to a different person on a weekend. The user could also set up a rule to exclude generating any outbound messages for a particular InfiniLinks component

THE_CONFIGURATOR

THE_CONFIGURATOR is a graphical user-interface application that allows users to create and modify Alert Agent Configuration Files (AACF). Files created using this tool are loaded into the *dhcwatch* application for use with the InfiniLinks Mail Agent. THE_CONFIGURATOR **is illustrated in Figure 1-2** ..

Use THE_CONFIGURATOR to define which InfiniLinks servers the Mail Agent will monitor and configure rules that will determine which e-mail message are generated from events that occur on a server. Once these parameters are set up, THE_CONFIGURATOR provides a validation tool, allowing the user to confirm that the file is properly configured before it is loaded into the Mail Agent.

THE_CONFIGURATOR also includes a tool for viewing the events stored in the Mail Agent's database. From here, a user can modify the contents of the event and change the event's status. Unprocessed events can be modified or deleted, and handled events can be reset to unprocessed. THE_CONFIGURATOR allows the user full control over all alerts generated by the agent. Please note that events cannot be viewed or modified while the mail agent is running.



Server Parameters

Name:	DHSC
HostName:	localhost
PortNumber:	8050
PollInterval:	100
QueueThreshold:	5
TimeOut:	30
CommTimeOut:	1

Figure 1-2 -- THE_CONFIGURATOR

Finally, THE_CONFIGURATOR also includes a full featured template designer, providing a WYSIWYG editor for designing e-mail templates. Not only does the designer allow the user to determine who receives a message, but also configure what the message will contain and what the body of the message will look like when it is sent.

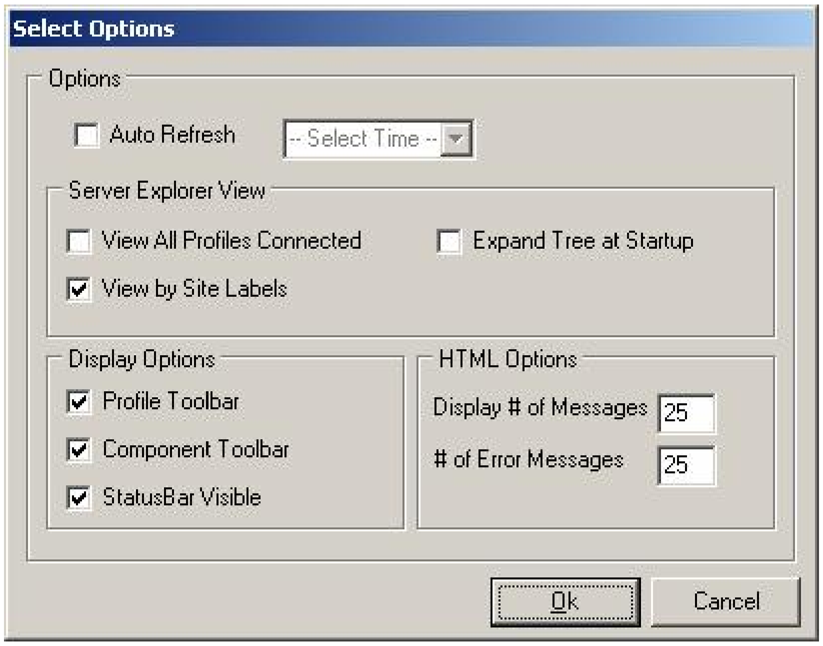
<u>Agent Test Utility</u>
The Agent Test Utility is a non-Windows application that allows users to control an InfiniLinks Mail Agent running on a Linux or HP-UX server. The Agent Test Utility is a command-line, “roll-and-scroll” application similar to the *clientt* utility used to control an InfiniLinks Server. Don’t let its command-line interface fool you; however, the Agent Test Utility provides user with the same level of control (and in some cases more) over the Mail Agent that Windows users have.

The Mail Agent runs as a background process on non-Windows platforms. The Mail Agent opens a TCP/IP server socket for the Agent Test Utility to connect. When the user starts the utility, it connects to the Mail Agent and allows the user to control all of the Agent’s subsystems.

<u>Monitor Enhancements</u>
A critical new feature added to the Monitor is support for custom filters. With increasingly complex InfiniLinks deployments, some servers now maintain over 200 InfiniLinks components. A mechanism was added to the Monitor to allow users to define multiple views that filter out unwanted components from the display and show only those components necessary to deal with the issue at hand.

Defining a view starts in the Server Initialization File (SIF). The SIF now has a section for [Labels] and each InfiniLinks component now has a corresponding label component. Use labels to group like-related components. For example, users may group all components for a particular facility under one label, or group all of the components associated with a particular ADT interface under a different label.

When the Monitor connects to the server, it automatically downloads all of the labels defined within the SIF. The labels are then displayed in a combo box attached to the Profiles tool bar. This combo box is disabled by default. To enable support for custom views, select the ‘Options’ item on the Advanced menu. In the resulting dialog, shown in **Figure 1.4**, select the ‘View By Site Labels’ check box and press OK. The Labels combo box is now enabled.



Select Options

Options

- [ ] Auto Refresh -- Select Time --

Server Explorer View

- [ ] View All Profiles Connected
- [ ] Expand Tree at Startup
- [✓] View by Site Labels

Display Options

- [✓] Profile Toolbar
- [✓] Component Toolbar
- [✓] StatusBar Visible

HTML Options

Display # of Messages	25
# of Error Messages	25

Figure 1.4 -- The Monitor Options Dialog

To filter the InfiniLinks components by a particular label, select a label name within the combo box. The current profile is updated by hiding all components that are not associated with the selected label. **Figure 1.5** illustrates the changes made to the profile when a particular label is selected.

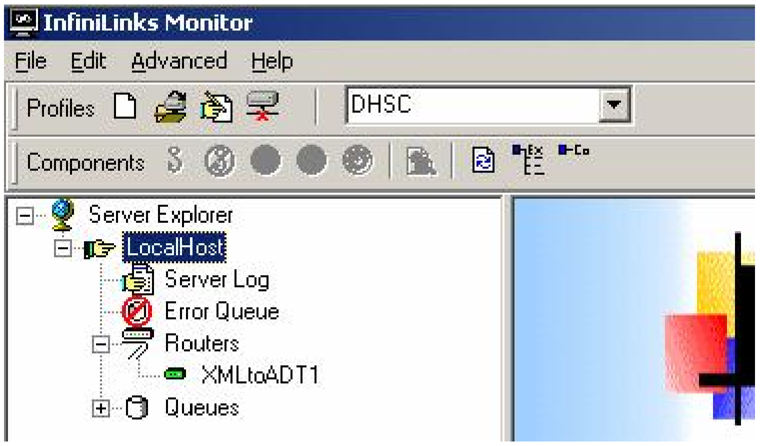


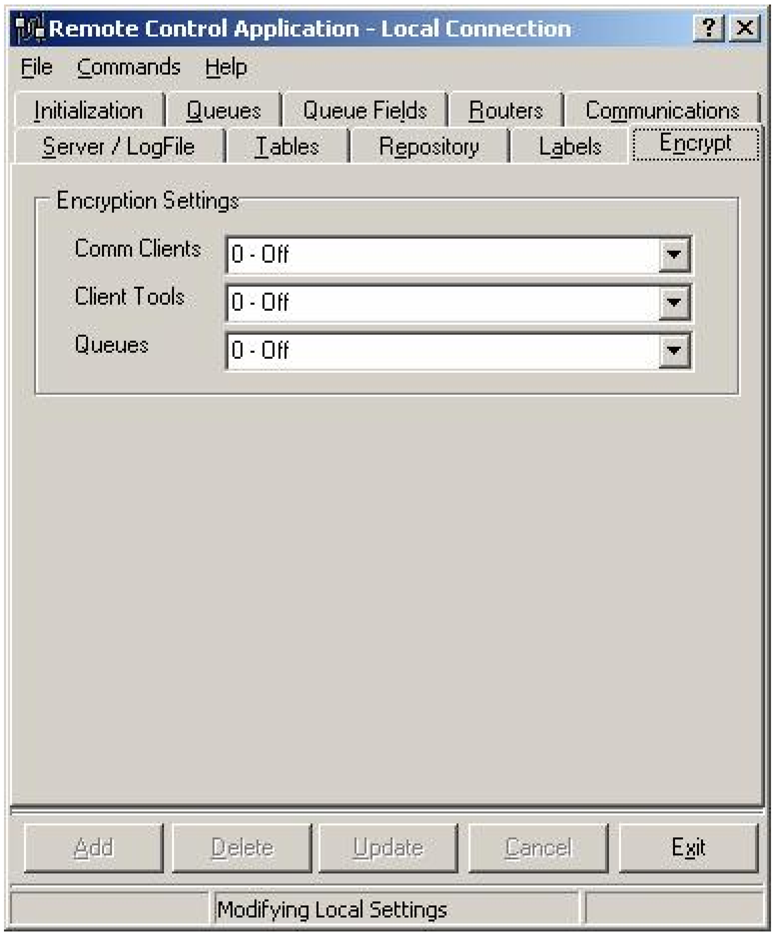
Figure 1.5 -- Profile filtered via Labels

In this particular example, the InfiniLinks server associated with the profile has five router components defined. XMLtoADT1 is the only router displayed, as it is the only one that has its label property set to 'DHSC'.

To stop filtering and display all InfiniLinks components again, select 'View All' from the Labels combo box. To disable filtering completely, return to the 'Select Options' dialog and check View All Profiles Connected'.

<u>RCA Enhancements</u>

A number of changes have been made to the Remote Control Application (RCA). Most noticeably, two new tabs have been added. The first tab is labeled encryption and is illustrated in **Figure 1-6**. Use this tab to toggle the server's encryption parameters on and off.


Remote Control Application - Local Connection
File Commands Help
Initialization Queues Queue Fields Routers Communications
Server / LogFile Tables Repository Labels Encrypt
Add Delete Update Cancel Exit

Encryption Settings

Comm Clients 0 - Off

Client Tools 0 - Off

Queues 0 - Off

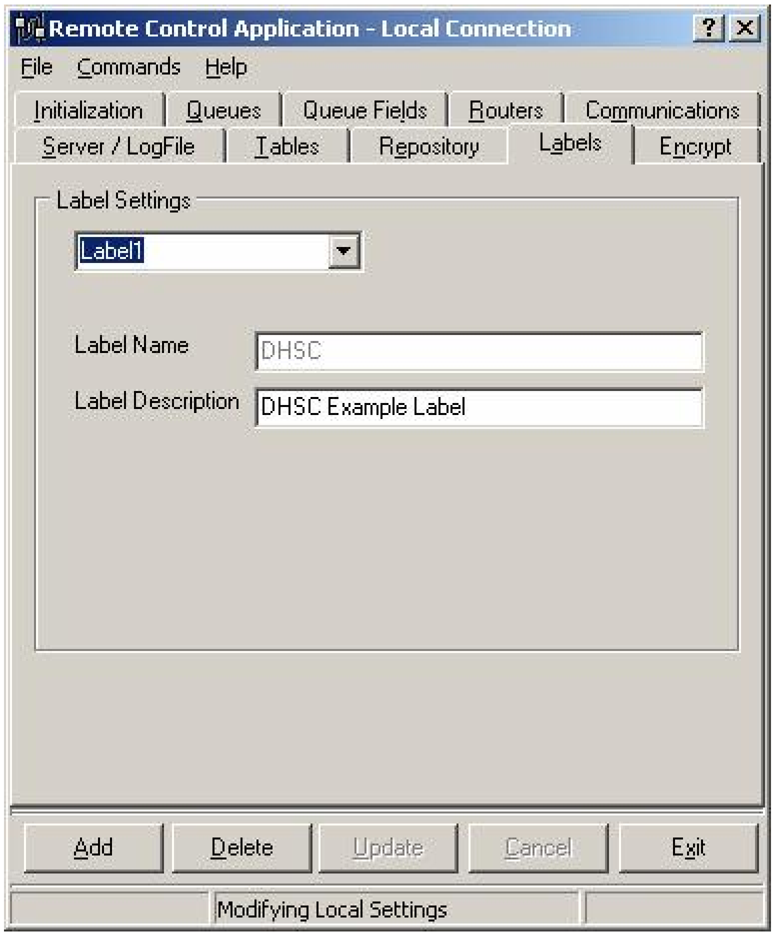
Modifying Local Settings

Figure 1.6 -- The RCA's Encryption Tab

The values on this page correspond to the [Encrypt] section of the SIF. For all parameters, setting the value to '0' disables the encryption parameter, while setting the value to "1", enables the parameter.

The second tab, shown in **Figure 1-7**, is the Labels tab. Use this section to configure component labels (discussed in the previous section). All labels must be configured on this tab before the label description can be associated with the component.

To complement this tab, all high level InfiniLinks components (routers, queues, and communications components) now have a new 'Label' property on their respective pages. When adding or modifying a component, the RCA shows a combo box that lists all of the labels defined in the SIF. Select a label from the list to associate it with the component.



Label Settings

Label1

Label Name: DHSC

Label Description: DHSC Example Label

Modifying Local Settings

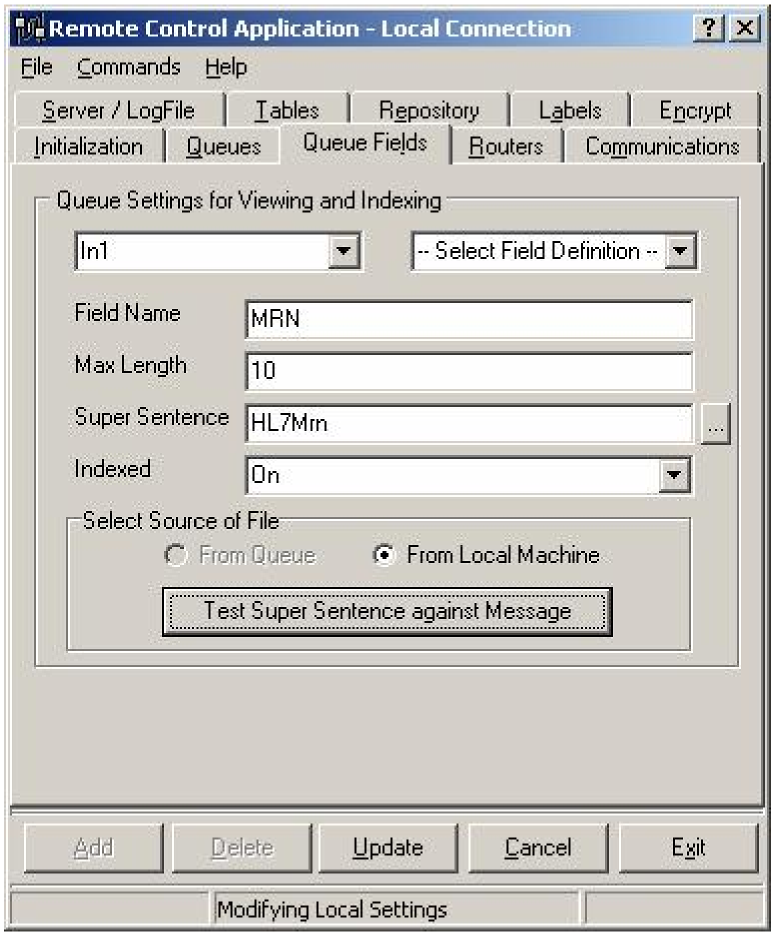
Figure 1.7 -- The RCA's Label Tab

Labels are optional parameters for a component. Leaving the property blank indicates that the component will not be associated with a custom view and will only display when ‘View All’ is set as the active filter in the Monitor.

Finally, the “Queue Fields’ tab has been modified to allow users to test queue field definitions against a message stored in the queue. **Figure 1.7** shows the changes made to this tab.

To test a queue field, first select a queue and a field in the combo boxes at the top of the panel. Users may choose to test against a message stored in a disk file or stored within the queue on the InfiniLinks server. Please note that testing a sentence from a queue is available only when the RCA has a remote connection with an active server.

Once a source for the message has been chosen, the Test Repository Value dialog appears. The raw message appears in the source window. Click the ‘Test TML’ button to test the queue field definition against the raw message. When finished, click on the ‘x’ in the top right corner of the window to dismiss the dialog box.



Queue Settings for Viewing and Indexing

In1 | -- Select Field Definition --

Field Name: MRN

Max Length: 10

Super Sentence: HL7Mrn

Indexed: On

Select Source of File: From Queue [✓ From Local Machine]

Test Super Sentence against Message

Modifying Local Settings

Figure 1.8 -- The RCA version 2.1 Queue Fields Tab

New Template Files

This release of InfiniLinks also sees the addition of two new MDL structure files. The first structure file is HL7v24.mdl and includes record and field definitions for the HL7 version 2.4 standard. This file contains full definitions for all of the most commonly used message types including ADT, orders, results, billing, scheduling, and pharmacy orders.

The second file is x12.mdl, and includes record and field definitions for claims and billing messages. Both template files are located in the *mdx* directory, located underneath the root InfiniLinks directory.

Documentation

The *InfiniLinks Reference Guide* has been updated for version 2.1. The guide is located in the *docs* directory underneath the root InfiniLinks directory. A chapter on encryption and a chapter on the Alert Agent and THE_CONFIGURATOR have been added. Also, please review the installation section in Chapter 1 prior to installing the software.

Documentation for the Monitor and RCA is provided in the form of Help files, which are located in the */help* directory underneath the default installation directory. The help file for the Monitor is *monitor.hlp*, while the help file for the RCA is *rca.hlp*. The Mail Agent and THE_CONFGIURATOR share the *agent.hlp* file.

Known Issues/Errata

InfiniLinks Server

- **Tab characters in SIF cause server to fail to start –** 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.

MML sub-module

- **Changed behavior in copying repeating fields –** When performing a copy on a repeating field, requesting a copy operation on an instance of a field that does not exist resulted in a NULL value placed in the destination node in previous versions of InfiniLinks. IN version 2.1 requesting a node that does not exist resulting in error message -4094.
- **Extraneous sub-component delimiters added to translated messages –** 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 –** 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. .