UDP Relay

Version: 0.9.1

Natural Resources Canada
Geodetic Survey Division

September 5, 2003

Table of Contents

Table of Contents i
List of Figures ii
List of Tables ii
Introduction 1
Relay to Relay Messages 4
Relay Shared Memory 5
  Connection Group Shared Memory Area 5
  Connection Shared Memory Area 6
Configuration File 7
Relay Functions 12
  Get Message Type Function 13
  Get Station ID Function 13
  Translate Message Function 14
  Custom Administration Request Hook Function 14
Administration Requests 15
  AddSite Admin Request 16
  DeleteSite Admin Request 16
  AddSendTo Admin Request 17
  DeleteSendTo Admin Request 17
  AddConnection Admin Request 17
  DeleteConnection Admin Request 19
  ModifyConnection Admin Request 19
  QueryConfig Admin Request 20
Admin Responses 20
CUDPRelayAdmin C++ Class 21
udpRelayAdminCGI Program 24
  Template HTML Files 26
    udpRelayErr.html 26
    udpRelayInvReq.html 26
    udpRelayMissing.html 26
    udpRelayConnect.html 27
    udpRelaySite.html 27
    udpRelayModConn.html 28
    udpRelayAddConn.html 30
    udpRelaySendTo.html 30
  Template Actions 31
    CONNECT Action 31
    ADD_SITE Action 31
    DEL_SITE Action 32
    ADD_CONN Action 32
    DEL_CONN Action 33
    MOD_CONN Action 33

SITE_INFO Action 34
CONN_INFO Action 34
SENDTO_INFO Action 35
ADD_SENDTO Action 35
DEL_SENDTO Action 35
FILL_TMPT Action 36
Glossary 38

List of Figures

Figure 1: Single udpRelayd with Multiple Data Source(s)/Destination(s) 1
Figure 2: udpRelayd to udpRelayd with Multiple Data Source(s)/Destination(s) 1
Figure 3: udpRelayd Message Header 4
Figure 4: Connection Group Shared Memory Structure 5
Figure 5: Connection Shared Memory Structure 6
Figure 6: Relay Configuration File Example 10

List of Tables

Table 1: udpRelayd Actions 2
Table 2: udpRelayd Command Line Options 2
Table 3: udpRelayd Message Header 4
Table 4: udpReader Command Line Options 5
Table 5: Relay XML Configuration Tags 7

Introduction

This document describes the UDP (User Datagram Protocol) relay tool. The UDP relay tool, udpRelayd, is a generic daemon process that is used to "relay" data using UDP in (near) real-time. The udpRelayd process can be used to ensure that the data is securely and reliably "relayed" by enabling encryption. All messages "relayed" between udpRelayd processed are MAC (Message Authentication Code) to ensure message integrity. The encryption and MAC is performed using public/private keys that can be dynamically generated for each session, or fixed by specifying a phrase.

The "relaying" of data is performed based on source and destination identification. This identification for the udpRelayd is the "station ID". Each source and destination connection is identified by this "station ID".

Messages that are transmitted between relays can be optionally marked to provide a guarantee delivery by requiring an acknowledgement of receipt. Messages can be optionally "translated" before they are "relayed" to a destination.

The udpRelayd process supports the following configurations:

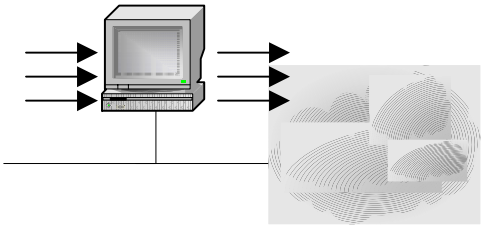

Figure 1: Single udpRelayd with Multiple Data Source(s)/Destination(s)

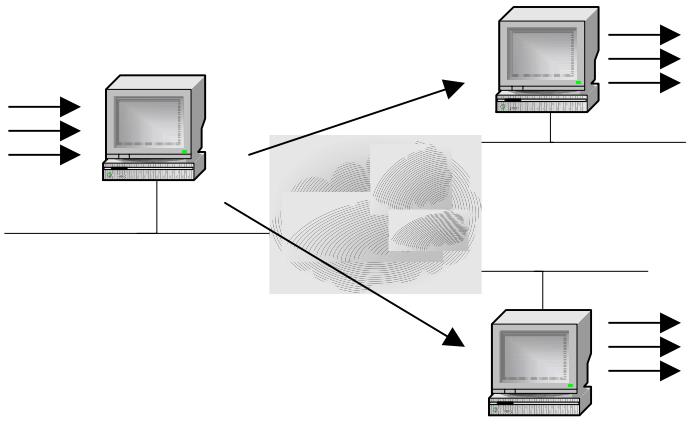

Figure 2: udpRelayd to udpRelayd with Multiple Data Source(s)/Destination(s)

Note that the udpRelayd **only** supports UDP Unicast or Multicast connections. Note that some routers may not support Multicast.

Depending on the connection, various actions can be performed as follows:

Table 1: udpRelayd Actions

Connection Type	MAC	Encryption	Message Type Identification	Message Translation
Relay to Relay	Yes	Optional	No	No
Source to Relay	No	No	Yes	No
Relay to Destination	No	No	No	Optional

The udpRelayd process supports logging of events to either a user supplied log file or to the system log file (syslog). This log file can also be used for debugging purposes to show all messages received and "relayed".

The udpRelayd process was written to run as an UNIX daemon process (in the future as an NT service). The following command line options are available:

Table 2: udpRelayd Command Line Options

Command Line Option	Description
`-c configFile`	Specifies the configuration file.
`-d debugLevel`	Run in debug mode; the debugLevel determines how much information is logged. Currently the maximum debugLevel is two.
`-f`	Run in the foreground (i.e. don't become a daemon).
`-l logFile`	Specifies log file for errors and debug information.
`-s`	Use syslog file for errors.
`-v`	Display version.
`-z`	Compress the configuration file.

If you specify both the '`-d`' and '`-l logFile`' options, debug information will be printed into the log file.

Specifying the '`-d`' option will print debug information to stdout when the process is parsing the configuration file. Note that in order to see this debug information, you should run the process in the foreground (i.e. use the '`-f`' option).

The configuration of the udpRelayd process is performed via an XML (eXtensible Markup Language) configuration file. The udpRelayd process also supports runtime configuration via XML administration messages to a separate administration port. This admin port **only** accepts messages that are both encrypted and MAC'ed. The admin port is a UDP port that must be configured via the initial XML configuration file.

The udpRelayd process creates a shared memory segment to hold the configuration in memory.

The admin messages follow an XML-RPC structure, with the following XML tags:

- methodCall
- methodName
- params

The response will be an XML message with the following tags:

- methodResponse
- fault
- params
- param
- value

A C++ class has been created to provide easy access to administration requests. The C++ class, CUDPRelayAdmin, is defined in the CUDPRelayAdmin.hpp header file.

A CGI process has been created, using the CUDPRelayAdmin class, to provide a web interface for making administration requests. The process, udpRelayAdminCGI, uses HTML template files that it fills in with information to present to the user.

Relay to Relay Messages

The udpRelayd process adds a 24 byte header to messages that are sent between relays. The contents of the header are as follows as defined in udpRelay.hpp:

```
typedef struct _udpRelayMsgHdr_t {
     unsigned char       sync1;          /*  1 byte  */
     unsigned char       sync2;          /*  1 byte  */
     unsigned short      msgSize;        /*  2 bytes */
     unsigned short      dataSize;       /*  2 bytes */
     unsigned short      cntrl;          /*  2 bytes */
     unsigned short      msgType;        /*  2 bytes */
     unsigned char       paddingBytes;   /*  1 bytes */
     unsigned char       stationCfg;     /*  1 bytes */
     unsigned short      stationID;      /*  2 bytes */
     unsigned short      msgSeq;         /*  2 bytes */
     struct timeval      timestamp;      /*  8 bytes */
} udpRelayMsgHdr_t;                      /*----------*/
                                         /* 24 bytes */
```

Figure 3: udpRelayd Message Header

A description of each of the values is:

Table 3: udpRelayd Message Header

Value	Description
sync1	Sync byte for the start of messages. This is a constant whose value is 0x05.
sync2	Sync byte for the start of messages. This is a constant whose value is 0x64.
msgSize	This value indicates the total number of bytes for the message and is the sum of the actual data bytes, MAC bytes, and any padding bytes.
dataSize	This value indicates the total number of data bytes.
cntrl	This value is used to for inter-relay flags. Currently the only flag that is defined is for message acknowledgement (value = 0x01).
msgType	This is a relay independent message type. This value is used by the translation function.
paddingBytes	If the message is encrypted, the message data bytes must be aligned to eight bytes. This value indicates how many padding bytes were added to the data bytes.
stationCfg	Not currently used.
stationID	Source/destination specific station ID. The special value of 255 (0xff) is used to indicate that all stations should be sent to this destination connection.

Value	Description
msgSeq	A message sequence number that is incremented for each message.
timestamp	The current system time. This value is used by the relay for statistical calculations. The system time should be the same between all hosts running the udpRelayd process (e.g. UTC).

Relay Shared Memory

The udpRelayd process creates a shared memory segment to hold the current configuration. Other processes are able to attach to this shared memory to look at the configuration. Various statistics are also kept in the shared memory segment. The shared memory segment is divided into two areas. The first area contains a fixed number of connection group information (currently a maximum of 100). The second are contains a fixed number of connection information (currently a maximum of 100).

The udpReader utility can be used to read values from the shared memory segment. The following command line options are available:

Table 4: udpReader Command Line Options

Command Line Option	Description
-c configFile	Specifies the configuration file.
-k shMemKey	Specifies the shared memory key. The key can be listed using the UNIX command "ipcs". This can be used for looking at the shared memory segment that is created by the udpRelayAdminCGI process (see udpRelayAdminCGI Program).

Connection Group Shared Memory Area

The connection group shared memory area is at the start of the shared memory segment. The structure is:

```
typedef struct _udpRelayInfo_t {
    bool                inUse;          /* is being used        */
    char                name[15];       /* unique name          */
    unsigned int        connectionCnt;  /* # of connections     */
    /* array of connection offsets */
    int                 connections[UDP_RELAY_MAX_CONN];
    unsigned int        sendToCnt;      /* # of send to         */
    /* array of relay info offsets  */
    int                 sendTo[UDP_RELAY_MAX_SITES];
} udpRelayInfo_t;
```

Figure 4: Connection Group Shared Memory Structure

The offsets are all relative from each area.

Connection Shared Memory Area

The connection shared memory area is located at the end of connection group shared memory area. The structure is:

```
typedef struct _udpRelayConnInfo_t {
    unsigned short           stationID;      /* station ID           */
    unsigned short           port;           /* port to connect to   */
    unsigned short           seqNum;         /* sequnce number       */
    char                     hostname[57];   /* hostname             */
    unsigned char            ttl;            /* time to live         */
    bool                     connected;      /* connection flag      */
    bool                     multicast;      /* true=multi,false=udp */
    bool                     encrypt;        /* true=encrypt         */
    bool                     isRelay;        /* true=fwd w/msghdr    */
    bool                     enabled;        /* true=send            */
    bool                     translate;      /* true=call translate  */
    bool                     testing;        /* true=test mode       */
    unsigned char            testInterval;   /* seconds between tests*/
    unsigned char            retryCnt;       /* # of retries         */
    unsigned char            waitingOnAck;   /* cnt of acks e`xpected*/
    unsigned short           timeout;        /* timeout msecs.       */
    int                      sockSend;       /* send socket fd       */
    int                      sockRecv;       /* receive socket fd    */
    struct sockaddr_in       *peeraddr;      /* send socket address  */
    struct _udpRelayInfo_t   *relayInfo;     /* ptr. back to info    */
    struct _udpRelayConnInfo_t *secondary;   /* ptr. to redundant    */
    struct timeval           timeOfConnect;  /* time of connection   */
    unsigned long            pktsSent;       /* # of packets sent    */
    unsigned long            pktsRecv;       /* # of packets recv'ed */
    unsigned long            acksSent;       /* # of acks sent       */
    unsigned long            acksRecv;       /* # of acks recv'ed    */
    double                   minDelay;       /* min. delay (secs.)   */
    double                   maxDelay;       /* max. delay (secs.)   */
    double                   avgDelay;       /* avg. delay (secs.)   */
    string                   *getMsgTypeFunc;/* get msg type fnc name*/
    string                   *translateFunc; /* translate fnc name   */
    string                   *getStationIDFunc;/* get sta ID fnc name*/
    unsigned long            reserved;
    CUDPRelayCrypt           *cryptInfo;     /* encryption object    */
    unsigned char            heartbeatInt;   /* heartbeat (secs.)    */
    unsigned char            heartbeatCnt;   /* # heartbeat failures */
    unsigned char           heartbeatDis;   /* # of fails to discnt  */
    bool                     inUse;          /* is being used        */
} udpRelayConnInfo_t;
```

Figure 5: Connection Shared Memory Structure

Note: all pointer values will **not** be valid.

Configuration File

The udpRelayd configuration file is an XML formatted file with the following XML tags:

Table 5: Relay XML Configuration Tags

Tag	Description	Value	Comment
<udpRelay:Config>	Start of configuration		
<udpRelay:UseSysLog>	Use syslog (‘-s’)	True or False	
<udpRelay:LogFile>	Use logfile (‘-f’)	Full path to log file.	
<udpRelay:AdminPort>	Admin port	Number	0-65535
<udpRelay:MACPhrase>	MAC phrase	String	
<udpRelay:CryptPhrase>	Encryption phrase	String	
<udpRelay:ReconnectInt>	Re-connect interval (secs.)	Number	0-65535
<udpRelay:QueueLength>	Max. buffer queue length	Number	(0-255)
<udpRelay:ConnectionGroup>	Start of connection group configuration		Start of connection group
<udpRelay:Name>	Connection group name	String	
<udpRelay:Connection>	Start of connection configuration		
<udpRelay:StationID>	Connection station ID	Number	0-65535
<udpRelay:Hostname>	Connection hostname	string	IP address or hostname
<udpRelay:Port>	Connection port	Number	
<udpRelay:Multicast>	Connection uses multicast	True or False	
<udpRelay:Encrypt>	Connection uses encryption	True or False	
<udpRelay:Relay>	Connection is a relay	True or False	
<udpRelay:Enabled>	Connection is enabled	True or False	
<udpRelay:Translate>	Connection uses message translation	True or False	
<udpRelay:AckTimeout>	Connection ack timeout (msecs)	Number	0-65535
<udpRelay:TTL>	Connection time to live count	Number	0-255
<udpRelay:RetryCnt>	Connection ack retry count	Number	0-255
<udpRelay:GetMsgTypeFunc>	Connection get message type function name	Name of function	
<udpRelay:TranslateFunc>	Connection translate message function name	Name of function	
<udpRelay:GetStationIDFunc>	Get Station ID function name	Name of function	
<udpRelay:SendTo>	Connection group send to’s	Another connection group name	
<udpRelay:KeyExpire>	Encryption expiry (hours)	Number	0-65535
<udpRelay:HeartbeatInt>	Heartbeat interval (secs)	Number	0-255
<udpRelay:HBDisconnect>	Number of heartbeat failures before the connection will be disabled	Number	0-255
<udpRelay:Testing>	Testing flag	True or False	

Tag	Description	Value	Comment
<udpRelay:TestInterval>	Testing interval (secs)	Number	0-255

The XML configuration file consists of the following formatted structures: Config, ConnectionGroup, Connection, and SendTo. The Config structure is used to indicate the beginning and the end of the encapsulating structure. A ConnectionGroup will consist of a number of Connection(s) and SendTo structures.

Care must be taken to prevent port conflicts. Similarly all StationID's within the iGPSDR must be unique.

```
<?xml version="1.0"?>
<udpRelay:Config xmlns:udpRelay="http://www.tessernet.com/udpRelay/0.9.0/">

	<udpRelay:UseSysLog>False</udpRelay:UseSysLog>
	<udpRelay:LogFile>/tmp/udpRelay.log</udpRelay:LogFile>
	<udpRelay:AdminPort>23400</udpRelay:AdminPort>
	<udpRelay:MACPhrase>Admin port MAC phrase.</udpRelay:MACPhrase>
	<udpRelay:CryptPhrase>Admin port crypt phrase.</udpRelay:CryptPhrase>
	<udpRelay:ReconnectInt>10</udpRelay:ReconnectInt>

	<udpRelay:ConnectionGroup>
		<udpRelay:Name>ABC</udpRelay:Name>
		<udpRelay:Connection>
			<udpRelay:StationID>10</udpRelay:StationID>
			<udpRelay:Hostname>1.2.3.4</udpRelay:Hostname>
			<udpRelay:Port>23456</udpRelay:Port>
			<udpRelay:Multicast>False</udpRelay:Multicast>
			<udpRelay:Encrypt>True</udpRelay:Encrypt>
			<udpRelay:CryptPhrase>Some phrase</udpRelay:CryptPhrase>
			<udpRelay:MACPhrase>Some MAC phrase</udpRelay:MACPhrase>
			<udpRelay:Relay>True</udpRelay:Relay>
			<udpRelay:Enabled>True</udpRelay:Enabled>
			<udpRelay:Translate>False</udpRelay:Translate>
			<udpRelay:AckTimeout>1000</udpRelay:AckTimeout>
			<udpRelay:TTL>10</udpRelay:TTL>
			<udpRelay:RetryCnt>3</udpRelay:RetryCnt>
		<udpRelay:TranslateFunc>MyTranslateFunc</udpRelay:TranslateFunc>
		</udpRelay:Connection>

		<udpRelay:SendTo>DEF</udpRelay:SendTo>
		<udpRelay:SendTo>Ghi</udpRelay:SendTo>
		<udpRelay:SendTo>jkL</udpRelay:SendTo>
		<udpRelay:SendTo>MNo</udpRelay:SendTo>
	</udpRelay:ConnectionGroup>

	<udpRelay:ConnectionGroup>
		<udpRelay:Name>DEF</udpRelay:Name>
		<udpRelay:Connection>
			<udpRelay:StationID>10</udpRelay:StationID>
			<udpRelay:Hostname>10.1.2.3</udpRelay:Hostname>
			<udpRelay:Port>23458</udpRelay:Port>
			<udpRelay:Multicast>False</udpRelay:Multicast>
			<udpRelay:Relay>True</udpRelay:Relay>
			<udpRelay:Encrypt>False</udpRelay:Encrypt>
			<udpRelay:Enabled>True</udpRelay:Enabled>
			<udpRelay:Translate>False</udpRelay:Translate>
			<udpRelay:TTL>1</udpRelay:TTL>
			<udpRelay:RetryCnt>2</udpRelay:RetryCnt>
```

```
            <udpRelay:AckTimeout>500</udpRelay:AckTimeout>
      </udpRelay:Connection>
</udpRelay:ConnectionGroup>

<udpRelay:ConnectionGroup>
      <udpRelay:Name>Ghi</udpRelay:Name>
      <udpRelay:Connection>
            <udpRelay:StationID>10</udpRelay:StationID>
            <udpRelay:Hostname>2.3.4.5</udpRelay:Hostname>
            <udpRelay:Port>23458</udpRelay:Port>
            <udpRelay:Multicast>False</udpRelay:Multicast>
            <udpRelay:Relay>True</udpRelay:Relay>
            <udpRelay:Encrypt>False</udpRelay:Encrypt>
            <udpRelay:Enabled>True</udpRelay:Enabled>
            <udpRelay:Translate>False</udpRelay:Translate>
            <udpRelay:TTL>255</udpRelay:TTL>
            <udpRelay:RetryCnt>2</udpRelay:RetryCnt>
            <udpRelay:AckTimeout>500</udpRelay:AckTimeout>
      </udpRelay:Connection>
</udpRelay:ConnectionGroup>
<udpRelay:ConnectionGroup>
      <udpRelay:Name>jkL</udpRelay:Name>
      <udpRelay:Connection>
            <udpRelay:StationID>10</udpRelay:StationID>
            <udpRelay:Hostname>3.4.5.6</udpRelay:Hostname>
            <udpRelay:Port>23458</udpRelay:Port>
            <udpRelay:Multicast>False</udpRelay:Multicast>
            <udpRelay:Relay>True</udpRelay:Relay>
            <udpRelay:Encrypt>False</udpRelay:Encrypt>
            <udpRelay:Enabled>True</udpRelay:Enabled>
            <udpRelay:Translate>False</udpRelay:Translate>
            <udpRelay:TTL>255</udpRelay:TTL>
            <udpRelay:RetryCnt>2</udpRelay:RetryCnt>
            <udpRelay:AckTimeout>500</udpRelay:AckTimeout>
      </udpRelay:Connection>
</udpRelay:ConnectionGroup>
<udpRelay:ConnectionGroup>
      <udpRelay:Name>MNo</udpRelay:Name>
      <udpRelay:Connection>
            <udpRelay:StationID>10</udpRelay:StationID>
            <udpRelay:Hostname>4.5.6.7</udpRelay:Hostname>
            <udpRelay:Port>23458</udpRelay:Port>
            <udpRelay:Multicast>False</udpRelay:Multicast>
            <udpRelay:Relay>True</udpRelay:Relay>
            <udpRelay:Encrypt>False</udpRelay:Encrypt>
            <udpRelay:Enabled>True</udpRelay:Enabled>
            <udpRelay:Translate>False</udpRelay:Translate>
            <udpRelay:TTL>255</udpRelay:TTL>
            <udpRelay:RetryCnt>2</udpRelay:RetryCnt>
            <udpRelay:AckTimeout>500</udpRelay:AckTimeout>
      </udpRelay:Connection>
</udpRelay:ConnectionGroup>
<udpRelay:ConnectionGroup>
      <udpRelay:Name>pqr</udpRelay:Name>
      <udpRelay:Connection>
            <udpRelay:StationID>20</udpRelay:StationID>
            <udpRelay:Hostname>8.9.0.1</udpRelay:Hostname>
            <udpRelay:Port>23458</udpRelay:Port>
            <udpRelay:Multicast>False</udpRelay:Multicast>
            <udpRelay:Relay>False</udpRelay:Relay>
            <udpRelay:Encrypt>False</udpRelay:Encrypt>
            <udpRelay:Enabled>True</udpRelay:Enabled>
            <udpRelay:Translate>False</udpRelay:Translate>
```

```
                <udpRelay:TTL>255</udpRelay:TTL>
                <udpRelay:RetryCnt>2</udpRelay:RetryCnt>
                <udpRelay:AckTimeout>500</udpRelay:AckTimeout>
                <udpRelay:GetMsgTypeFunc>MyGetMsgType</udpRelay:GetMsgTpypeFunc>
        </udpRelay:Connection>

        <udpRelay:SendTo>STU</udpRelay:SendTo>
    </udpRelay:ConnectionGroup>
    <udpRelay:ConnectionGroup>
        <udpRelay:Name>STU</udpRelay:Name>
        <udpRelay:Connection>
                <udpRelay:StationID>20</udpRelay:StationID>
                <udpRelay:Hostname>23.4.5.6</udpRelay:Hostname>
                <udpRelay:Port>23458</udpRelay:Port>
                <udpRelay:Multicast>False</udpRelay:Multicast>
                <udpRelay:Relay>True</udpRelay:Relay>
                <udpRelay:Encrypt>False</udpRelay:Encrypt>
                <udpRelay:Enabled>True</udpRelay:Enabled>
                <udpRelay:Translate>False</udpRelay:Translate>
                <udpRelay:TTL>255</udpRelay:TTL>
                <udpRelay:RetryCnt>2</udpRelay:RetryCnt>
                <udpRelay:AckTimeout>500</udpRelay:AckTimeout>
        </udpRelay:Connection>
    </udpRelay:ConnectionGroup>
    <udpRelay:ConnectionGroup>
        <udpRelay:Name>VWx</udpRelay:Name>
        <udpRelay:Connection>
                <udpRelay:StationID>30</udpRelay:StationID>
                <udpRelay:Hostname>7.8.9.10</udpRelay:Hostname>
                <udpRelay:Port>23458</udpRelay:Port>
                <udpRelay:Multicast>False</udpRelay:Multicast>
                <udpRelay:Relay>False</udpRelay:Relay>
                <udpRelay:Encrypt>False</udpRelay:Encrypt>
                <udpRelay:Enabled>True</udpRelay:Enabled>
                <udpRelay:Translate>False</udpRelay:Translate>
                <udpRelay:TTL>255</udpRelay:TTL>
                <udpRelay:RetryCnt>2</udpRelay:RetryCnt>
                <udpRelay:AckTimeout>500</udpRelay:AckTimeout>
                <udpRelay:GetStationID>MyGetStaID</udpRelay:GetStationID>
        </udpRelay:Connection>

        <udpRelay:SendTo>YZ</udpRelay:SendTo>
    </udpRelay:ConnectionGroup>
    <udpRelay:ConnectionGroup>
        <udpRelay:Name>YZ</udpRelay:Name>
        <udpRelay:Connection>
                <udpRelay:StationID>30</udpRelay:StationID>
                <udpRelay:Hostname>11.2.3.4</udpRelay:Hostname>
                <udpRelay:Port>23458</udpRelay:Port>
                <udpRelay:Multicast>False</udpRelay:Multicast>
                <udpRelay:Relay>False</udpRelay:Relay>
                <udpRelay:Encrypt>False</udpRelay:Encrypt>
                <udpRelay:Enabled>True</udpRelay:Enabled>
                <udpRelay:Translate>False</udpRelay:Translate>
                <udpRelay:TTL>255</udpRelay:TTL>
                <udpRelay:RetryCnt>2</udpRelay:RetryCnt>
                <udpRelay:AckTimeout>500</udpRelay:AckTimeout>
        </udpRelay:Connection>
    </udpRelay:ConnectionGroup>
</udpRelay:Config>
```

Figure 6: Relay Configuration File Example

Some items to note:

1. The following line contains the udpRelayd version number:

```
<udpRelay:Config xmlns:udpRelay="http://www.tessernet.com/udpRelay/0.9.0/">
```

   and **must** match the current udpRelayd version number (e.g. `0.9.0`). If it doesn't match, the udpRelayd will fail to start.

2. The `<udpRelay:MACPhrase>` and `<udpRelay:CryptPhrase>` tags can be used both within the Config and Connection scopes. When used within the Config scope, the MAC and encryption phrases will be used for the administration port messages. When used within the Connection scope, the MAC and encryption phrases will be used for that specific Station ID's connection messages. If the MAC and encryption phrase is specified for a connection, the public/private key will **not** be exchanged with the connection.

3. Messages arriving from connection group "ABC" will be sent to connection groups "DEF", "Ghi", "jkL', and "MNo". Note that the Station ID configured for receipt hosts is **not** looked at. The Station ID is **only** used for outgoing messages. Therefore, only Station ID 10's data will be sent to the corresponding Connections specified for the above send to connection groups (since this is the only Station ID configured at each of these connection groups). If a message arrives with a Station ID not equal to ten, it will be silently dropped (unless debugging has been enabled). If all data is to be sent to a connection, use the special Station ID of 255. Also note that all messages received from connection group "ABC" will be translated using the "MyTranslateFunc" function **before** it is transmitted to the send to connection groups.

4. Data received from the "pqr" connection group, will be marked with a Station ID of 20 and relayed to connection group "STU". Because connection group "pqr" is **not** a relay (i.e. `<udpRelay:Relay>False</udpRelay:Relay>`) and connection group "STU" **is** a relay; the udpRelayd process will try to extract a message type by calling the function defined in the "GetMsgTypeFunc" tag (i.e. MyGetMsgType). As part of the "GetMsgTypeFunc" function's return values, it also indicates what "cntrl" flags should be set (see

Relay to Relay Messages); which currently indicates whether the message requires acknowledgement or not.

5. Data received from the "VWx" connection group will be relayed to connection group "YZ". Because connection group "VWx" is **not** a relay (i.e. `<udpRelay:Relay>False</udpRelay:Relay>`) and a "GetStationIDFunc" function was defined, udpRelayd will call this function to get the Station ID.

If the "Testing" tag is set to true, the udpRelayd will send a 1024 byte test message every "TestInterval" seconds to this connection.

If the message indicates that an acknowledgement is required, the udpRelayd will wait "AckTimeout" milli-seconds for the acknowledgement. If the acknowledgement is not received, it will attempt to re-send the message "RetryCnt" times before it will give up.

The reconnect interval (ReconnectInt), specifies how often failed connections will try re-connecting.

If a heartbeat interval is specified (HeartbeatInt), a header message will be sent to the connection. The heartbeat acknowledgement is expected to be received within the acknowledgement timeout (AckTimeout). If a heartbeat disconnect value (HBDisconnect) is specified and no heartbeat acknowledgement has been received for the specified number of heartbeat intervals, the connection will be disabled (and disconnected).

Note that "KeyExpire" has not been implemented yet.

Relay Functions

The udpRelayd can be customized by writing various "functions". These extension functions can be used to:

- Determine the message type of a source message and whether the message requires an acknowledgement.
- Determine the station ID of a source message.
- Translate a message before it is relayed to a destination.
- Handle custom administration requests.

The function prototypes for the get message type, get station ID, and translate message functions are defined in `udpRelayFuncs.h`. Creating these functions requires the following steps:

1. Write the function using the defined prototype: `UDP_RELAY_GET_MSG_FUNC_ARGS` for get message type functions, `UDP_RELAY_TRANS_FUNC_ARGS` for translate

message functions, and `UDP_RELAY_GET_STA_ID_FUNC_ARGS` for get station ID functions.

2. Add the function prototype into "`funcDefs.cpp`".
3. Add the function into the appropriate global array: `g_getMsgFuncs` for get message functions, `g_translateFuncs` for translate message functions, and `g_getStationIDFuncs` for get station ID functions.
4. Re-make udpRelayd.

A future enhancement is to load these functions through a shared library.

Get Message Type Function

The get message type functions are called when a message is received from a connection that is **not** a relay connection and is being relayed to a connection that **is** a relay connection.

The function prototype is:

```
#define       UDP_RELAY_GET_MSG_FUNC_ARGS     unsigned char  *msg,     \
                                              unsigned short *cntrl,   \
                                              unsigned short *msgSize
```

The function should take the incoming message (msg) and return an integer value. The integer value should be either:

- A value less than zero to indicate an error has occurred.
- A value of zero to indicate that this message should be ignored.
- A value greater than zero to indicate the message type that should be placed in the header.

The function can modify the outgoing message by saving the modified message back into the message pointer (msg). It will be guaranteed to have at least enough buffer space for up to 65535 bytes. The message size (msgSize) is both the incoming message size and if the message is modified, the resulting modified message size.

The function **must** set the control flags (cntrl) to zero upon function entry and set this value to `UDP_RELAY_CNTRL_ACK` if the message should be acknowledged by the receiving relay.

Get Station ID Function

The get station ID functions are called when a message is received from a connection that has a get station ID function defined.

The function prototype is:

```
#define         UDP_RELAY_GET_STA_ID_FUNC_ARGS  unsigned char  *msg
```

The function should take the incoming message (msg) and return an integer value. The integer value should be either:

- A value less than zero to indicate an error has occurred.
- A value of zero to indicate that this message should be ignored.
- A value greater than zero to indicate the station ID that should be placed in the header.

Translate Message Function

The translate message functions are called when the receiving connection has the translate flag set.

The function prototype is:

```
#define       UDP_RELAY_TRANS_FUNC_ARGS       unsigned char  *msg,    \
                                              unsigned short msgType, \
                                              unsigned short msgSize
```

The function should take the incoming message (msg) and return an integer value. The integer value should be either:

- A value less than zero to indicate an error has occurred.
- A value of zero to indicate that this message should be ignored.
- A value greater than zero to indicate the message type that should be placed in the header.

The function should translate the message by saving the modified message back into the message pointer (msg). It will be guaranteed to have at least enough buffer space for up to 65535 bytes. The message size (msgSize) is both the incoming message size and the resulting translated message size.

Custom Administration Request Hook Function

The custom administration request hook function is called when a verified administration request (see Administration Requests) is received that is not understood by the udpRelayd. The hook function **must** be written in C++. The actual function called will be:

```
extern int udpRelayAdminReqHook (string                 &methodName,
                                 map<string, string>    &paramMap,
                                 string                 &respMsg);
```

The “methodName” parameter is a string containing the method name that was in the administration request.

The “paramMap” parameter is a string to string map containing a list of all parameters in the administration request (i.e. values defined in the “params” tag).

The “respMsg” parameter should be set by the hook function to contain a string that will be returned to the calling client as the error string.

The function should return either:

- A value less than zero to indicate an error has occurred.
- A value of zero or greater to indicate that the administration request was successfully handled. This value will be returned to the calling client as part of the response.

Administration Requests

All valid administration messages received by the relay will be logged (as long as a log method has been configured). udpRelayd admin requests are XML formatted messages with the following format:

```
<?xml version=”1.0”>
<methodCall>
     <methodName>
     </methodName>
     <params>
     </params>
</methodCall>
```

Valid values for the “methodName” tag are:

- AddSite
- DeleteSite
- DeleteSendTo
- AddConnection
- DeleteConnection
- ModifyConnection
- QueryConfig

Depending on the method used, additional parameters may be required. These additional parameters are specified within the “params” tag.

AddSite Admin Request

The “AddSite” method is used to add a new connection group to the relay. Valid “params” tags are:

- Name – specifies the name of the new connection group. It is an error to try and add a connection group that already exists.

Example:

```
<?xml version=”1.0”>
<methodCall>
     <methodName>AddSite</methodName>
     <params>
          <Name>tesserNet</Name>
     </params>
</methodCall>
```

DeleteSite Admin Request

The “DeleteSite” method is used to delete and existing connection group. All connections defined for the specified connection group will also be deleted. All send to(s) will also be cleaned up. Valid “params” tags are:

- Name – specifies the name of the connection group to delete.

Example:

```
<?xml version=”1.0”>
<methodCall>
     <methodName>DeleteSite</methodName>
     <params>
          <Name>tesserNet</Name>
     </params>
</methodCall>
```

AddSendTo Admin Request

The “AddSendTo” method is used to add a “send to” to an existing connection group. Valid “params” tags are:

- Site – specifies the name of the connection group to add the “send to” to.
- SendTo – specifies the name of the connection group to “send to”.

Example:

```
<?xml version=”1.0”>
<methodCall>
     <methodName>AddSendTo</methodName>
     <params>
          <Site>tesserNet</Site>
          <SendTo>NRCan</SendTo>
     </params>
</methodCall>
```

DeleteSendTo Admin Request

The “DeleteSendTo” method is used to delete an existing “send to” from an existing connection group. Valid “params” tags are:

- Site – specifies the name of the connection group to delete the “send to” from.
- SendTo – specifies the name of the connection group to delete from the “send to”.

Example:

```
<?xml version=”1.0”>
<methodCall>
     <methodName>DeleteSendTo</methodName>
     <params>
          <Site>tesserNet</Site>
          <SendTo>NRCan</SendTo>
     </params>
</methodCall>
```

AddConnection Admin Request

The “AddConnection” method is used to add a new connection to an existing connection group. Valid “params” tags are:

- Site – specifies the name of the connection group to add the connection to.
- StationID – specifies the station ID for the new connection (0-65535).
- Hostname – specifies the hostname/IP address for the new connection.
- Port – specifies the port number for the new connection.
- Multicast – specifies whether the connection is a multicast connection or not (True/False).
- Encrypt – specifies whether the connection is encrypted or not (True/False).
- CryptPhrase – specifies the encryption phrase for the new connection.
- Relay – specifies whether the connection is a relay or not (True/False).
- Enabled – specifies whether the connection is enabled or not (True/False).
- Translate – specifies whether the connection requires translation or not (True/False).
- AckTimeout – specifies the acknowledgement timeout (0-65535 milli-seconds).
- TTL – specifies the Time To Live value for the connection (0-255).
- RetryCnt – specifies the retry count for the connection (0-255).
- GetMsgTypeFunc – specifies the name of the function to call to retrieve a message type.
- TranslateFunc – specifies the name of the function to call to translate a message.
- GetStationID – specifies the name of the function to call to retrieve the station ID from a message.

Example:

```
<?xml version="1.0">
<methodCall>
     <methodName>AddConnection</methodName>
     <params>
          <Site>tesserNet</Site>
          <StationID>10</StationID>
          <Hostname>abc.domain.com</Hostname>
          <Port>23456</Port>
          <Multicast>False</Multicast>
          <Encrypt>True</Encrypt>
          <CryptPhrase>Some phrase for abc…</CryptPhrase>
          <Relay>True</Relay>
          <Enabled>True</Enabled>
          <Translate>False</Translate>
          <AckTimeout>1000</AckTimeout>
          <TTL>10</TTL>
          <RetryCnt>3</RetryCnt>
          <GetMsgTypeFunc>NRCGetMsgType</GetMsgTypeFunc>
     </params>
</methodCall>
```

DeleteConnection Admin Request

The "DeleteConnection" method is used to delete an existing connection from an existing connection group. Valid "params" tags are:

- Site – specifies the name of the connection group to delete the connection from.
- StationID – specifies the station ID of the connection to delete.

Example:

```
<?xml version="1.0">
<methodCall>
     <methodName>DeleteConnection</methodName>
     <params>
          <Site>tesserNet</Site>
          <StationID>10</StationID>
     </params>
</methodCall>
```

ModifyConnection Admin Request

The "ModifyConnection" method is used to make changes to an existing connection. Valid "params" tags are:

- Site – specifies the name of the connection group the contains the connection to modify.
- StationID – specifies the station ID to modify.
- Encrypt – specifies a new value for the connection's encryption (True/False).
- CryptPhrase – specifies the new encryption phrase.
- Relay – specifies a new value for the connection's relay (True/False).
- Enabled – specifies a new value for the connection's enable (True/False).
- Translate – specifies a new value for the connection's translate (True/False).
- AckTimeout – specifies a new value for the acknowledgement timeout (0-65535 milli-seconds).
- TTL – specifies a new value for the connection's time to live (0-255).
- RetryCnt – specifies a new value for the connection's retry count (0-255).
- GetMsgTypeFunc – specifies a new function name for the get message type.
- TranslateFunc – specifies a new function name for the translation.
- GetStationID – specifies a new function name for the get station ID.

Note that the following connection values can **NOT** be changed once it has been created:

- Port

- Hostname
- Multicast

If any of these values need to be changed, the connection should be deleted and re-created.

Example:

```
<?xml version="1.0">
<methodCall>
     <methodName>ModifyConnection</methodName>
     <params>
          <Site>tesserNet</Site>
          <StationID>10</StationID>
          <Encrypt>False</Encrypt>
     </params>
</methodCall>
```

QueryConfig Admin Request

The "QueryConfig" method is used to query for the relay's current configuration. There are no additional "params" tags required. The relay will send back the current configuration (in XML format) if successful.

Example:

```
<?xml version="1.0">
<methodCall>
     <methodName>QueryConfig</methodName>
</methodCall>
```

Admin Responses

udpRelay admin responses are XML formatted messages with the following two possible formats:

```
<?xml version="1.0">
<methodResponse>
     <params>
          <param>
               <value></value>
          </param>
     </params>
```

```
</methodResponse>
```

The “value” returned is the return code from the method (usually zero).

or

```
<?xml version=”1.0”>
<methodResponse>
     <fault>
          <param>
               <value></value>
          </param>
     </fault>
</methodResponse>
```

The “value” returned is a string indicating the reason for the failure.

The response will only be sent back to the client if a valid message was received. If a message was received that could not be un-encrypted or if the MAC authentication fails, no response will be sent back. The client should have a timeout value for a response.

The admin response will be sent back for all method requests except for the “QueryConfig” where the configuration will be sent back.

CUDPRelayAdmin C++ Class

The CUDPRelayAdmin C++ class provides a layer of abstraction for relay administration requests. The class is defined as:

```
class CUDPRelayAdmin {

  public:

    /* The CUDPRelayAdminException class. */
    class CUDPRelayAdminException {

      public:
        CUDPRelayAdminException (string &str) {errStr = str;}

        string          errStr;
    };

    CUDPRelayAdmin (const string &hostIP, const string &servName,
                    const string &cryptPhrase, const string &macPhrase,
                    unsigned int replyTimeout);
    ~CUDPRelayAdmin ();

    virtual string &    GetErrStr () {return (m_errStr);}
```

```
virtual bool        AddSite (string &siteName);
virtual bool        DeleteSite (string &siteName);

virtual bool        AddSendTo (string &siteName,
                               string &sendToName);
virtual bool        DeleteSendTo (string &siteName,
                                  string &sendToName);
virtual bool        AddConnection (string           &siteName,
                                   unsigned short   stationID,
                                   string           &hostName,
                                   unsigned short   port,
                                   bool             isEnabled,
                                   bool             isRelay,
                                   bool             multicast,
                                   bool             encrypt,
                                   bool             translate,
                                   bool             testing,
                                   string           &cryptPhrase,
                                   string           &macPhrase,
                                   string           &transFunc,
                                   string           &msgTypeFunc,
                                   string           &getStaIDFunc,
                                   unsigned short   ackTimeout =0,
                                   unsigned char    retryCnt =0,
                                   unsigned char    heartbeatInt=0,
                                   unsigned char    testInt =0,
                                   unsigned char    ttl =255);
virtual bool        AddConnection (string           &siteName,
                                   unsigned short   stationID,
                                   string           &hostName,
                                   string           &port,
                                   bool             isEnabled,
                                   bool             isRelay,
                                   bool             multicast,
                                   bool             encrypt,
                                   bool             translate,
                                   bool             testing,
                                   string           &cryptPhrase,
                                   string           &macPhrase,
                                   string           &transFunc,
                                   string           &msgTypeFunc,
                                   string           &getStaIDFunc,
                                   unsigned short   ackTimeout =0,
                                   unsigned char    retryCnt =0,
                                   unsigned char    heartbeatInt=0,
                                   unsigned char    testInt =0,
                                   unsigned char    ttl =255);
virtual bool        DeleteConnection (string &siteName,
                                      unsigned short stationID);
virtual bool        DeleteConnection (string &siteName,
                                      string &stationID);

virtual bool        QueryConfig (string &config);

virtual void        ModifyConnectionEnabled (bool isEnabled);
```

```
    virtual void        ModifyConnectionRelay (bool isRelay);
    virtual void        ModifyConnectionMulticast (bool isMulticast);
    virtual void        ModifyConnectionEncrypt (bool isEncrypted);
    virtual void        ModifyConnectionTesting (bool isTesting);
    virtual void        ModifyConnectionTranslate (bool isTranslating);
    virtual void        ModifyConnectionCryptPhrase (
                                          string &cryptPhrase);
    virtual void        ModifyConnectionMACPhrase (string &macPhrase);
    virtual void        ModifyConnectionTransFunc (string &transFunc);
    virtual void        ModifyConnectionMsgTypeFunc (
                                 string &msgTypeFunc);
    virtual void        ModifyConnectionGetStaIDFunc (
                                 string &getStaIDFunc);
    virtual void        ModifyConnectionAckTimeout (
                               unsigned short ackTimeout);
    virtual void        ModifyConnectionRetryCnt (
                               unsigned char retryCnt);
    virtual void        ModifyConnectionHeartbeatInt (
                               unsigned char heartbeatInt);
    virtual void        ModifyConnectionTestInt (
                               unsigned char testInt);
    virtual void        ModifyConnectionTTL (unsigned char ttl);
    virtual bool        ModifyConnection (string &siteName,
                                 unsigned short stationID);

  protected:
    bool                _addEncryptionMAC (unsigned char *msg,
                                 int *msgLength);
    bool                _decryptMAC (unsigned char *msg,
                               int *msgLength);
    bool                _recvResp ();
    bool                _sendIt (unsigned char *reqMsg, int &reqSize);

    map<string, string> _modConnMap;

    string              m_errStr;
    string              m_cryptPhrase;
    string              m_macPhrase;
    CUDPRelayCrypt      *m_encryptInfo;
    CUDPRelayCrypt      *m_macInfo;
    int                 m_cryptBlkSize;
    int                 m_macSize;
    struct sockaddr_in  m_peeraddrIn;
    int                 m_sock;
    int                 m_replyTimeout;
    unsigned char       m_reqMsg[1024];
    char                m_recvBuffer[65535];
};
```

In order to create an instance of, the user is required to know the following information for the relay:

- host name/IP address,
- port number,
- encryption phrase,

- MAC phrase.

Different methods within the class provide access to all of the administration requests. The class handles the encryption, MAC, sending and response receipt.

For the "ModifyConnection" method, make calls to the other "ModifyConnection…" methods to first setup any tags that require modification before calling the "ModifyConnection" method to send the request. For example,

```
CUDPRelayAdmin          *relayAdmin =
      new CUDPRelayAdmin ("abc.domain.com", "23456",
                          "Some encryption phrase…",
                          "Some MAC phrase…", 10);

…

relayAdmin->ModifyConnectionEnabled (true);
relayAdmin->ModifyConnectionRetryCnt (5);

if (relayAdmin->ModifyConnection ("tesser", 10)) {
      …
}
else {
      …
}

…
```

Each of the request methods return true if the request was successful; otherwise they return false indicating a failure. The failure string can be retrieved using the "GetErrStr" method.

udpRelayAdminCGI Program

udpRelayAdminCGI is a CGI (Common Gateway Interface) program used to provide a web interface for relay administration requests. This program uses HTML template files, which it fills out with information to present to the user.

The HTML template files are searched for in the following directory:

```
/opt/udpRelay/html
```

This can be over-ridden by the "`UDPRELAY_TEMPLATEDIR`" environment variable. Note that this variable will need to be set by the web server.

The udpRelayAdminCGI program uses the GNU cgicc library for CGI access and the Template2doc library (by Rikard Thunberg) for HTML templating. The Template2doc

library requires that any repeating pieces of information be enclosed in "markers" and the template repeated twice.

Each time udpRelayAdminCGI is called, it must be provided the following set of CGI values as a minimum (usually provided as a hidden field in the form):

- `HOSTNAME` – specifies the hostname/IP address of the relay to send the admin request to,
- `PORT` – specifies the port number of the relay to send the admin request to,
- `CRYPT_PHRASE` – specifies the encryption phrase for the admin request,
- `MAC_PHRASE` – specifies the MAC phrase for the admin request,
- `ACTION` – specifies the admin request to make. Valid values are:
    - `CONNECT` – requests a connection to the relay and to retrieve it's current configuration (Note that some requests will automatically issue a re-connection request to update the configuration).
    - `ADD_SITE` – requests a connection group to be added.
    - `DEL_SITE` – request a connection group to be deleted.
    - `ADD_CONN` – request a new connection to be added to a connection group.
    - `DEL_CONN` – request a connection to be deleted from a connection group.
    - `MOD_CONN` – request a connection to be modified.
    - `SITE_INFO` – request connection group information to be filled in.
    - `FILL_TMPT` – request a generic template file to be filled in with some template information.
    - `CONN_INFO` – request connection information to be filled in.
    - `SENDTO_INFO` – request send to information to be filled in.
    - `ADD_SENDTO` – request a "send to" to be added to a connection group.
    - `DEL_SENDTO` – request a "send to" to be deleted from a connection group.

Each action may require additional CGI values.

The udpRelayAdminCGI program tries to be language sensitive by searching for template files with the current user language appended to it. If it is unable to find the template file for the language, it will default back to English (en). The following template files will be loaded by udpRelayAdminCGI:

- `udpRelayErr.html`
- `udpRelayInvReq.html`
- `udpRelayMissing.html`
- `udpRelayConnect.html`
- `udpRelaySite.html`
- `udpRelayModConn.html`
- `udpRelayAddConn.html`
- `udpRelaySendTo.html`

Example template files are provided for the English (en) language (i.e. udpRelayConnect.html.en).

Each template file contains variables and markers that indicate where information should be placed by the udpRelayAdminCGI program. The first line of each file **must** indicate the content:

```
Content-Type: text/html
```

Each file provides a different set of variables/markers. All variables/markers are proceeded by the '$' character.

Template HTML Files

The following section describes each of the HTML template files, what they are used for and what variables and/or markers are supported.

udpRelayErr.html

This HTML template file is used whenever an error has been detected (either by the CUDPRelayAdmin request or by the response). The following variable is used by this template file:

- `$error` – replaced with the error string (English only).

udpRelayInvReq.html

This HTML template file is used when an invalid CGI ACTION value has been received. The following variable is used by this template file:

- `$invRequest` – replaced with the ACTION value.

udpRelayMissing.html

This HTML template file is used when a required CGI variable was NOT provided as part of the request. The following variable is used by this template file:

- `$elementName` – replaced by the missing CGI variable name.

udpRelayConnect.html

This HTML template file is used to display connection group information that is returned from the initial connection query request. The following variable are used by this template file:

- `$hostname` – replaced by the hostname/IP address of the relay to send the admin request to.
- `$port` – replaced by the port number of the relay to send the admin request to.
- `$cryptPhrase` – replaced by the encryption phrase for the admin request.
- `$macPhrase` – replaced by the MAC phrase for the admin request.
- `$site` – replaced by the connection group name. Note that this place holder should be put into a table so that each connection group can be listed (one connection group for each table record).
- `$sName1` – same as `$site`.
- `$sName2` – same as `$site`.
- `$sName3` – same as `$site`.

The following marker is used by this template for the (`$site`, `$sName1`, `$sName2`, and `$sName3` variables):

- `$table`

udpRelaySite.html

This HTML template file is used to display connection information for a specific connection group. The following variables are used by this template file:

- `$connStationID` – replaced with the connection's station ID.
- `$connEnabled` – replaced with zero or one indicating whether the connection is enabled or not.
- `$connHostname` – replaced with the connection's hostname/IP address.
- `$connPort` – replaced with the connection's port.
- `$connMulticast` – replaced with zero or one indicating whether the connection is using UDP Multicast or UDP Unicast.
- `$connEncrypt` – replaced with zero or one indicating whether the connection requires encryption or not.
- `$connCryptPhrase` – replaced with the connection's encryption phrase.
- `$connMACPhrase` – replaced with the connection's MAC phrase.

- $connRelay – replaced with zero or one indicating whether the connection is a relay or not.
- $connTranslate – replaced with zero or one indicating whether the connection requires translation or not.
- $connAckTimeout – replaced with the connection's acknowledgement timeout value.
- $connTtl – replaced with the connection's time to live value.
- $connRetryCnt – replaced with the connection's retry count.
- $connHBInt – replaced with the connection's heartbeat interval.
- $connGetMsgTypeFunc – replaced with the connection's get message type function.
- $connTransFunc – replaced with the connection's translate message function.
- $connGetStationIDFunc – replaced with the connection's get station ID function.

- $hostname – replaced by the hostname/IP address of the relay to send the admin request to.
- $port – replaced by the port number of the relay to send the admin request to.
- $cryptPhrase – replaced by the encryption phrase for the admin request.
- $macPhrase – replaced by the MAC phrase for the admin request.
- $site – replaced by the current connection group's name.

The following marker is used by this template for all of the "$conn*" variables:

- $conn_table

udpRelayModConn.html

This HTML template file is used to display a specific connection's information for modification. The following variables are used by this template file:

- $conn – replaced by the connection's station ID.
- $connEnabled – replaced with "CHECKED" if the connection is enabled.
- $connDisabled – replaced with "CHECKED" if the connection is disabled.
- $connHostname – replaced with the connection's hostname/IP address.
- $connPort – replaced with the connection's port.
- $connMulticast – replaced with zero or one indicating whether the connection is using UDP Multicast or UDP Unicast.
- $connDoEncrypt – replaced with "CHECKED" if the connection requires encryption.
- $connNoEncrypt – replaced with "CHECKED" if the connection does not require encryption.

- $connCryptPhrase – replaced with the connection's encryption phrase.
- $connMACPhrase – replaced with the connection's MAC phrase.
- $connRelay – replaced with zero or one indicating whether the connection is a relay or not.
- $connDoTranslate – replaced with "CHECKED" if the connection requires translation.
- $connNoTranslate – replaced with "CHECKED" if the connection does not require translation.
- $connDoGetMsg – replaced with "CHECKED" if the connection has a get message type function.
- $connNoGetMsg – replaced with "CHECKED" if the connection does not have a get message type function.
- $connAckTimeout – replaced with the connection's acknowledgement timeout value.
- $connTtl – replaced with the connection's time to live value.
- $connRetryCnt – replaced with the connection's retry count.
- $connHBInt – replaced with the connection's heartbeat interval.

- $getmsg_func – replaced with a list of get message type function(s).
- $getmsg_selected – replaced with "SELECTED" if the current get message type function should be selected.
- $trans_func – replaced with a list of translate message function(s).
- $trans_selected – replaced with "SELECTED" if the current translate message function should be selected.
- $staid_func – replaced with a list of get station ID function(s).
- $staid_selected – replaced with "SELECTED" if the current get station ID function should be selected.

- $hostname – replaced by the hostname/IP address of the relay to send the admin request to.
- $port – replaced by the port number of the relay to send the admin request to.
- $cryptPhrase – replaced by the encryption phrase for the admin request.
- $macPhrase – replaced by the MAC phrase for the admin request.
- $site – replaced by the current connection group's name.

The following marker is used by this template for the $getmsg_func and $getmsg_selected variables:

- $getmsg_list

The following marker is used by this template for the $trans_func and $trans_selected variables:

- $trans_list

The following marker is used by this template for the $staid_func and $staid_selected variables:

- $staid_list

udpRelayAddConn.html

This HTML template file is used to add a new connection for a connection group. The following variables are used by this template file:

- $hostname – replaced by the hostname/IP address of the relay to send the admin request to.
- $port – replaced by the port number of the relay to send the admin request to.
- $cryptPhrase – replaced by the encryption phrase for the admin request.
- $macPhrase – replaced by the MAC phrase for the admin request.
- $site – replaced by the connection group name. Note that this place holder should be put into a table so that each connection group can be listed (one connection group for each table record).

- $getmsg_func – replaced with a list of get message type function(s).
- $trans_func – replaced with a list of translate message function(s).
- $staid_func – replaced with a list of get station ID function(s).

The following marker is used by this template for the $getmsg_func variable:

- $getmsg_list

The following marker is used by this template for the $trans_func variable:

- $trans_list

The following marker is used by this template for the $staid_func variable:

- $staid_list

udpRelaySendTo.html

This HTML template file is used to display a connection group's send to information. The following variables are used by this template file:

- `$hostname` – replaced by the hostname/IP address of the relay to send the admin request to.
- `$port` – replaced by the port number of the relay to send the admin request to.
- `$cryptPhrase` – replaced by the encryption phrase for the admin request.
- `$macPhrase` – replaced by the MAC phrase for the admin request.
- `$site` – replaced by the current connection group's name.

- `$sendto_site1` – replaced by the name of the send to connection group name.
- `$sendto_site2` – same as `$sendto_site1`.
- `$sendto_site` – replaced with a list of send to connection group(s).

The following marker is used by this template for the `$sendto_site1` and `$sendto_site2` variables:

- `$sendtos_table`

The following marker is used by this template for the `$sendto_site` variable:

- `$sendto_list`

Template Actions

The following section describes each of the HTML template form actions, what they are used for and what CGI values are required.

CONNECT Action

The CONNECT action is used to initially connect to the udpRelayd and request the latest configuration information.

This action does not require and additional CGI values.

The udpRelayAdminCGI program will fill in the "`udpRelayConnect.html`" template file for display by the client.

ADD_SITE Action

The ADD_SITE action is used to add a new connection group to the udpRelayd. The following CGI value is required by this action:

- SITE – specifies the new connection group name.

The udpRelayAdminCGI program will call run the CONNECT Action if the connection group was successfully added.

DEL_SITE Action

The DEL_SITE action is used to delete an existing connection group. The following CGI value is required by this action:

- SITE – specifies the connection group name to delete.

The udpRelayAdminCGI program will call run the CONNECT Action if the connection group was successfully deleted.

ADD_CONN Action

The ADD_CONN action is used to add a new connection to an existing connection group. The following CGI values are required by this action:

- SITE – specifies the connection group name where this connection should be added.
- CONN – specifies the station ID for the new connection.
- NEW_HOSTNAME – specifies the hostname/IP address for the new connection.
- NEW_PORT – specifies the port number for the new connection.
- NEW_ENABLED – specifies whether the new connection is enabled or not (the value should be either "1" or "on").
- NEW_IS_RELAY – specifies whether the new connection is a relay or not (the value should be either "1" or "on").
- NEW_MULTICAST – specifies whether the new connection is a multicast connection or a unicast connection (the value should be either "1" or "on").
- NEW_ENCRYPT – specifies whether the new connection requires encryption or not (the value should be either "1" or "on").
- NEW_CRYPT_PHRASE – specifies the connection's encryption phrase for the new connection.
- NEW_MAC_PHRASE – specifies the connection's MAC phrase for the new connection.
- NEW_ACK_TIMEOUT – specifies the acknowledgement timeout value for the new connection.
- NEW_RETRY_CNT – specifies the retry count for the new connection.
- NEW_HB_INTERVAL – specifies the heartbeat interval for the new connection.

- NEW_TTL – specifies the time to live value for the new connection.
- NEW_TRANSLATE – specifies whether the connection requires translation or not (value should be "1" or "on").
- NEW_TRANS_FUNC – specifies the translation function name for the new connection.
- NEW_GET_MSG – specifies whether the connection requires a get message type function or not (value should be "1" or "on").
- NEW_GET_MSG_FUNC – specifies the new get message function name for the new connection.
- NEW_TESTING – specifies whether the connection is in testing mode or not (value should be either "1" or "on").
- NEW_TEST_INT – specifies the testing interval for the new connection.
- NEW_GET_STA_ID_FUNC – specifies the get station ID function name for the new connection.

The udpRelayAdminCGI program will call run the SITE_INFO Action if the connection was successfully added.

DEL_CONN Action

The DEL_CONN action is used to delete an existing connection from a connection group. The following CGI values are required by this action:

- SITE – specifies the connection group name where this connection should be deleted.
- CONN – specifies the station ID of the connection to be deleted.

The udpRelayAdminCGI program will call run the SITE_INFO Action if the connection was successfully added.

MOD_CONN Action

The MOD_CONN action is used to modify an existing connection from a connection group. The following CGI values are required by this action:

- SITE – specifies the connection group name where this connection should be modified.
- CONN – specifies the station ID for the connection.
- NEW_ENABLED – specifies whether the connection should be enabled or not (the value should be either "1" or "on").
- NEW_IS_RELAY – specifies whether the connection is a relay or not (the value should be either "1" or "on").

- NEW_ENCRYPT – specifies whether the connection should provide encryption or not (the value should be either "1" or "on").
- NEW_CRYPT_PHRASE – specifies the connection's encryption phrase for the connection.
- NEW_MAC_PHRASE – specifies the connection's MAC phrase for the connection.
- NEW_ACK_TIMEOUT – specifies the acknowledgement timeout value for the connection.
- NEW_RETRY_CNT – specifies the retry count for the connection.
- NEW_HB_INTERVAL – specifies the heartbeat interval for the connection.
- NEW_TTL – specifies the time to live value for the connection.
- NEW_TRANSLATE – specifies whether the connection requires translation or not (value should be "1" or "on").
- NEW_TRANS_FUNC – specifies the translation function name for the connection.
- NEW_GET_MSG – specifies whether the connection requires a get message type function or not (value should be "1" or "on").
- NEW_GET_MSG_FUNC – specifies the new get message function name for the connection.
- NEW_TESTING – specifies whether the connection is in testing mode or not (value should be either "1" or "on").
- NEW_TEST_INT – specifies the testing interval for the connection.
- NEW_GET_STA_ID_FUNC – specifies the get station ID function name for the connection.

The udpRelayAdminCGI program will call run the SITE_INFO Action if the connection was successfully added.

SITE_INFO Action

The SITE_INFO action is used to display the connections for a specific connection group. The following CGI value is required by this action:

- SITE – specifies the connection group name where this connection should be modified.

The udpRelayAdminCGI program will fill in the "udpRelaySite.html" template file for display by the client.

CONN_INFO Action

The CONN_INFO action is used to fill in specific connection information. The following CGI values are required by this action:

- SITE – specifies the connection group name for this connection.
- CONN – specifies the station ID for this connection.

If the "CONN" CGI value is provided and is **not** zero, the udpRelayAdminCGI program will fill in the "udpRelayModConn.html" template file for display by the client; otherwise it will fill in the "udpRelayAddConn.html" template file for display.

SENDTO_INFO Action

The SENDTO_INFO action is used to display send to information for a specific connection group. The following CGI value is required by this action:

- SITE – specifies the connection group name where this connection should be modified.

The udpRelayAdminCGI program will fill in the "udpRelaySendTo.html" template file for display by the client.

ADD_SENDTO Action

The ADD_SENDTO action is used to add a new send to for a specific connection group. The following CGI values are required by this action:

- SITE – specifies the connection group name for this connection.
- SENDTO_SITE – specifies the name of the connection group to send data to.

The udpRelayAdminCGI program will call run the SENDTO_INFO Action if the send to was successfully added.

DEL_SENDTO Action

The DEL_SENDTO action is used to delete a send to for a specific connection group. The following CGI values are required by this action:

- SITE – specifies the connection group name for this connection.
- SENDTO_SITE – specifies the name of the connection group to send data to.

The udpRelayAdminCGI program will call run the SENDTO_INFO Action if the send to was successfully added.

FILL_TMPT Action

The FILL_TMPT action is used to fill in any other custom template file as required by the webmaster.

The following variables in the template file will be replaced:

- `$hostname` – replaced by the hostname/IP address of the relay to send the admin request to.
- `$port` – replaced by the port number of the relay to send the admin request to.
- `$cryptPhrase` – replaced by the encryption phrase for the admin request.
- `$macPhrase` – replaced by the MAC phrase for the admin request.
- `$site` – replaced by the connection group name if the `SITE` CGI value was specified.

Glossary

CGI	Common Gateway Interface
HTML	Hyper-Text Markup Language
MAC	Message Authentication Code
Multicast	A form of addressing in which a set of computers is assigned one address, a copy of any datagram sent to the address is delivered to each of the computers in the set.[ξ]
RPC	Remote Procedure Call
UDP	User Datagram Protocol
Unicast	A form of packet delivery in which each computer is assigned a unique address. When a packet is sent to a unicast address, exactly one copy of the packet is delivered to the computer to which the address corresponds.[ξ]
XML	eXtensible Markup Language

[ξ] From the Glossary of "Computer Networks and Internets – Second Edition" by Douglas E. Comer, published by Prentice-Hall, Inc. 1999.