Genesis High Availability Proxy Server

CONFIGURATION GUIDE

Documentation Revision 0.0
26 May 2022

Table of Contents

Genesis High Availability Proxy Server Installation Guide

American Tel-A-Systems, Inc. dba Amtelco
800-356-9148
4800 Curtin Drive, McFarland, WI 53558
4145 North Service Road, Suite 200 Burlington, Ontario L7L 6A3

1.0 Introduction

The Genesis High Availability Proxy Server provides a SIP proxy server to facilitate high availability services for Genesis.

The proxy server itself is deployed as a high availability system using an active/passive arrangement. If the currently active proxy server node fails, the system will promote one of the passive nodes so that it becomes the new active node. Depending on the type of failure and the system configuration, current VoIP calls will be maintained with minimal disruption.

This active/passive configuration uses a “floating” or “virtual” IP address to facilitate failovers. When a node becomes active, it is assigned the floating IP address. It sends a gratuitous ARP to inform the network of its location, thus ensuring that network traffic is routed correctly.

The proxy server monitors one or more sets of Genesis systems. If one of the Genesis system fails, that system will be removed from service and calls will not be routed to it. Various recovery options are available for restoring the Genesis system when it returns to service.

Please note that high availability does not imply 100% availability. Instead it provides automatic recovery mechanisms that are intended to minimize downtime in the event of a single point failure. In some cases, recovery from a failure can occur with no noticeable disruption of service. In other cases, however, detection of the failure and subsequent failure recovery may take more than a minute. (Consider the case where a network cable is disconnected. Normal retry mechanisms for SIP can take 30 seconds or longer to detect the failure. Thus, the system may not recover until the SIP retries complete.)

1.1 Terminology

The following terms are used throughout this document:

- `Cluster`: The set of computers comprising the High Availability Proxy Server.
- `Node`: An individual server in the cluster.
- `Fence`: An external device that is used to forcibly remove a node from the cluster. One common fencing device is an intelligent power switch that can kill power to the node. Another option is a managed ethernet switch that can disable the port connected to the switch.
- `Floating IP` or `Virtual IP`: An IP address that is assigned dynamically to the active node in the cluster. Each node will have its own dedicated IP address, but the active node will also be assigned the floating/virtual IP address.

- `Dispatcher`: The proxy server has a dispatcher feature that provides a mechanism for load balancing and failover. Multiple SIP endpoints can be placed in a dispatcher set. When a call is received the proxy server can route it to one of the endpoints in the set using various load balancing algorithms. If an endpoint fails, the dispatcher can detect that and avoid routing calls to that endpoint. Furthermore, if all endpoints in a set fail, the dispatcher can fallback to a secondary set. These capabilities are used in the proxy server as an aid to providing high availability features for Genesis.

1.2 Features and Capabilities

The proxy server provides a number of features related to VoIP communications with Genesis. In addition to basic failover control for Genesis, the proxy server provides protection against attacks (similar to an SBC) as well as NAT traversal capabilities.

1.2.1 Genesis Failover

The initial implementation of Genesis High Availability uses two Genesis systems. One will be active (primary) and the other one will be available as a backup if the primary system fails. All calls will be routed to the primary Genesis system. If that system fails, the backup will be promoted to active. In this configuration, switching from a primary system only occurs if that system fails, or if Genesis/IS instructs the proxy server to switch primary systems.

The proxy server also supports an alternate recover mode in which calls are routed to the original primary system as soon as it recovers.

The system can be configured so that multiple Genesis systems are included in the primary and secondary sets. In that mode, a round robin load balancing algorithm is used to route calls to all systems in the primary set. The secondary set is only used if all of the primary systems fail.

Note that the proxy server does not recover calls when a Genesis system fails.

1.2.2 Protection From Attacks

Several options are available for minimizing the impact of attacks on Genesis. In general, the proxy server's function in these cases is to limit the amount of illicit SIP traffic that is routed to Genesis. There are several mechanisms used for this capability:

- Flood Detection: The proxy server monitors the amount of traffic from all source IPs and blocks all messages from any IP that exceed the allowed rate (unless the address is in the Allow List described below).
- Blocking of known scanners: All messages from a handful of known attackers (such as 'friendly-scanner' and 'sipvicious') are blocked.

- Deny List: The proxy server will block all traffic from any IP address in its Deny List.
- Allow List: The proxy server will allow all traffic from any IP address in its Allow List.
- Registration: The proxy server allows registered devices to use it as a typical proxy server (with some limitations). Unregistered devices cannot do so.

1.2.3 NAT Traversal

The proxy server provides NAT traversal capabilities for translating from the local network that Genesis is using to the system's public IP. This feature ensures that all SIP messaging includes appropriate routing information and also routes RTP traffic appropriately between the local network and the public IP.

The proxy server's NAT traversal feature does not fixup far end NAT issues at this time.

When the proxy server's NAT Traversal feature is enabled, the local internet router should not have SIP ALG enabled.

1.2.4 System Architecture and Call Routing

The proxy server is designed to reside between Genesis and most other devices. Figure 1 shows the typical configuration for the system.

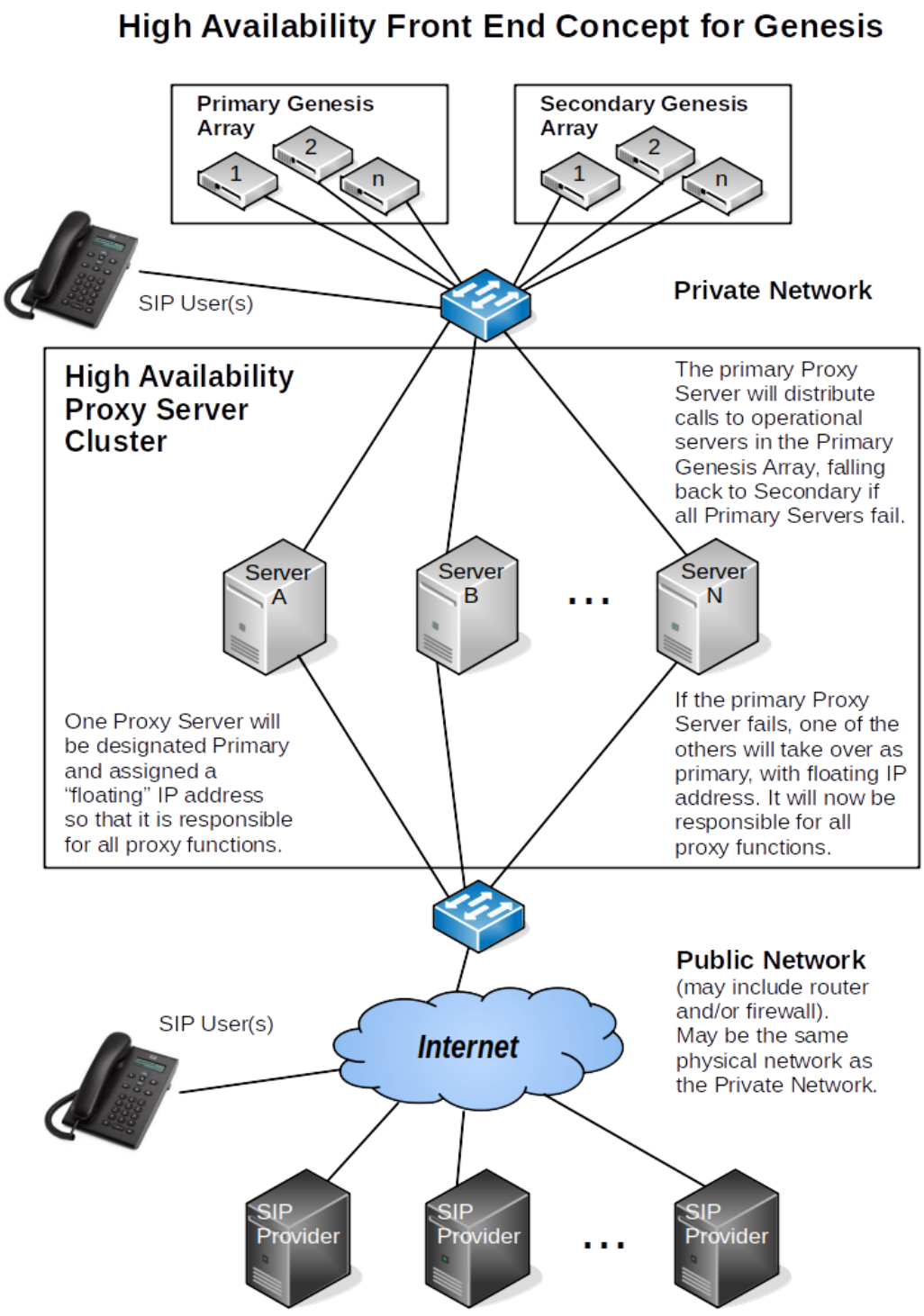
High Availability Front End Concept for Genesis
Primary Genesis Array
1
2
n
Secondary Genesis Array
1
2
n
SIP User(s)
Private Network
High Availability Proxy Server Cluster
The primary Proxy Server will distribute calls to operational servers in the Primary Genesis Array, falling back to Secondary if all Primary Servers fail.
Server A
Server B
...
Server N
One Proxy Server will be designated Primary and assigned a "floating" IP address so that it is responsible for all proxy functions.
If the primary Proxy Server fails, one of the others will take over as primary, with floating IP address. It will now be responsible for all proxy functions.
Public Network
(may include router and/or firewall).
May be the same physical network as the Private Network.
SIP User(s)
Internet
SIP Provider
SIP Provider
...
SIP Provider

Figure 1

In general, Genesis will not need to change its call routing and configuration significantly. The major differences include the following:

- Each Genesis system will register with the proxy server to facilitate routing and failover.
- Genesis will not register with SIP providers. Instead, if a SIP provider requires registration, the proxy server will do so.
- All Genesis endpoints that are to be routed through the proxy server will specify the proxy server as their outbound proxy. Note that Genesis must specify loose routing by appending '`\;lr`' to its outbound proxy settings.

The proxy server's call routing algorithm has the following characteristics:

- Calls from Genesis with the Request URI domain set to the proxy server's public or private IP address are forwarded to the specified user if the user in the Request URI is registered with the proxy server.
- Calls from Genesis whose Request URI domain is not the proxy server's public IP or private IP address are proxied to the specified URI, if possible.
- Calls from devices other than Genesis whose Request URI domain is the proxy server's public or private IP address are forwarded to Genesis.
- Calls from devices other than Genesis whose Request URI domain is not the proxy server's public or private IP address are proxied to the destination if possible, but only if the source IP is in the `proxyList`.

1.3 System Configuration

The High Availability Proxy Server system is configured via a web interface. The interface can be accessed via a browser, but also provides a RESTful API for non-browser access.

This document in primarily intended to provide details of the configuration programming that are implemented in the web interface. It will focus on the browser interface, but all of the features described are implemented similarly in the RESTful API.

The web interface supports both http and https. It incorporates basic authentication for security. (Note that basic authentication is not secure on its own. If used with http, the password is just base 64 encoded and can be decoded easily. Therefore, access via https should be preferred on anything other than a local network. The web interface uses nginx as its front end. Although nginx does have a module that can do digest authentication, it is not considered to be adequately tested to ensure security.)

2.0 System Configuration Web Interface

System configuration of the Genesis High Availability Proxy Server is accomplished via a web interface. The web interface provides a basic browser entry point at '`http[s]://<host>[:<port>]/`'. A RESTful API can be accessed at '`http[s]://<host>[:<port>]/api/`'. Additionally, openapi documentation for the RESTful API is at '`http[s]://<host>[:<port>]/openapi/`'

This document will primarily focus on the browser interface. However, all parameters and capabilities of the browser interface are incorporated into the RESTful API.

The browser interface includes a number of configuration pages, index pages, and some status and control pages. Configuration values are stored in a database as key/value pairs. To support this arrangement, configuration pages list a set of keys along with a brief description of the key, and provide an appropriate means of assigning values to the keys.

Values entered on configuration pages are not saved unless and until the user clicks on the '`Save`' button in the upper right corner of the page. If the user browses away from a page without saving, all changes are discarded. Saved changes are written to a database specific to the browser, not to the high availability system.

After configuration changes are made and saved in the browser, they must be committed to the high availability system before they take effect. This is accomplished via the `Commit Configuration` link in the browser interface.

The home page for the web interface provides a list of links and some information about the system. It is used as a launching point for all of the web interface's capabilities. To select an individual page, click on one of the links in the left window. See Figure 2.

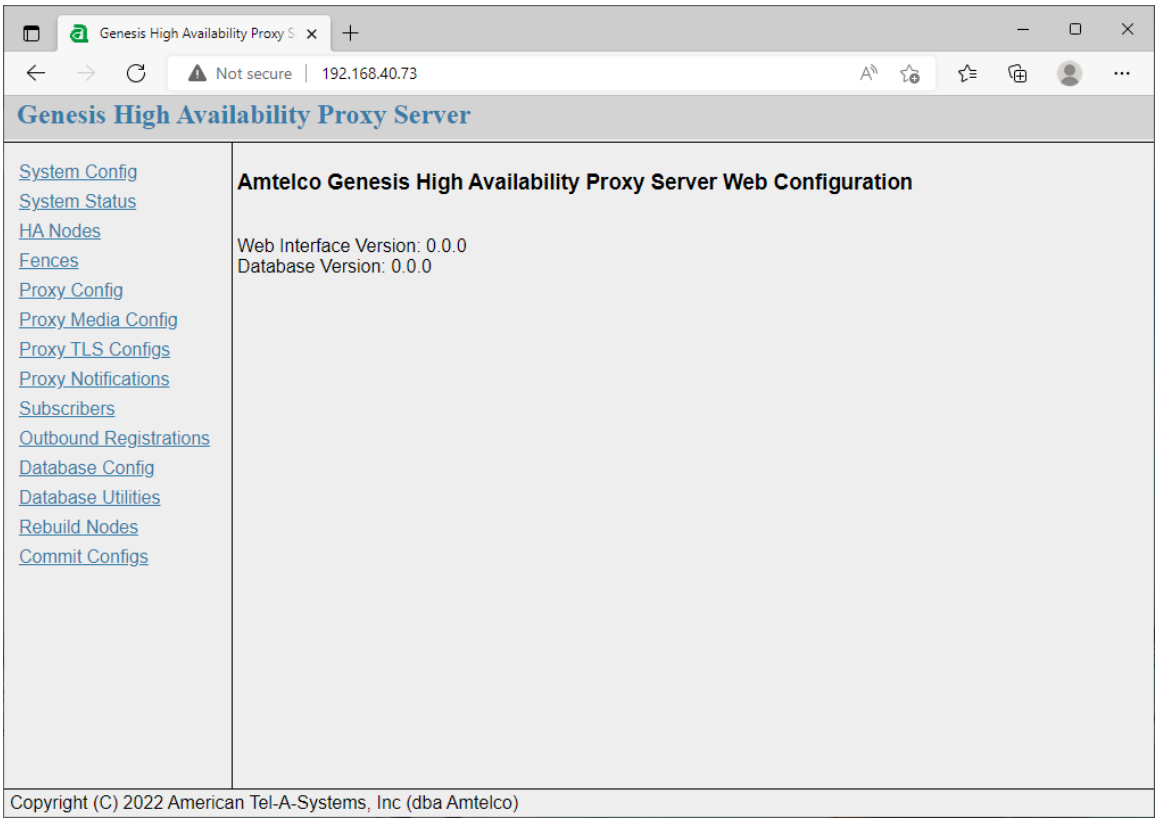

Figure 2 Web Interface Home Page

3.0 Global System Configuration

The `System Config` link provides access to system-wide configuration settings. See Figure 3.

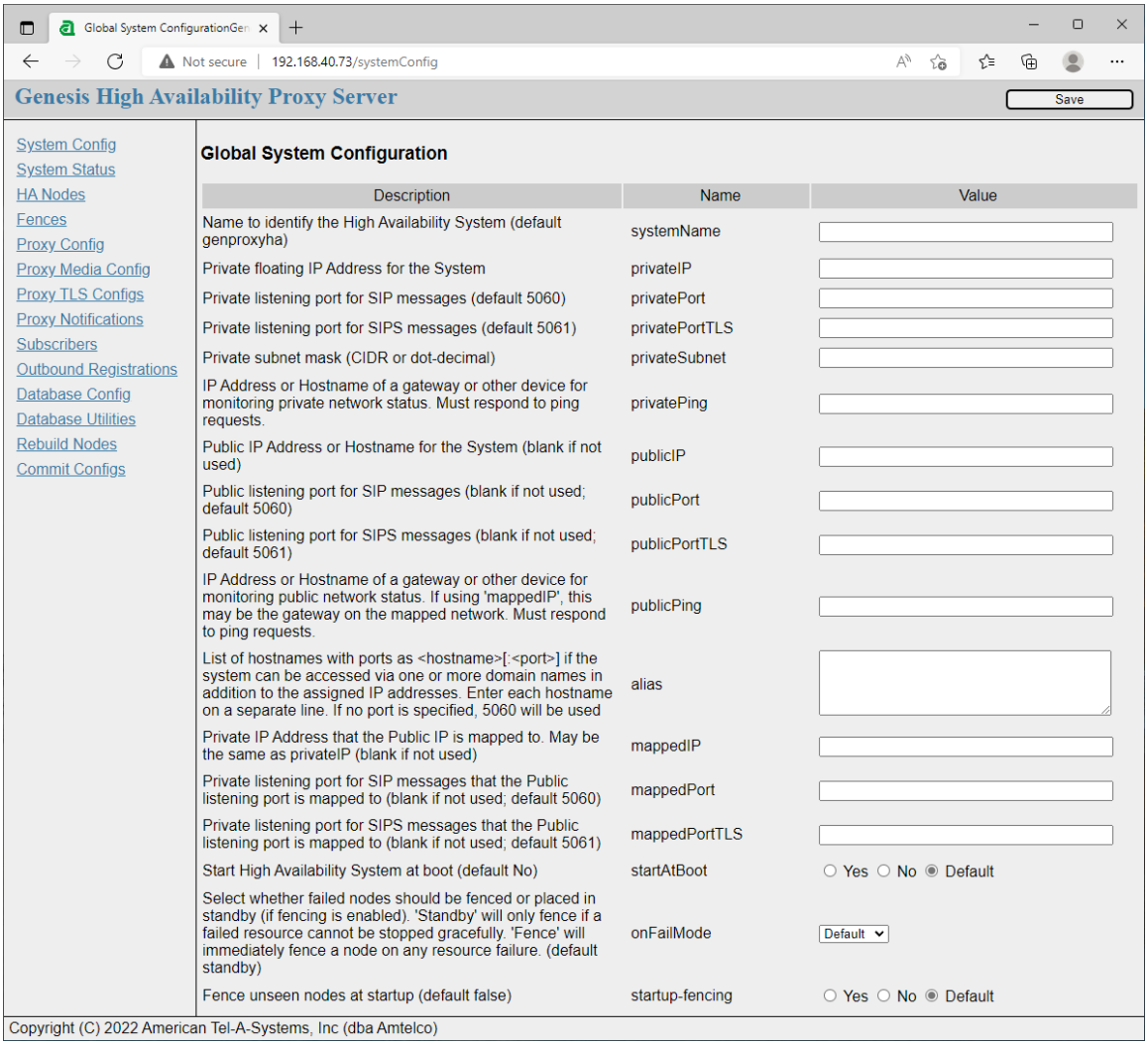

Figure 3 Global System Configuration Settings

3.1 systemName

`systemName` is an arbitrary name assigned to the High Availability System. If nothing is entered, a default name is assigned.

3.2 privateIP

`privateIP` is the “floating” or “virtual” IP address assigned to the local network for the system. This is the IP address that Genesis and other local devices will use to communicate with the system. This parameter is required for all systems.

3.3 privatePort

`privatePort` is the listening port used for all SIP messages on the local network. If not assigned, the default SIP port (`5060`) will be used.

3.4 privatePortTLS

`privatePortTLS` is the listening port used for SIPS messages from the local network. If not assigned, the default SIPS port (`5061`) will be used. The current implementation does include support for SIPS. However, use of SIPS is not recommended at this time.

3.5 privateSubnet

`privateSubnet` is the subnet mask for the local network. It can be in CIDR (e.g. `24`) or dot-decimal (e.g. `255.255.255.0`) notation. A value must be supplied for this parameter.

3.6 privatePing

`privatePing` is the IP address or hostname of a gateway or other network device that can be used to monitor the health of the private network. The selected device must respond to ping requests. A value must be supplied for this parameter.

3.7 publicIP

`publicIP` is the IP address or hostname assigned to the public network for the system. It should be set to the IP address or hostname that internet connected devices can use to reach the system. If the system is not connected to the internet, leave the parameter blank.

If the system is connected directly to the public network (i.e. with no router or address translation), all nodes in the system must be assigned addresses on the same subnet as `publicIP`. Using the system in this manner is not recommended.

NOTE: If using a hostname, `mappedIP` is required and the public network IP address MUST be entered in the `alias` parameter. Include the port number in `alias` if not using port 5060.

3.8 publicPort

`publicPort` is the listening port used for SIP messages from the public network (internet). If not assigned, the default SIP port (`5060`) will be used.

3.9 publicPortTLS

`publicPortTLS` is the listening port used for SIPS messages from the public network (internet). If not assigned, the default SIPS port (`5061`) will be used. The current implementation does include support for SIPS. However, use of SIPS is not recommended at this time.

3.10 publicPing

`pubplicPing` is the IP address or hostname of a gateway or other network device that can be used to monitor the health of the public network. The selected device must respond to ping requests. A value must be supplied for this parameter if `publicIP` is used, unless `mappedIP` is on the same subnet as `privateIP`. If `mappedIP` is used, this value may be an address on that subnet (i.e. the gateway for the internet router). Alternatively, this could be a public IP address or hostname that is known to respond to pings, which could help ensure that the system can detect failures on the public network.

3.11 alias

`alias` is used to provide a list of hostnames with ports that can be used to access the system from either the local network or the public network (internet) as `<hostname>[:<port>]`. If no port is specified, port 5060 will be assumed. Thus, if using a non-standard port or TLS, the port value is required. Generally, the alias list includes the FQDN for the system. Multiple aliases can be assigned, if needed.

3.12 mappedIP

`mappedIP` is used when the system is connected to the internet via a router. It is the local IP address that the router uses when forwarding internet traffic. If the system does not use an internet router, this field must be blank. It may be set to the same value as `privateIP`.

If `mappedIP` and `mappedPort` are the same values as `privateIP` and `privatePort` the proxy server will add route headers to SIP messages so that SIP devices on the local network will send messages to the public socket rather than the private socket. This means that each SIP device on the local network must have the public network in its route table. Additionally, the network router must have hairpin NAT traversal enabled. Since all SIP traffic would then travel through the router, this type of

operation is not recommended. Instead, ensure that either the privateIP and mappedIP are different or the privatePort and mappedPort are different. (The same is true if mappedIP and mappedPortTLS are the same as privateIP and privatePortTLS.)

3.13 mappedPort

mappedPort is the listening port used for mappedIP. If not assigned, the default SIP port (5060) will be used. Refer to mappedIP for additional information.

3.14 mappedPortTLS

mappedPortTLS is the listening port used for SIPS messages on mappedIP. If not assigned, the default SIPS port (5061) will be used. The current implementation does include support for SIPS. However, use of SIPS is not recommended at this time. Refer to mappedIP for additional information.

3.15 startAtBoot

startAtBoot is used to configure whether or not each node automatically starts the high availability system when the node boots. For most systems, this parameter should be set to 'No' or 'Default'. This ensures that if a node failure occurs, the failure can be corrected before the node is added back into the high availability system.

3.16 onFailMode

onFailMode configures the fencing operation if a node failure occurs. 'Standby' or 'Default' only enable fencing if normal cluster operations cannot isolate a failed node. 'Fence' will always activate fencing when a node failure occurs. For most situations, the default value (Standby) should be acceptable. Only applicable if fencing is enabled.

3.17 startup-fencing

startup-fencing configures whether or not the first node that starts the cluster fences all of the other nodes. Although enabling this feature is considered safer than disabling it, it also complicates system startup. For most systems, the default value (Disabled) should be acceptable. Only applicable if fencing is enabled.

4.0 Current System Status

The `System Status` link displays the current status of the high availability system. It provides information regarding the cluster and associated resources, as well as information about the current dispatcher status. The browser interface refreshes the information periodically. See Figure 4.

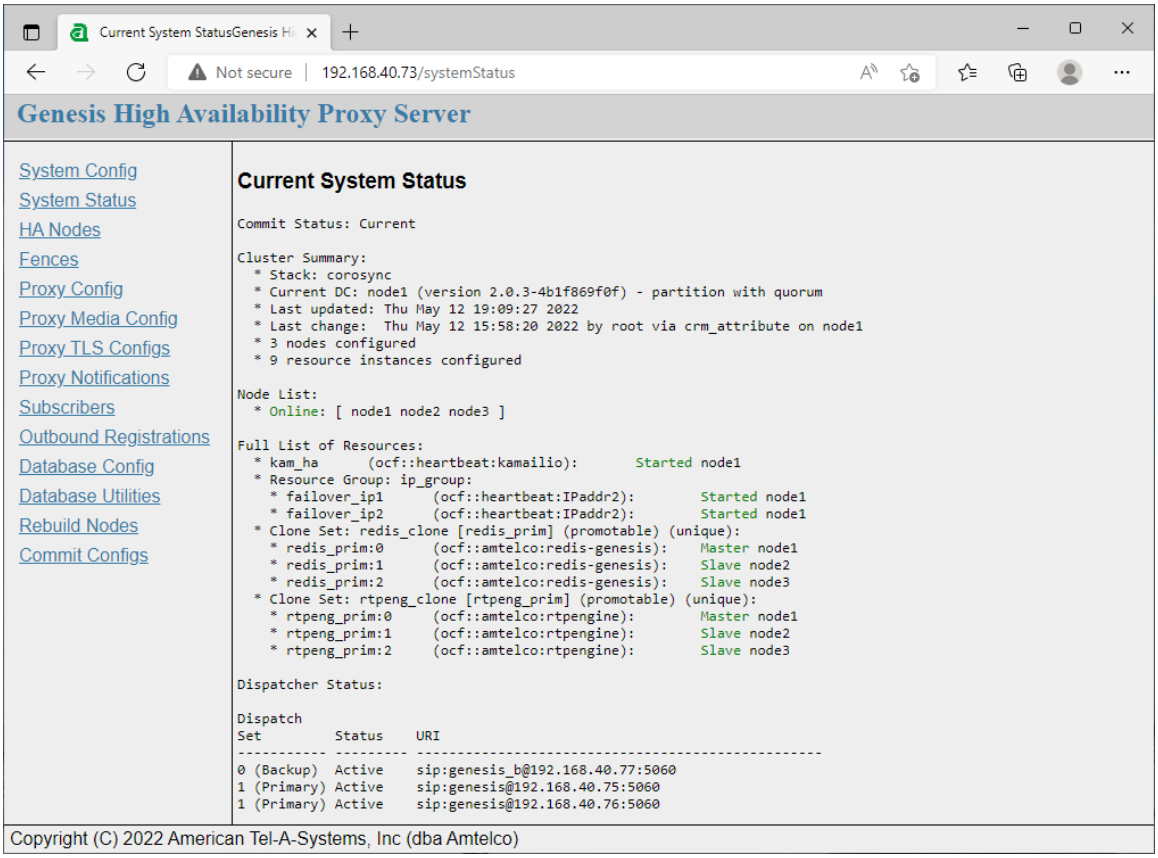

Figure 4: System Status

5.0 Node Configuration

The High Availability Cluster Server software uses multiple servers (nodes) configured in a cluster arrangement to provide the high availability services.

The `HA Node` link provides an interface for defining and editing nodes for the system.

5.1 Node Configuration Index

When the `HA Node` link is selected, the web interface displays an index of configured nodes that includes options for acting on those nodes as well as creating new nodes. See Figure 5.

The `Node Configuration` includes `nodeID` and `nodeName` columns to identify the individual nodes.

The index page also includes several actions for maintaining nodes. These are '`Create New`', '`Edit`', '`Delete`', '`Start System`', and '`Undelete`' as described below.

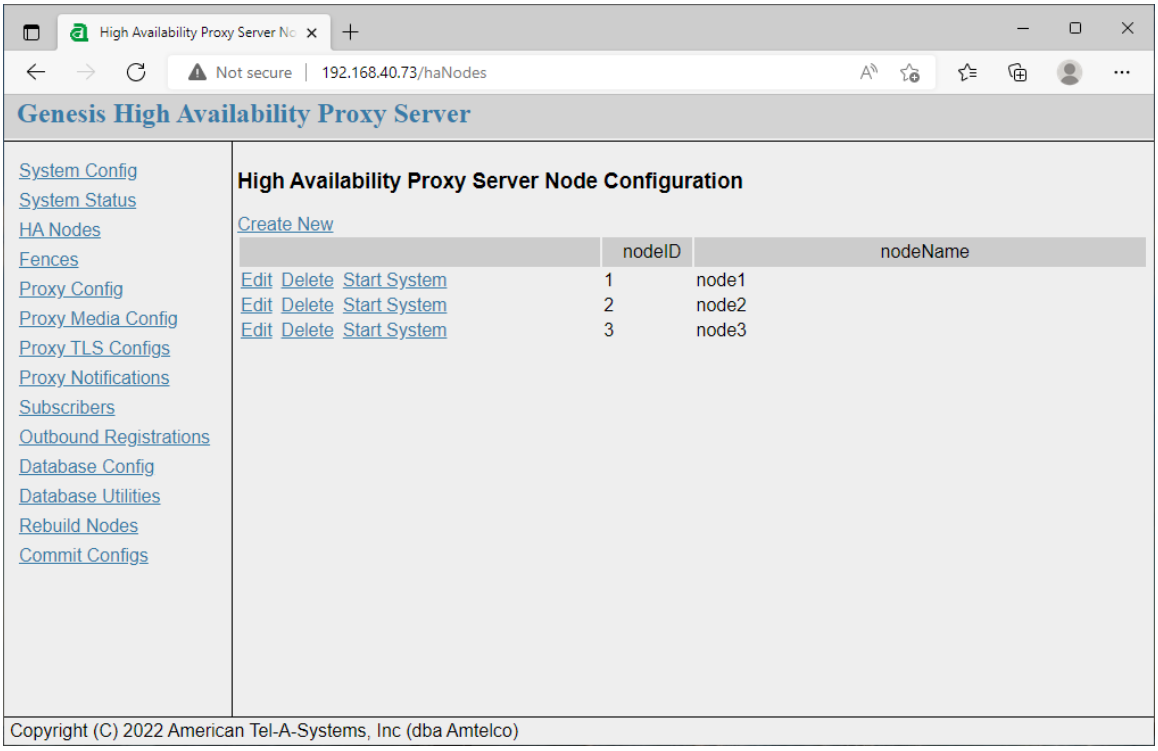

Figure 5: High Availability Node Configuration Index

5.1.1 Create New

`Create New` is used to create a new node for the cluster. When selected, a configuration screen is displayed with all of the configuration parameters needed for a new node. See Section `5.2: Node Configuration Settings`.

5.1.2 Edit

`Edit` is used to edit an existing cluster node. Selecting `Edit` on a line in the index will open the edit window for that line's node. See Section `5.2: Node Configuration Settings`. Note that once a node has been committed to the high availability system, most parameters for that node cannot be edited. If there is a need to modify any of those parameters, the node must be deleted and a new node created with the new parameters.

5.1.3 Delete

`Delete` is used to delete an existing cluster node. Selecting `Delete` on a line in the index will remove that line's node from the cluster. If there are any fences associated with the node, those fences will also be deleted.

5.1.4 Start System

`Start System` is used to start a node's high availability cluster software. Selecting `Start System` on a line in the index will start the high availability cluster software on that line's node. If the system is already started, this action will have no effect. The node must be booted and available on the network before selecting this option. Note that logging onto the node and issuing the command '`sudo crm cluster start`' has the same effect. (For reference, the following commands are also available: '`sudo crm cluster restart`' restarts the cluster, and '`sudo crm cluster stop`' shuts down the cluster. These commands have not been included in the web interface for security reasons.)

5.1.5 Undelete

If a node is deleted after it has been committed, the web interface marks it for deletion so that it will be permanently removed at the next commit. Until then, it will still be displayed in the index, but the only available action is '`UnDelete`'. If '`UnDelete`' is selected, the corresponding node's deletion flag will be cleared and the node will be returned to the list of available nodes.

Since fences are always associated with particular nodes, undeleting a node also undeletes any fences that were marked for deletion when the node was deleted.

5.2 Node Configuration Settings

If `Create New` or `Edit` is selected from the `Node Configuration`, a configuration page is displayed for the appropriate new or existing node. See Figure 6.

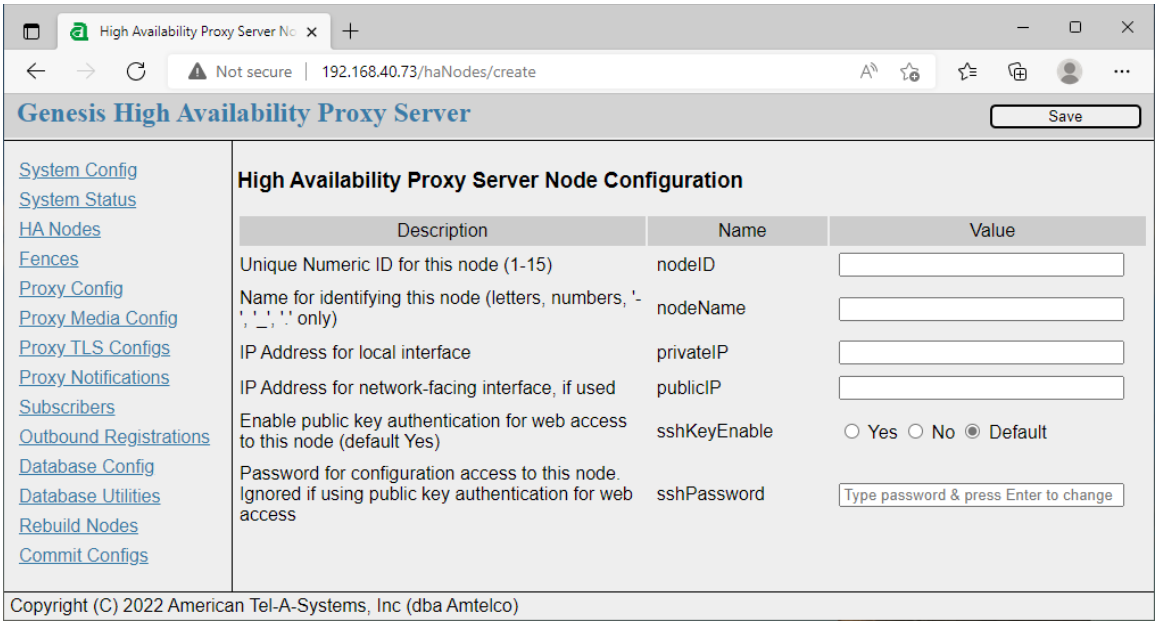

Figure 6: High Availability Node Configuration Settings

5.2.1 nodeID

`nodeID` is a numerical ID used by the cluster software to uniquely identify the node. It must be a number in the range 1-15. Once the node has been committed to the high availability system, the `nodeID` value cannot be changed.

5.2.2 nodeName

`nodeName` is a name used by the cluster to uniquely identify the node. This parameter is written, along with the node's `privateIP` to the file `/etc/hosts` on all nodes. Once the node has been committed to the high availability system, the `nodeName` value cannot be changed.

5.2.3 privateIP

`privateIP` is the IP address used for the local (private) network connection on the node. Once the node has been committed to the high availability system, the `privateIP` value cannot be changed.

5.2.4 publicIP

publicIP is the IP address used for the public network connection on the node. If there is no internet connection, this value is not used and should be blank. If the node is behind a router, the local (private) IP that is assigned for the router connection should be used. Otherwise, the public IP address for the system is used. Once the node has been committed to the high availability system, the publicIP value cannot be changed.

5.2.5 sshKeyEnable

sshKeyEnable selects whether the web interface uses public key authentication or plain text passwords to access high availability nodes. The web interface uses SSH connections to configure all of the nodes in the system, and must therefore have appropriate credentials to access those nodes. The default (and preferred) method is to use public key authentication for this purpose.

The scripts for installing the web interface can automatically create and install the appropriate keys on cluster nodes. Generally, all nodes should be built and available for SSH access before the web interface is installed. If so, when the user is prompted by the web interface installation script, the appropriate credentials can be entered for all nodes and the keys will be installed.

If nodes are added to the system after the web interface is installed, the appropriate keys can be added by running sudo cpsshkeys on the server running the web interface. This file is installed in /usr/local/bin, as well as in the root directory for the current version of the web interface.

If the web interface is running on multiple servers, the keys must be installed from all servers, not just one.

If plain text passwords are to be used for authentication, the lines

```
Match User genweb
PasswordAuthentication no
```

must be removed from /etc/ssh/sshd_config on all nodes. Furthermore, a password must be assigned for user genweb on all nodes.

5.2.6 sshPassword

sshPassword is only used if sshKeyEnable is set to 'No'. It is the password assigned to user genweb on all cluster nodes. Use of plain text password authentication is strongly discouraged, but can be used if absolutely necessary. See sshKeyEnable for more information.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

6.0 Fence Configuration

The High Availability Cluster Server software incorporates a feature that will forcibly remove failed nodes from the system. This feature, known as fencing, uses external devices such as ethernet switches or intelligent power distribution units to ensure that the failed node cannot disrupt the system. For instance, if the active node fails in such a way that it cannot remove the shared IP address, but still responds to network queries, there is no way for another node to assume the shared IP address. In this case, the cluster will be down until the network on the failed node is disabled. Fence devices are used to implement this capability automatically.

Use of fencing is highly recommended to ensure maximum reliability. Managed ethernet switches often have the ability to disable individual ports via SNMP or other means. TrippLite and APC both manufacture power distribution units that can be used to fence by shutting down power to a specified outlet. Other companies have similar offerings.

Many fence agents are installed on the high availability nodes. Amtelco has tested `fence_tripplite_snmp` that works with some TrippLite PDUs. Amtelco also developed a `fence_cisco_sgx00` fence agent that is intended to work with Cisco SGx00 switches (e.g. SG300-28) and a more generic `fence_ether_snmp` that may work with other switches that use standard SNMP entries for controlling ports. The `fence_ether_snmp` agent has been tested with Cisco SG300-28, Aruba 5640, and Linksys SRW2016 switches. Use of other devices may be considered if needed for specific customers.

The current implementation only supports fences that communicate via SNMP.

The `Fence` link provides an interface for defining and editing fence devices for the system.

6.1 Fence Configuration Index

When the `Fence` link is selected, the web interface displays an index of configured fences that includes options for acting on those fences as well as creating new fences. See Figure 7.

Each fence device will need to be configured appropriately so that its fencing capabilities are available and active. The fence configuration for the high availability system will then need to be configured accordingly.

The `Fence Configuration Index` includes `fenceName`, `fenceNodeName`, and `fencePriority` columns to identify the individual fence devices.

The index page also includes several actions for maintaining fences. These are ‘`Create New`’, ‘`Edit`’, ‘`Delete`’, ‘`UnFence`’, and ‘`UnDelete`’ as described below.

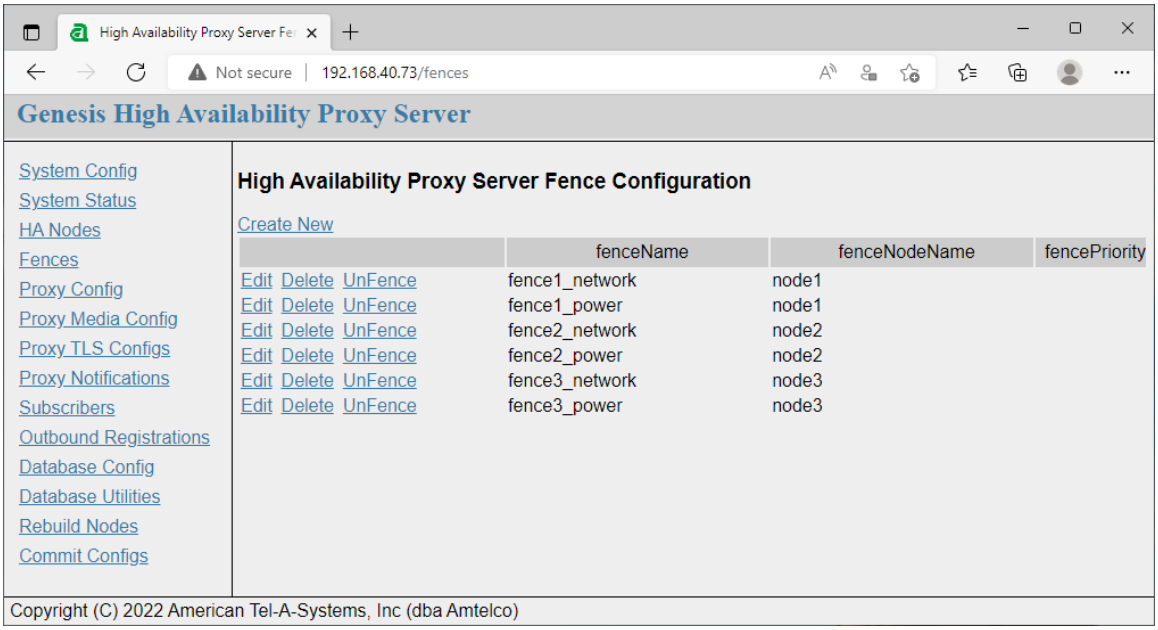

Figure 7: Fence Configuration Index

6.1.1 Create New

`Create New` is used to create a new fence for a cluster node. When selected, a configuration screen is displayed with all of the configuration parameters needed for a new fence device. See Section `6.2: Fence Configuration Settings`.

6.1.2 Edit

`Edit` is used to edit an existing fence device. Selecting `Edit` on a line in the index will open the edit window for that line’s fence. See Section `6.2: Fence Configuration Settings`.

6.1.3 Delete

`Delete` is used to delete an existing fence device. Selecting `Delete` on a line in the index will remove that line’s fence from the cluster.

6.1.4 UnFence

`UnFence` is used to return a fence device to its normal operational state so the affected node can be returned to the system. Fencing can also be controlled via the actual fence device, but the `UnFence` feature is provided as a convenience.

6.1.5 UnDelete

If a fence is deleted after it has been committed, the web interface marks it for deletion so that it will be permanently removed at the next commit. Until then, it will still be displayed in the index, but the only available action is '`UnDelete`'. If '`UnDelete`' is selected, the corresponding fence's deletion flag will be cleared and the fence will be returned to the list of available fences.

Since fences are always associated with particular nodes, deleting and undeleting a node also affects the fences assigned to that node. When a node is marked for deletion, all fences assigned to the node are also marked for deletion. However, in this case, the affected fences cannot be undeleted unless the node is undeleted, and will not be displayed in the index. If the node is undeleted, it's associated fences will be returned to the index.

If a fence is marked for deletion prior to its associated node being deleted, the fence's deletion flag is maintained if the node is deleted and then undeleted. In this case, when the node is undeleted, the fence will still be marked for deletion and is still available for undeletion.

6.2 Fence Configuration Settings

If `Create New` or `Edit` is selected from `Fence Configuration Index`, a configuration page is displayed for the appropriate new or existing fence. See Figure 8.

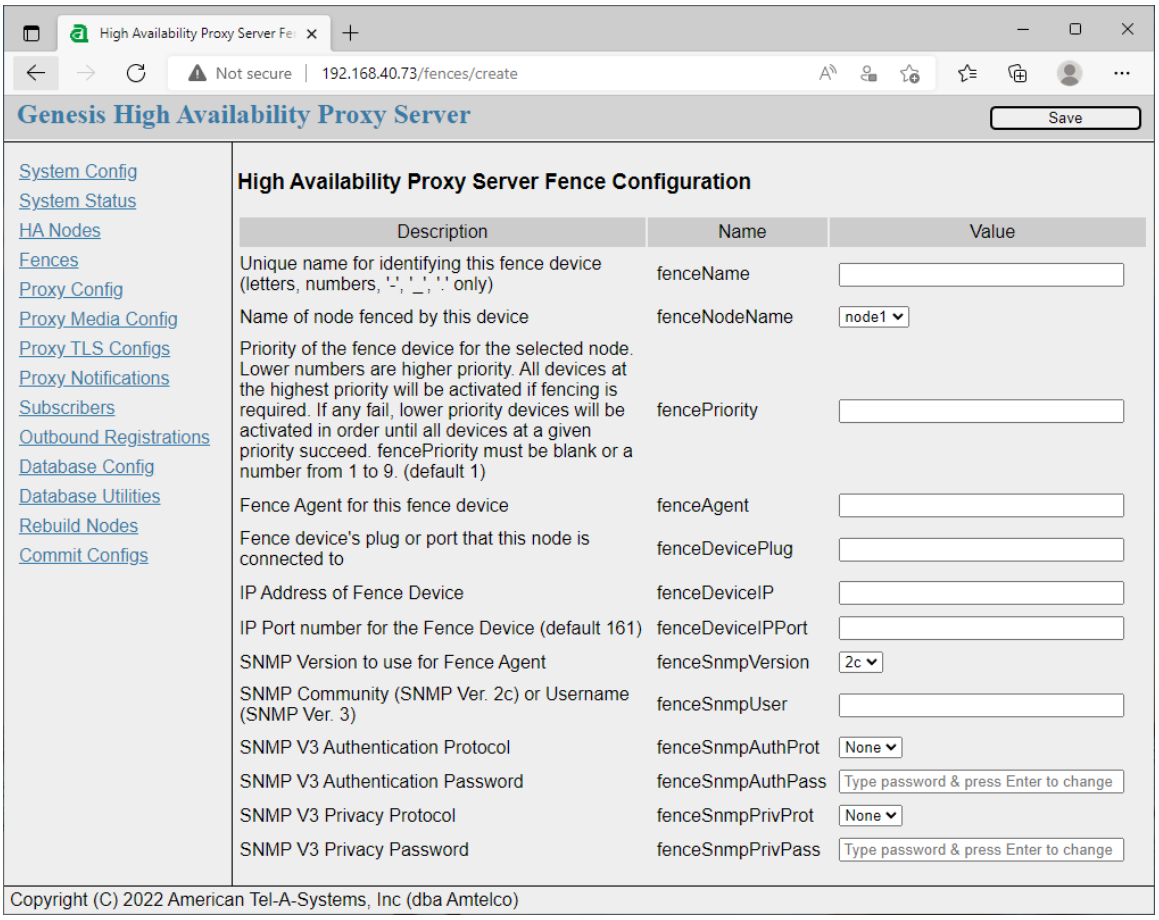

Figure 8: Fence Configuration Settings

6.2.1 fenceName

`fenceName` is a name selected by the user to uniquely identify the fence device. It must be unique within the set of fences.

6.2.2 fenceNodeName

`fenceNodeName` selects which cluster node the fence applies to. The pulldown provides a list of all defined nodes in the system. Only defined nodes can have fences assigned. When a node is deleted, all fences associated with that node are also deleted.

6.2.3 fencePriority

`fencePriority` specifies the priority of the fence when more than one fence is assigned to a cluster node. If specified, it must be a value from `1` to `9`, where `1` is highest priority and `9` is lowest priority. If blank, the priority defaults to `1`.

Fence priorities are used to determine the conditions under which individual fences are activated. When fencing is activated for a node due to node failure, all of the fence devices at the highest priority are activated. If any of these fail, all of the fences at the next highest priority are then activated. This process continues until all fences at a given level are successfully activated (or until there are no more fences to activate).

For example, if there are two fence devices for a node, one to disable the network and one to shut off power, the network device might be set to level 1 and the power device set to level 2. Then, if a node failure occurs, the network device would be activated first. If it succeeds, the power device will not be activated. If it fails, however, the power device will be activated to forcibly shut off power to the device. Since shutting off power without a clean shutdown is generally undesirable, this configuration helps to minimize the chances of having to shut off power after a failure.

6.2.4 fenceAgent

`fenceAgent` is the name of the script that is used to control the fence. The fence agents are installed on every node in `/usr/sbin` and have filenames beginning with '`fence_`'. Amtelco has qualified `fence_tripplite_snmp` with a TrippLite PDUMH15NET2LX Switched PDU, `fence_cisco_sgx00` with a Cisco SG300-28 switch, and `fence_ether_snmp` with Cisco SG300-28, Aruba 5640, and Linksys SRW2016 switches.

The current implementation only supports fence agents that use SNMP communication.

6.2.5 fenceDevicePlug

`fenceDevicePlug` specifies the fence device's connector used for the selected node. The actual value used depends on the fence agent. For many devices, it will be a number from `1` to the maximum number of ports on the device.

6.2.6 fenceDeviceIP

`fenceDeviceIP` is the IP address of the fence device.

6.2.7 fenceDeviceIPPort

`fenceDeviceIPPort` is the network port number used for communicating with the fence device. SNMP typically uses port `161`, which is the default value if none is specified.

6.2.8 fenceSnmpVersion

`fenceSnmpVersion` specifies the SNMP version to use for the fence agent. Versions `2c` and `3` are supported. Version 3 offers some security, whereas 2c just uses a plain text username for access.

6.2.9 fenceSnmpUser

`fenceSnmpUser` is used to specify the Community (SNMP Version 2c) or the Username (SNMP Version 3) that has access to the fencing function of the fence device.

6.2.10 fenceSnmpAuthProt

`fenceSnmpAuthProt` is the Authentication protocol to use when SNMP Version 3 is selected. It can be set to '`None`', '`SHA`', or '`MD5`'.

6.2.11 fenceSnmpAuthPass

`fenceSnmpAuthPass` is the password used when SNMP Version 3 authentication is enabled.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

6.2.12 fenceSnmpPrivProt

`fenceSnmpPrivProt` is the privacy protocol to use when SNMP Version 3 is selected. It can be set to '`None`', '`DES`', or '`AES`'.

6.2.13 fenceSnmpPrivPass

`fenceSnmpPrivPass` is the password used when SNMP Version 3 privacy is enabled.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is

accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

7.0 Proxy Configuration

The `Proxy Config` link provides access to configuration settings specific to the system's proxy server functionality. (See Figure 9, Figure 10, and Figure 11).

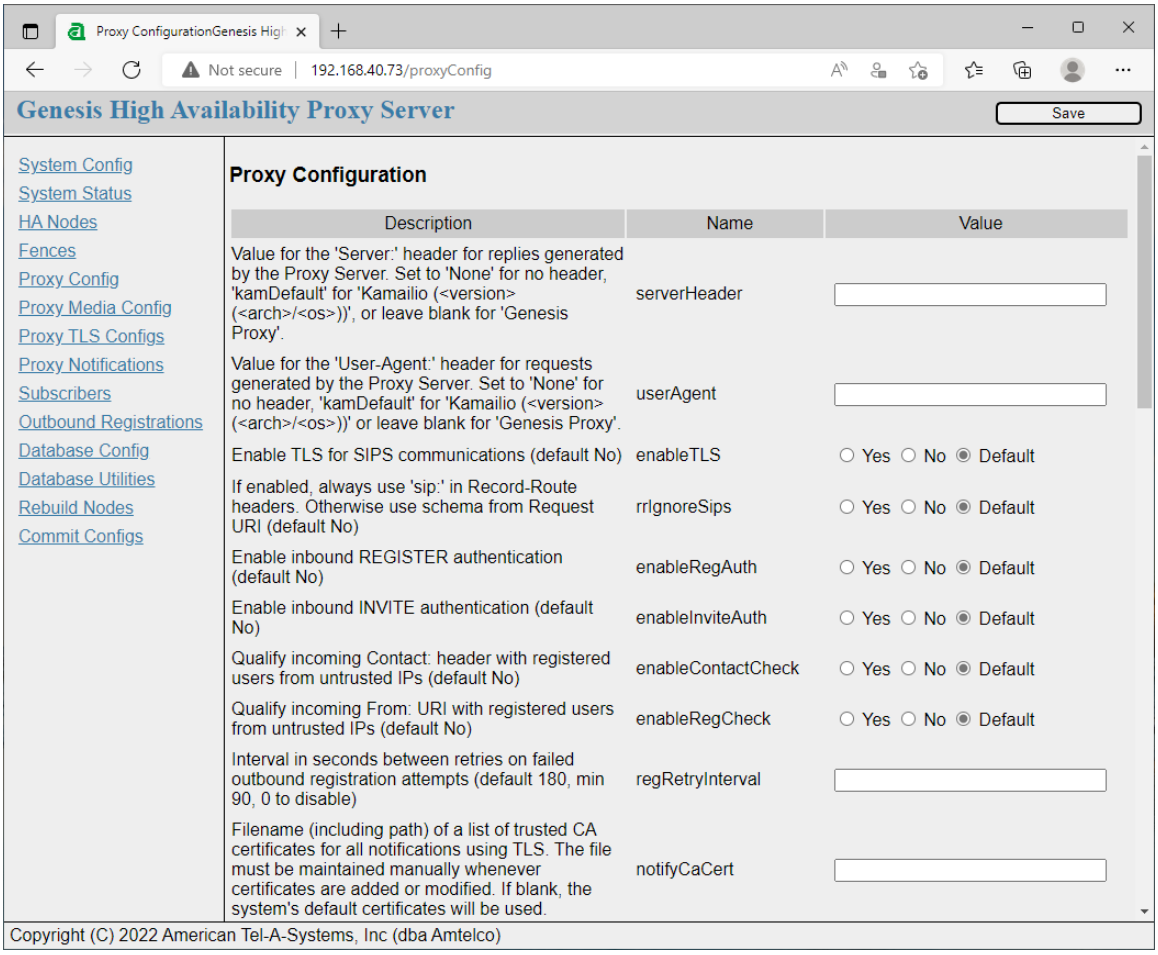

Figure 9: SIP Proxy Configuration Settings Page 1

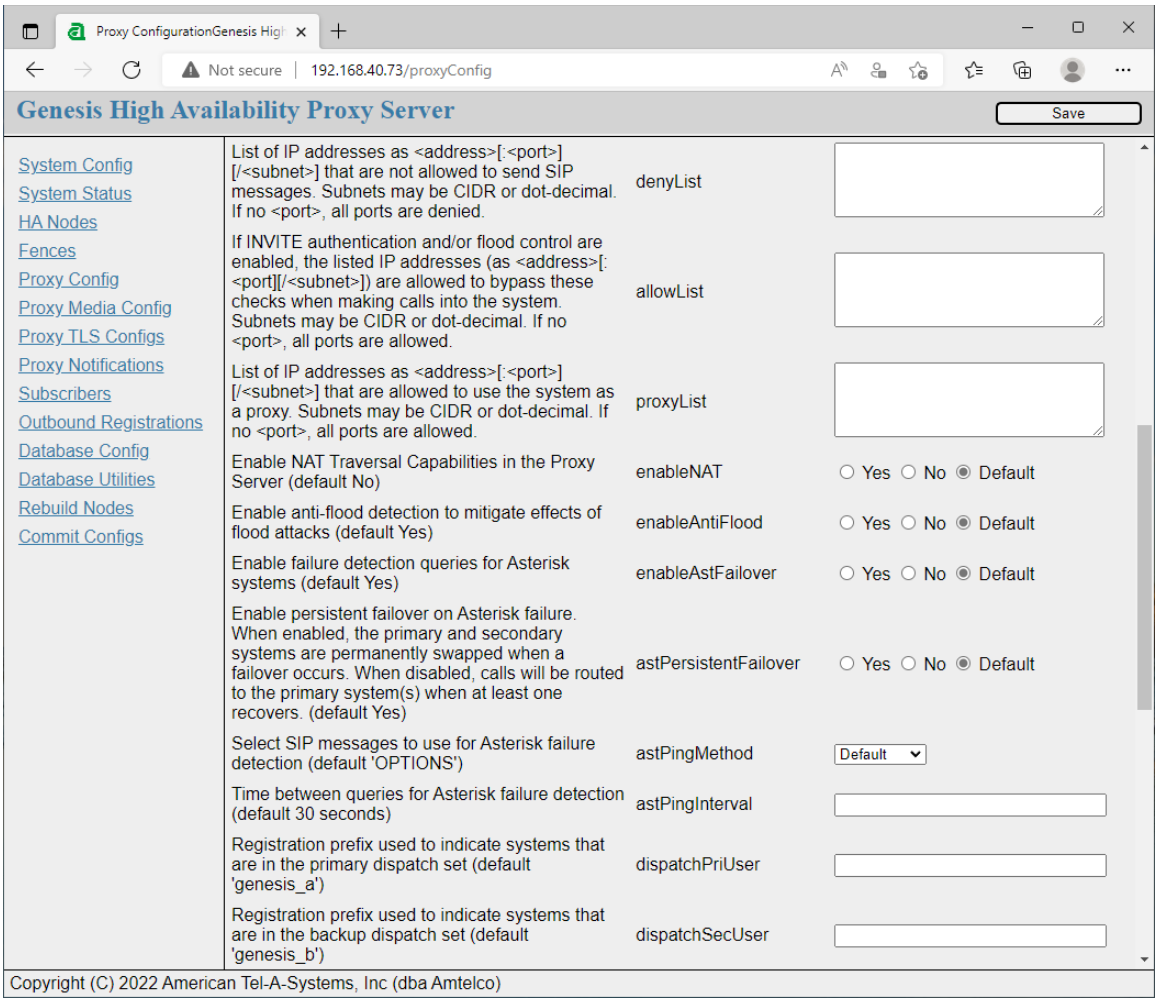

Figure 10: SIP Proxy Configuration Settings Page 2

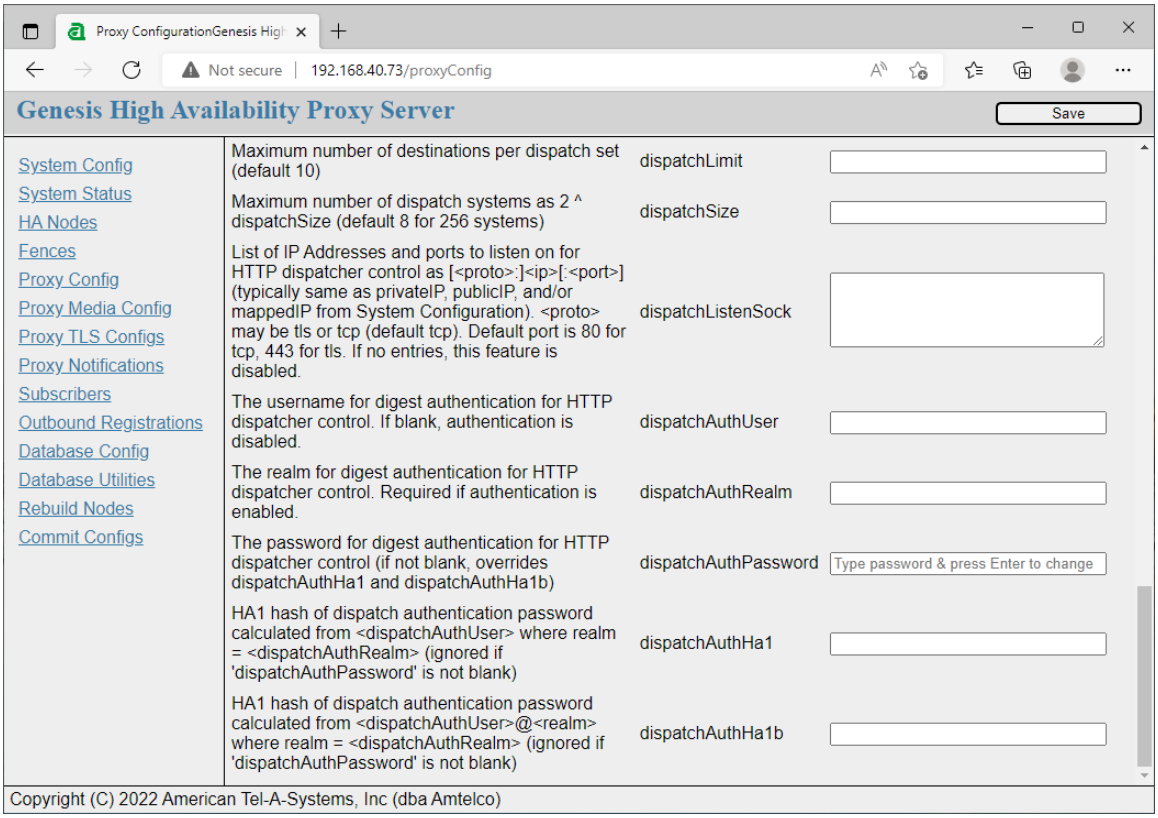

Figure 11: SIP Proxy Configuration Settings Page 3

7.1.1 serverHeader

`serverHeader` specifies the value used in the SIP `Server:` header when the proxy server sends replies. This may be useful when connecting to a provider that requires a particular header. If blank, the value will default to '`Genesis Proxy`'. If set to '`None`', no `Server:` header will be inserted in the reply. If set to '`kamDefault`', the default header supplied by Kamailio (the proxy server software package) will be used. Otherwise, the `Server:` header will be set to the supplied value.

7.1.2 userAgent

`userAgent` specifies the value used in the SIP `User-Agent:` header when the proxy server generates a request. This may be useful when connecting to a provider that requires a particular header. If blank, the value will default to '`Genesis Proxy`'. If set to '`None`', no `User-Agent:` header will be inserted in the request. If set to '`kamDefault`', the default header supplied by Kamailio (the proxy server software package) will be used. Otherwise, the `User-Agent:` header will be set to the supplied value.

7.1.3 enableTLS

`enableTLS` selects whether or not TLS is available for SIP messaging (i.e. SIPS). The current implementation does include support for SIPS. However, use of SIPS is not recommended at this time.

If `enableTLS` is set to '`Yes`', appropriate TLS settings will need to be made via the `Proxy TLS Configuration Settings` page.

7.1.4 rrIgnoreSips

`rrIgnoreSips` selects whether or not the proxy server sets the schema in Record-Route headers to '`sip:`' If `rrIgnoreSips` is set to '`Yes`', the proxy server will always use '`sip:`' in Record-Route headers. Otherwise, the proxy server will use the schema from the Request URI (either '`sip:`' or '`sips:`').

7.1.5 enableRegAuth

`enableRegAuth` selects whether or not inbound registrations (configured via the `Subscribers Settings` page) require authentication. If set to '`Yes`', all registration requests will be rejected with a `401 Unauthorized` response and will require a subsequent authenticated registration request.

7.1.6 enableInviteAuth

`enableInviteAuth` selects whether or not inbound `INVITE`s that are intended for Genesis require authentication. If set to '`Yes`', any `INVITE` for Genesis that is not in the `allowList` will be rejected with a `407 Proxy Authentication Required` response and will require a subsequent `INVITE` with authentication credentials. Note that endpoints that are not in the `allowList` cannot make calls to Genesis unless they are registered (as configured via the `Subscribers Settings` page).

7.1.7 enableContactCheck

`enableContactCheck` selects whether or not contact checking is enabled for inbound `INVITE`s that are intended for Genesis. If enabled, the proxy server checks to make sure the `From:` user is registered and that the `Contact:` header matches the registration. If the test fails and the `INVITE`'s source address is not in the `allowList`, the call is ignored.

7.1.8 enableRegCheck

`enableRegCheck` selects whether or not registration checking is enabled for inbound `INVITE`s that are intended for Genesis. If enabled, the proxy server only allows the call if the `From:` user is registered or the `INVITE`'s source address is in the `allowList`. Otherwise, the call is ignored. If `enableContactCheck` is enabled, this setting has no effect, as `enableContactCheck` always forces a registration check.

7.1.9 regRetryInterval

`regRetryInterval` specifies how often outbound registrations as configured via `Outbound Registrations Settings` are retried in case of failure. The timer used for this function runs once every 90 seconds and the value entered here is rounded to the nearest multiple of 90 seconds. If set to 0, failed registrations are never retried.

7.1.10 notifyCaCert

`notifyCaCert` specifies a certificate file for verifying servers for outbound notifications that use TLS (as configured in `Proxy Notifications Settings`). When using this parameter, concatenate all of the required certificates into the specified file.

This parameter is only used if `notifyVerifyHost` is enabled. If this field is blank and host verification is required, the proxy server will use system's default certificate list for the necessary certificate.

7.1.11 denyList

`denyList` is used to prevent certain source addresses from making calls through the proxy server. Any SIP messages from an IP in the `denyList` will be discarded. The `denyList` can include individual IP addresses with optional port and subnet. Each IP address or subnet is entered as `<IP Address>[:<port>][/<subnet>]`. The subnet, if included can be either CIDR format (e.g. `24`) or dot-decimal format (e.g. `255.255.255.0`). The port, if included, only denies messages from that port. Multiple comma- or space-delimited entries can be made on each line, if desired.

`denyList` modifications can be committed to the system without causing any down time.

7.1.12 allowList

`allowList` is used to allow certain source IP addresses to make calls to Genesis without regard to registration or flood detection checks and also without `INVITE` authentication. For example, if the system is using a SIP provider, that provider's IP

address would typically be included in the allowList, so that calls from the SIP provider to Genesis would be allowed.

Source IP addresses in the allowList are also allowed to send OPTIONS requests to the proxy server. Devices that need this capability should be included in either the allowList or the proxyList. Since the cluster software uses OPTIONS requests to monitor the health of the proxy server, the configuration software automatically adds all cluster nodes to the allowList.

The allowList can include individual IP addresses with optional port and subnet. Each item is entered as <IP Address>[:<port>][/<subnet>]. The subnet, if included, can be either CIDR format (e.g. 24) or dot-decimal format (e.g. 255.255.255.0). The port, if included, only allows messages from that port. Multiple comma- or space-delimited entries can be made on each line, if desired.

allowList modifications can be committed to the system without causing any down time.

7.1.13 proxyList

proxyList is used to allow certain source IP addresses to use the proxy system as a typical proxy server. This might be used if the system includes SIP phones on the local network that make calls to each other.

The proxyList can include individual IP addresses with optional port and subnet. Each IP address or subnet is entered as <IP Address>[:<port>][/<subnet>]. The subnet, if included, can be either CIDR format (e.g. 24) or dot-decimal format (e.g. 255.255.255.0). The port, if included, only allows messages from that port. Multiple comma- or space-delimited entries can be made on each line, if desired.

Genesis systems do not need to be included in the proxyList. Genesis is always allowed to use the proxy system as a proxy server.

Source IP addresses in the proxyList are also allowed to send OPTIONS requests to the proxy server. Devices that need this capability should be included in either the allowList or the proxyList. Since the cluster software uses OPTIONS requests to monitor the health of the proxy server, the configuration software automatically adds all cluster nodes to the allowList so there is no need to add cluster nodes to the proxyList.

proxyList modifications can be committed to the system without causing any down time.

7.1.14 enableNAT

`enableNAT` selects whether the NAT traversal capabilities of the proxy server are used. This feature is primary used for proxying RTP traffic between two networks, typically the local network and the internet. If the system only uses a local network, `enableNAT` should be disabled. In most other systems, it should be enabled.

7.1.15 enableAntiFlood

`enableAntiFlood` selects whether the proxy server will limit the number of incoming requests that are allowed from any given IP in any given time period. This is primarily useful for preventing Denial-of-Service (DoS) attacks. When enabled, if an IP that is not in the `allowList` or `proxyList` attempts to flood the system with requests, the proxy server will drop all requests that exceed the limit.

The `denyList` IP addresses are discarded before the anti-flood filter. Thus, requests from IP addresses in the `denyList` will always be discarded.

The anti-flood feature is active for **any** IP address that is not in the `allowList` or the `proxyList`. Thus, it is imperative that all IP addresses that can legitimately send large numbers of requests be included in one of these lists!

Typically, the anti-flood feature will block an IP if the proxy server detects more than approximately 8 requests per second for somewhere between 2 and 6 seconds. Once the IP address is blocked, the message rate will be checked every 5 minutes. As long as the rate remains above the threshold, the IP will continue to be blocked for 5 minute intervals. Note that these numbers are approximate. Some variance is expected.

7.1.16 enableAstFailover

`enableAstFailover` selects whether or not the proxy server should query Asterisk (or Genesis) systems for failures. If enabled, any endpoint that has registered with the username prefix specified in either `dispatchPriUser` or `dispatchSecUser` will be sent an `OPTIONS` message periodically to ensure that the system is still responding. By default, the feature is enabled.

This feature helps to ensure that failures of an Asterisk or Genesis system are detected even with low call traffic. If not enabled, failures will still be detected, but not until the proxy server sends a request that subsequently fails.

7.1.17 astPersistentFailover

`astPersistentFailover` specifies whether failure of all Asterisk (or Genesis) systems in the current primary dispatch set will persist even if some or all of the systems

recover. In either mode, the proxy server will fail to the secondary dispatch set if all primary systems fail.

If `astPersistentFailover` is disabled, if any system in the primary set recovers, the proxy server will revert to using it as the primary set.

If `astPersistentFailover` is enabled, the primary and secondary sets are swapped when all systems in the primary set fail. Thus, the original secondary set will become primary and remain primary until all system in its set fail (assuming at least one system in the original primary set has recovered.) Alternatively, the current primary dispatch set can be forced by issuing an appropriate POST request to one of the sockets in the `dispatchListenSock` parameter.

7.1.18 astPingMethod

`astPingMethod` selects whether `OPTIONS` or `INFO` messages are used to query Asterisk (or Genesis) systems when `enableAstFailover` is enabled. The values can be `Default`, `OPTIONS`, or `INFO`. `OPTIONS` is the default mode.

7.1.19 astPingInterval

`astPingInterval` specifies (in seconds) how often queries are sent to Asterisk (or Genesis) when `enableAstFailover` is enabled. The default value of 30 seconds should be adequate for most configurations, but can be changed, if necessary.

7.1.20 dispatchPriUser

`dispatchPriUser` specifies a prefix for the user portion of a registration's address of record that specifies the primary Asterisk (or Genesis) dispatch set. Asterisk/Genesis systems that will be in the primary set must register using the value in this field as the beginning of the user portion of the `To:` header in the registration. (This is the user portion of the `client_uri` field when using Asterisk's PJSIP channel driver.) The default value is '`genesis_a`'. Note that the value of `dispatchSecUser` cannot be a subset of this value.

The user portion of the `To:` header can just be the value of `dispatchPriUser`. However, additional text can be appended to the user if desired. Thus, if `dispatchPriUser` is set to '`genesis_a`', then the user value could be '`genesis_a`' or '`genesis_a-1`' or any other valid user starting with '`genesis_a`'.

When `astPersistentFailover` is enabled, the roles of the primary and secondary dispatch sets can switch; thus, this set becomes the secondary set after a failure. When `astPersistentFailover` is disabled, this set is always primary. If at least one

system in the set recovers after a failure, the proxy server will continue using it as the primary set.

The proxy server supports multiple Asterisk/Genesis systems in the primary dispatch set, if needed. All of the systems can use the same user, or can have unique user values, as long as the user starts with the `dispatchPriUser` value.

7.1.21 dispatchSecUser

`dispatchSecUser` specifies a prefix for the user portion of a registration's address of record that specifies the secondary Asterisk (or Genesis) dispatch set. Asterisk/Genesis systems that will be in the secondary set must register using the value in this field as the beginning of the user portion of the `To:` header in the registration. (This is the user portion of the `client_uri` field when using Asterisk's PJSIP channel driver.) The default value is '`genesis_b`'. Note that the value cannot be a subset of the value in `dispatchPriUser`.

The user portion of the `To:` header can just be the value of `dispatchSecUser`. However, additional text can be appended to the user if desired. Thus, if `dispatchSecUser` is set to '`genesis_b`', then the user value could be '`genesis_b`' or '`genesis_b-1`' or any other valid user starting with '`genesis_b`'.

When `astPersistentFailover` is enabled, the roles of the primary and secondary sets can switch; thus, this set becomes the primary set after a failure. When `astPersistentFailover` is disabled, this set is always secondary and will only be used when all systems in the primary set have failed. If at least one system in the primary set recovers after a failure, the proxy server will continue using it as the primary set.

The proxy server supports multiple Asterisk/Genesis systems in the secondary dispatch set, if needed. All of the systems can use the same user, or can have unique user values, as long as the user starts with the `dispatchSecUser` value.

7.1.22 dispatchLimit

`dispatchLimit` sets the maximum number of Asterisk/Genesis systems in any dispatch set. The default value (`10`) should be adequate for most systems.

7.1.23 dispatchSize

`dispatchSize` sets the maximum number of dispatch systems allowed. The value is a power of 2, so the default value of `8` gives 256 (2^8) systems. The default should be adequate for most systems.

7.1.24 dispatchListenSock

`dispatchListenSock` sets listening sockets for an http server that clients can use to query or set the current primary dispatcher set. Multiple sockets can be included and have the form `[<proto>:]<ip>[:<port>]`. `<proto>`, if included, can be either '`tcp`' or '`tls`'. If omitted, '`tcp`' is used. `<port>`, if included, specifies the listening port. If omitted, port 80 is used for TCP and port 443 is used for TLS. If there are no entries in this field, no listening sockets will be configured, so the http server will be disabled.

The sole purpose of the server is to provide a client the ability to query or set the current primary dispatch set. If a client issues a GET request, the server will return the current primary dispatch set. If a client issues a POST request, the server will set the primary dispatch set to the value provided. Details of the GET and POST requests are documented elsewhere.

7.1.25 dispatchAuthUser

`dispatchAuthUser` is used if authentication is desired for the HTTP server enabled via `dispatchListenSock`. If a username is entered in `dispatchAuthUser`, digest authentication will be enabled for the `dispatchListenSock` sockets. All of the sockets use the same username. If no username is entered, authentication will be disabled.

7.1.26 dispatchAuthRealm

`dispatchAuthRealm` is the realm to use when digest authentication is enabled via `dispatchAuthUser`. It must be specified if `dispatchAuthUser` is not blank. A typical value is the domain of the proxy server, but that is not a requirement.

7.1.27 dispatchAuthPassword

dispatchAuthPassword is the password to use when digest authentication is enabled via `dispatchAuthUser`. If blank, the password hashes in `dispatchAuthHa1` and `dispatchAuthHa1b` are used instead.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

7.1.28 dispatchAuthHa1

`dispatchAuthHa1` is the HA1 password hash to use when digest authentication is enabled via `dispatchAuthUser`. This field is only used if `dispatchAuthPassword` is blank. The hash is the MD5 checksum calculated from: `<dispatchAuthUser>:<dispatchAuthRealm>:<password>`

7.1.29 dispatchAuthHa1b

`dispatchAuthHa1b` is the HA1B password hash to use when digest authentication is enabled via `dispatchAuthUser`. This field is only used if `dispatchAuthPassword` is blank. The hash is the MD5 checksum calculated from: `<dispatchAuthUser>@<dispatchAuthRealm>:<dispatchAuthRealm>:<password>`

8.0 Proxy Media (RTP) Configuration

The `Proxy Media Config` link provides access to configuration settings specific to the system's SIP proxy media (RTP) server functionality. See Figure 12. These settings are only used when NAT traversal is enabled via the `enableNAT` setting.

When the proxy server is connected to two different networks (typically the internet and a local network), it can provide NAT traversal functions that route RTP traffic appropriately between the networks. The `Proxy Media Config` page configures settings relative to the RTP traffic.

Usually, the proxy server uses SIP messaging to enable and disable RTP traffic. However, if the active proxy server node fails, the system typically loses information about the state of all active calls. When this happens, the proxy server continues to process RTP traffic, but there may be no SIP messaging to end that traffic when the call ends. Several timeouts are implemented to ensure that this situation does not lead to a situation where the RTP traffic for a given call continues indefinitely.

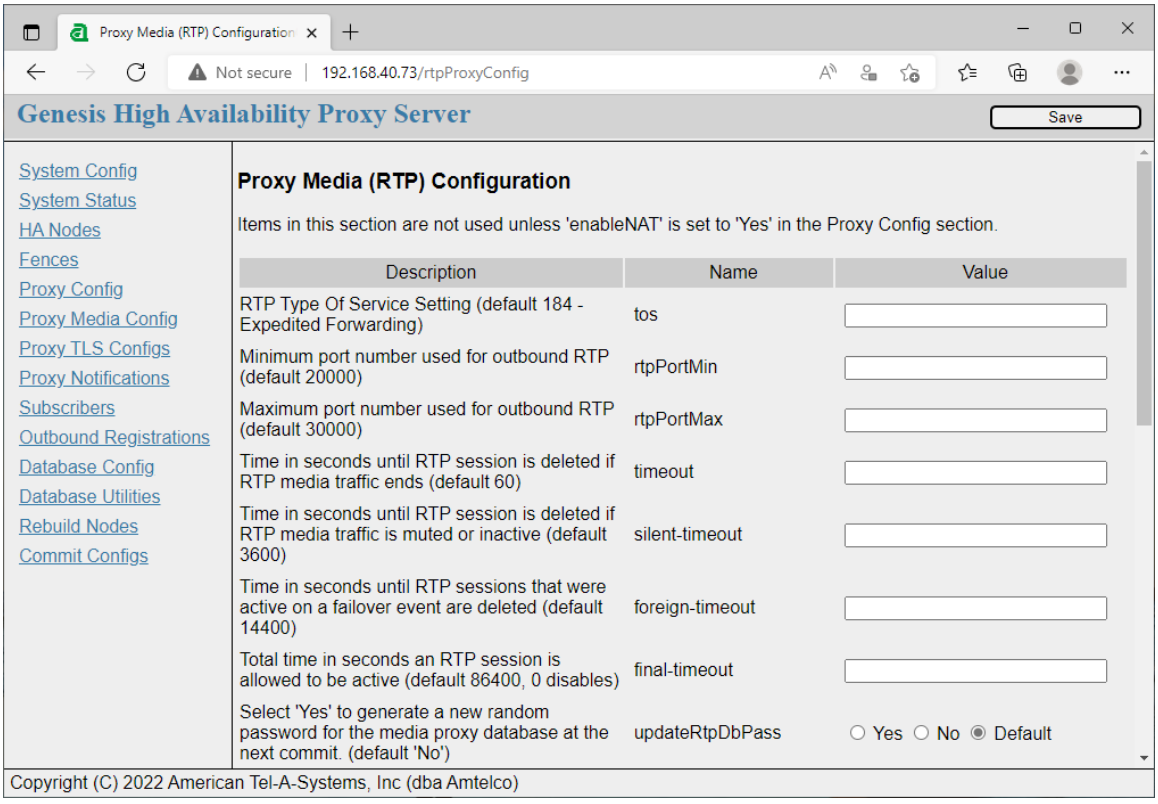

Figure 12: Proxy Media (RTP) Configuration Settings Page 1

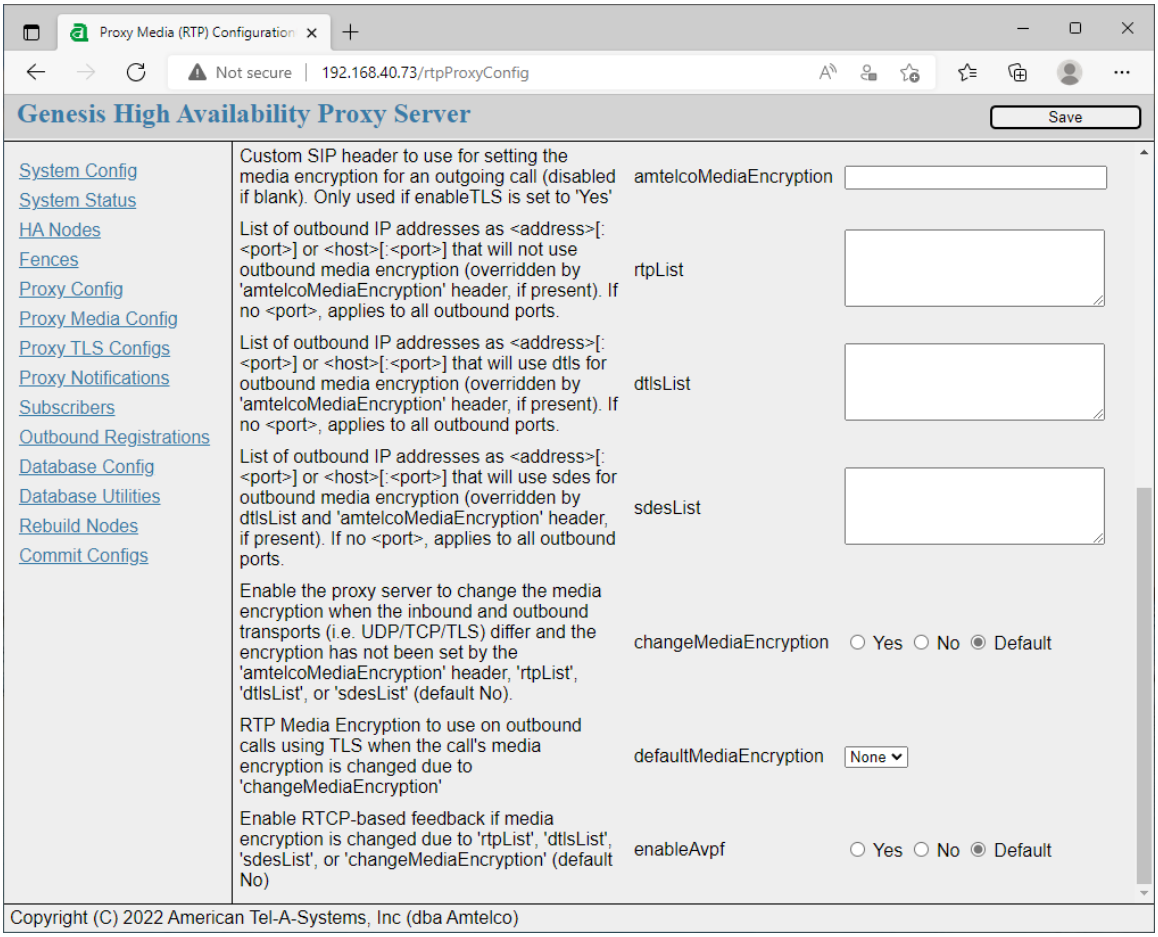

Figure 13: Proxy Media (RTP) Configuration Settings Page 2

8.1.1 tos

`tos` is used to set the Type Of Service as defined in RFC1349 for RTP packets that are transmitted by the proxy server. The value entered here is a decimal number.

8.1.2 rtpPortMin

`rtpPortMin` specifies the lowest port number used for outbound RTP traffic. It applies to both networks. This value, combined with `rtpPortMax`, determines the range of ports used for outbound RTP traffic. If a firewall or router is used, its configuration must forward traffic on these ports.

8.1.3 rtpPortMax

`rtpPortMax` specifies the highest port number used for outbound RTP traffic. It applies to both networks. This value, combined with `rtpPortMin`, determines the

range of ports used for outbound RTP traffic. If a firewall or router is used, its configuration must forward traffic on these ports.

8.1.4 timeout

`timeout` is used to set the time until an RTP session ends if no RTP media is received. The default is 60 seconds.

8.1.5 silent-timeout

`silent-timeout` is used to set the time until an RTP session ends if received RTP media traffic is muted or inactive. The default is 3600 seconds (1 hour).

8.1.6 foreign-timeout

`foreign-timeout` is used to set the time until an RTP session ends following a server failover event. The default is 14400 seconds (4 hours).

When failover occurs, the new active system marks all existing RTP sessions as foreign calls. This setting therefore allows those existing calls to continue for the specified time period. The default value is fairly long so that existing conversations are unlikely to be cut off prematurely.

8.1.7 final-timeout

`final-timeout` is used to set the maximum duration of each RTP session. The default is 86400 seconds (24 hours). This is essentially a catch-all timeout that ensures that RTP sessions do not continue indefinitely. The default value is long so that conversations are very unlikely to be cut off prematurely. This timeout is unlikely to be used, since other timeouts and mechanisms typically end RTP sessions earlier.

8.1.8 updateRtpDbPass

`updateRtpDbPass` is used to force creation of a new random password for the database used for RTP Media recovery after failover. The database uses this password to help protect against malicious access to the database.

If this field is set to '`Yes`', a new password will be generated on the next commit, and the value will be reset to '`Default`'. All nodes in the high availability system will restart, resulting in significant down time.

If the system's internet connection uses a firewall that only allows appropriate SIP and RTP ports, there is no need to activate this feature. It is primarily intended as a convenience.

8.1.9 amtelcoMediaEncryption

`amtelcoMediaEncryption` is used to set a custom SIP header that can be used to force the RTP encryption mode between the proxy server and the destination endpoint. This may be useful if Genesis or a similar system is on the local network where RTP encryption is not needed, but is communicating with an endpoint on the public network that does require encryption. In this case, Genesis can insert the specified header with the desired encryption mode. If no value is specified, this feature is disabled.

For example, if `amtelcoMediaEncryption` is set to '`Amtelco.media_encryption`', the corresponding header would be like '`Amtelco.media_encryption: sdes`' to force sdes encryption of the RTP media.

Valid encryption values are '`none`', '`none_f`', '`sdes`', '`sdes_f`', '`dtls`', and '`dtls_f`'. Values with '`_f`' will set the media to use RTCP based feedback.

If TLS and NAT are both enabled on the proxy server, the encryption will be forced to the specified mode, regardless of any other settings.

8.1.10 rtpList

`rtpList` sets the outbound media for specified endpoints to unencrypted. If a call's outbound endpoint is in `rtpList` and the encryption has not been set by the `amtelcoMediaEncryption` header, the outbound media will not be encrypted. RTCP-based feedback for the call will be set as specified by `enableAvpf`.

8.1.11 dtlsList

`dtlsList` sets the outbound media for specified endpoints to use DTLS encryption. If a call's outbound endpoint is in `dtlsList` and the encryption has not been set by the `amtelcoMediaEncryption` header or by `rtpList`, the outbound media will be encrypted using DTLS. RTCP-based feedback for the call will be set as specified by `enableAvpf`.

8.1.12 sdesList

`sdesList` sets the outbound media for specified endpoints to use SDES encryption. If a call's outbound endpoint is in `sdesList` and the encryption has not been set by the `amtelcoMediaEncryption` header or by `rtpList` or `dtlsList`, the outbound media will be encrypted using SDES. RTCP-based feedback for the call will be set as specified by `enableAvpf`.

8.1.13 changeMediaEncryption

`changeMediaEncryption` is used to enable the proxy server to select the media encryption on the outbound leg of the call when the inbound and outbound transports differ and the encryption is not set by the `amtelcoMediaEncryption` header or by `rtpList`, `dtlsList`, or `sdesList`. If `changeMediaEncryption` is disabled, the outbound encryption will be the same as the inbound encryption. If `changeMediaEncryption` is enabled, the encryption will be set as shown in the following table:

Inbound SIP Transport	Outbound SIP Transport	Outbound Media Encryption
UDP/TCP	UDP/TCP	Same as inbound media encryption
UDP/TCP	TLS	Set to the encryption specified by `defaultMediaEncryption`
TLS	UDP/TCP	No encryption
TLS	TLS	Same as inbound media encryption

RTCP-based feedback for the call will be set as specified by `enableAvpf`.

8.1.14 defaultMediaEncryption

`defaultMediaEncryption` sets the encryption used by `changeMediaEncryption`.

8.1.15 enableAvpf

`enableAvpf` enables RTCP based feedback for outbound calls if the encryption is set via `rtpList`, `dtlsList`, `sdesList`, or `changeMediaEncryption`.

9.0 Proxy TLS Configuration

The proxy server includes options for using TLS for various notifications, dispatcher control, and SIPS communication.

The `Proxy TLS Config` link is used to configure parameters for TLS whenever TLS support is needed.

There are several aspects of TLS that make configuration challenging. These include:

- SIP endpoints act as both clients and servers, so require both private keys/certificates when acting as server and certificates for all servers it will connect to as client.
- SIP endpoints can request client verification when acting as server, so client configuration may require private keys/certificates.
- SIP does not support wildcards in domain names, so it may be necessary to obtain a certificate specifically for the SIPS endpoint.
- If private keys/certificates are used, the certificates need to be available on all systems that will use SIPS to communicate with the server that owns the private key.
- Certificate lists must be maintained so that expired certificates are renewed when needed.

For the high availability proxy server, certificates must be maintained identically on all of the nodes in the system. Generally, the private key(s) for server side operation will need to be copied to all nodes and maintained on all nodes.

The proxy server requires private keys to be installed in /etc/ssl/genproxy_pvt. This directory's group must be 'genproxy' and have group read privileges. The keys in the genproxy_pvt directory must also have their group set to 'genproxy' and have group read privileges. This requirement ensures that the proxy server software can access the keys as necessary, but still limits access to private keys for most users.

At a very simplistic level, keys and certificates must be available as follows:

- Every SIPS endpoint (proxy server node, Genesis, SIP provider, telephone, etc.) must have a private key, either from a self-signed certificate or a certificate authority. For the proxy server, this key is installed in /etc/ssl/genproxy_pvt on every server node.
- Every SIPS endpoint must have access to certificates for every SIPS endpoint it will access. This access may be via the system-wide certificate list, via a specific certificate file, or via a specific certificate hash list.

Details for installing and maintaining private keys and certificates is beyond the scope of this document.

Although TLS support is available in the proxy server, its use is not recommended at this time.

9.1 Proxy TLS Configuration Index

When the `Proxy TLS Config` link is selected, the web interface displays an index of configured TLS sections that includes options for acting on each section and creating new sections. See Figure 14.

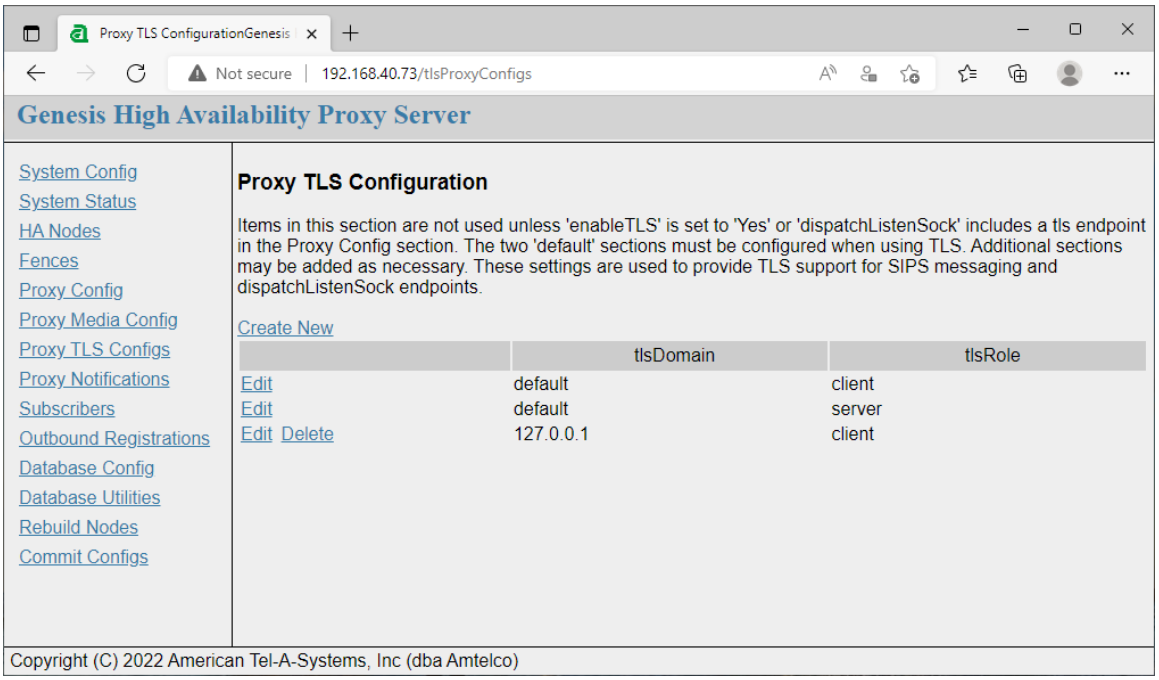

Figure 14: Proxy Server TLS Configuration Index

There are always two sections in the `Proxy TLS Configuration Index`: a default client and a default server. These sections cannot be deleted. In many cases, there will not be a need for any sections other than the defaults. Configuration and use of other sections is beyond the scope of this document.

The `Proxy TLS Configuration Index` includes a `tlsDomain` column and a `tlsRole` column. These uniquely identify each entry in the TLS Configuration section. `tlsRole` indicates whether the corresponding section is for server side or client side. Since SIPS endpoints act as both clients and servers, each domain will typically include both client and server roles.

The index page includes several actions for maintaining TLS configuration sections. These are ‘Create New’, ‘Edit’, ‘Delete’, and ‘UnDelete’ as described below.

9.1.1 Create New

Create New is used to create a new TLS configuration section. When selected, a configuration screen is displayed with all of the configuration parameters needed for a new TLS configuration section. See Section 9.2: Proxy TLS Configuration Settings.

9.1.2 Edit

Edit is used to edit an existing TLS Configuration section. Selecting Edit on a line in the index will open the edit window for that line’s section. See Section 9.2: Proxy TLS Configuration Settings.

9.1.3 Delete

Delete is used to delete an existing TLS Configuration section. Selecting Delete on a line in the index will remove that line’s section. The two default sections cannot be deleted, and therefore do not have a ‘Delete’ action.

9.1.4 UnDelete

If a TLS Configuration section is deleted after it has been committed, the web interface marks it for deletion so that it will be permanently removed at the next commit. Until then, it will still be displayed in the index, but the only available action is ‘UnDelete’. If ‘UnDelete’ is selected, the corresponding TLS Configuration section’s deletion flag will be cleared and the section will be returned to the list of available TLS Configurations.

9.2 Proxy TLS Configuration Settings

If Create New or Edit is selected from the Proxy TLS Configuration Index, a configuration page is displayed for the appropriate new or existing TLS Configuration section. See Figure 15.

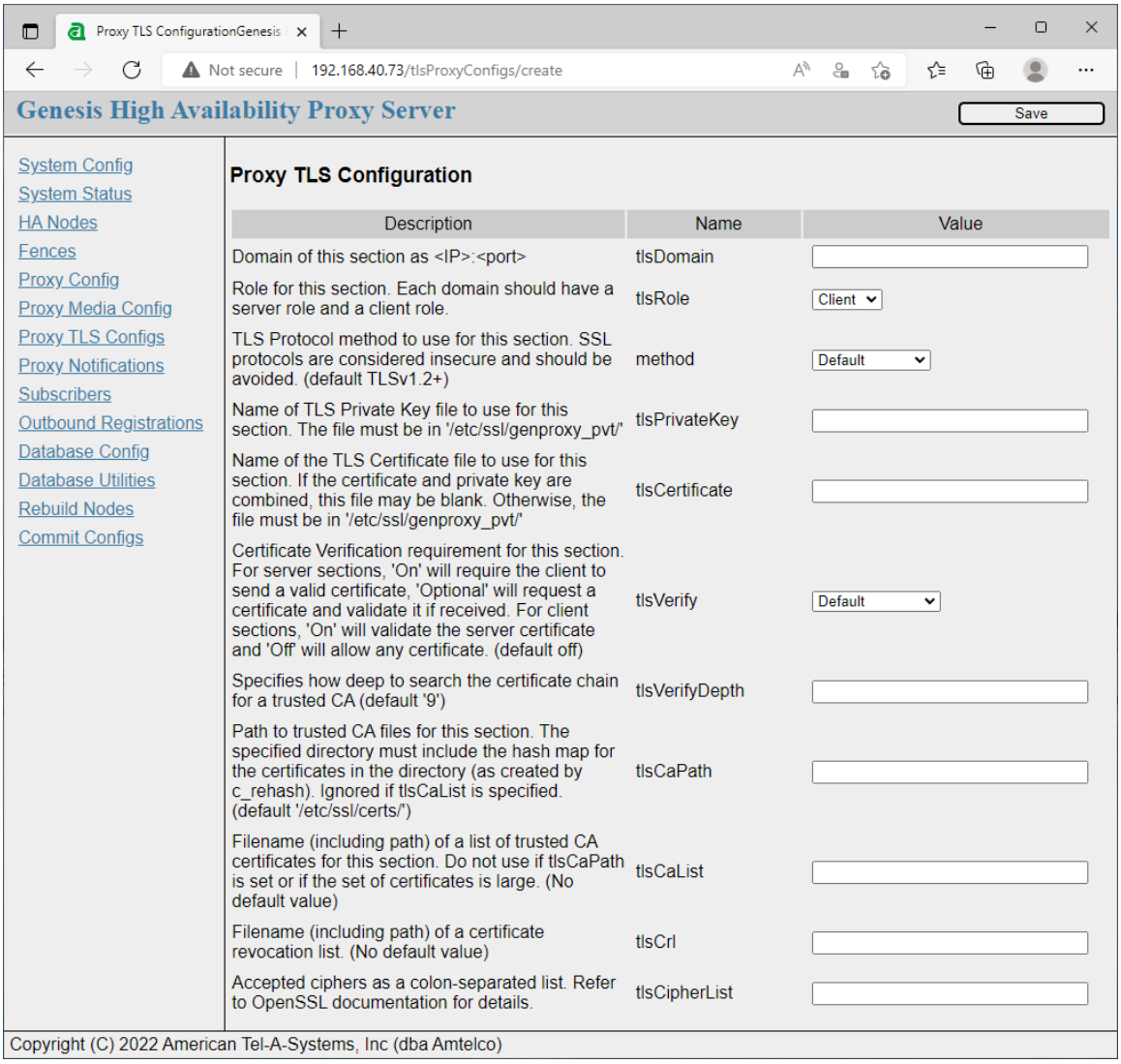

Figure 15: Proxy Server TLS Configuration Settings

9.2.1 tlsDomain

`tlsDomain` is the domain for this TLS Configuration section as `<IP Address>:<port>`. Both the address and port are required. The combination of `tlsDomain` and `tlsRole` must be unique.

9.2.2 tlsRole

`tlsRole` is the role for this TLS Configuration section. The options are '`client`' (for client side configuration) and '`server`'(for server side configuration). The combination of `tlsDomain` and `tlsRole` must be unique.

9.2.3 method

`method` specifies the TLS version to use for this section. The default value (TLS V1.2+) should be acceptable for most installations. Note that earlier versions of TLS (V1.0, V1.1) as well as the SSL options are not considered secure.

9.2.4 tlsPrivateKey

`tlsPrivateKey` is the name of the TLS Private Key file to use for this section. If `tlsRole` is '`Client`', this parameter is not required unless the server side will require client verification. The key file must be in `/etc/ssl/genproxy_pvt`. The `genproxy` group must have read access to the file.

Note that the proxy server always requires a private key when TLS is enabled. To meet this requirement, a default private key/certificate file will be used if none is specified here. If the system will act as a server or is to support client verification, the default file should not be used.

9.2.5 tlsCertificate

`tlsCertificate` is the name of the TLS Certificate file to use for this section. If the certificate and private keys are combined into one file, this line may be blank. The certificate file must be in `/etc/ssl/genproxy_pvt`. The `genproxy` group must have read access to the file.

Note that the proxy server always requires a certificate when TLS is enabled. To meet this requirement, a default private key/certificate file will be used if none is specified here. If the system will act as a server or is to support client verification, the default file should not be used.

9.2.6 tlsVerify

`tlsVerify` selects the verification requirements for this section.

If `tlsRole` is set to '`Client`', the verification requirements are:

- `Default`: Any certificate is accepted.
- `Off`: Any certificate is accepted.
- `On`: The certificate is verified against the certificate list determined by `tlsCaPath` and `tlsCaList`.
- `Optional`: Not applicable for clients. Probably behaves as '`On`'.
- `Optional No CA`: Not applicable for clients. Probably behaves as '`On`'.

If `tlsRole` is set to '`Server`', the verification requirements are:

- `Default`: No client certificate is requested.
- `Off`: No client certificate is requested.
- `On`: Clients must provide a certificate. The certificate is verified against the certificate list determined by `tlsCaPath` and `tlsCaList`.
- `Optional`: A client certificate is requested. If provided, the certificate is verified against the certificate list determined by `tlsCaPath` and `tlsCaList`. This selection is of limited usefulness. However, it might be useful if some endpoints are known to provide certificates and verification of those certificates is desired.
- `Optional No CA`: A client certificate is requested, but does not need to be provided and is not verified if provided. This selection is of limited (if any) usefulness.

9.2.7 tlsVerifyDepth

`tlsVerifyDepth` specifies how deep to search the certificate chain. The default value of `9` is probably appropriate for most applications.

9.2.8 tlsCaPath

`tlsCaPath` is the directory that contains a list of certificates to use for certificate verification. The directory must contain symbolic links to hash files for the certificates as created by `c_rehash` or equivalent. If `tlsCaList` is specified, this parameter is ignored. If neither `tlsCaPath` nor `tlsCaList` is specified, `/etc/ssl/certs` will be searched for the certificate.

This parameter could be used to limit the number of certificates that are allowed. In some cases, the default system list may not provide adequate security. If this parameter is used, copy the required certificates to the specified directory and then issue the command '`sudo c_rehash <tlsCaPath>`', where `<tlsCaPath>` is the path used for `tlsCaPath`. This must be done on all proxy server nodes. (Note that if no directory is provided, `c_rehash` creates the hashes for `/etc/ssl/certs`.)

9.2.9 tlsCaList

`tlsCaList` is used to specify a file that contains all certificates to allow for TLS verification. The entry must include the entire path. If `tlsCaList` is specified, `tlsCaPath` is ignored. When using this parameter, concatenate all of the required certificates into the specified file. The file must be installed in the specified location on all proxy server nodes. Do not use if the number of certificates required is large—use `tlsCaPath` instead.

9.2.10 tlsCrl

`tlsCrl` is used to specify a certificate revocation list. The entry must include the entire path. This parameter is only required if there is a need to use a certificate revocation list on the system. The file must be installed in the specified location on all proxy server nodes.

9.2.11 tlsCipherList

`tlsCipherList` is used to specify the list of ciphers that are to be accepted. Usually, the default cipher list is adequate, so there is no need to use this parameter.

10.0 Proxy Notifications

The proxy server can be configured to send notifications of system status events to one or more http(s) servers.

The `Proxy Notifications` link is used to specify where to send notifications along with appropriate parameters for each destination.

Notification events provide information about changes in the high availability proxy system. Notifications may relate to dispatcher state (i.e. that the system has assigned a new primary dispatch set), proxy server node state (i.e. that a node has gone offline or online), or fence state (i.e. that a fence device is active).

The proxy server uses http(s) `POST` requests with JSON formatted data indicating the event. The details of these messages are described elsewhere.

Although the system supports use of TLS (https), its use is not recommended at this time.

10.1 Proxy Notifications Index

When the `Proxy Notifications` link is selected, the web interface displays a list of configured notification destinations. See Figure 16.

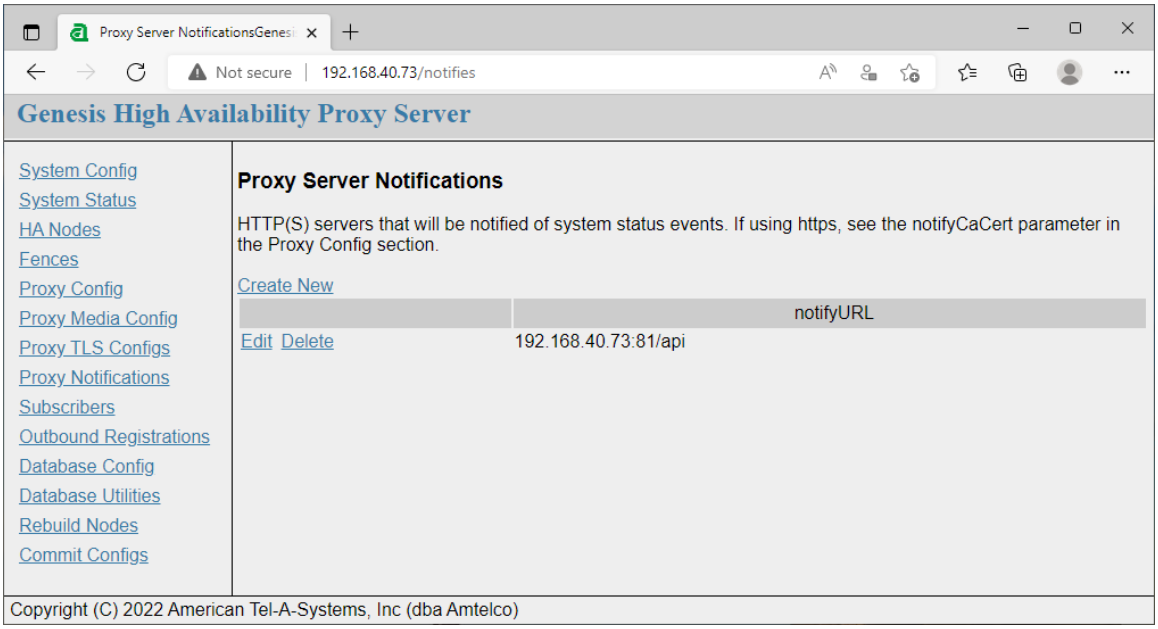

Figure 16: Notifications Index

The Proxy Notifications Index includes a notifyURL column that identifies each configured notification endpoint.

The index page includes several actions for maintaining notifications. These are ‘Create New’, ‘Edit’, ‘Delete’, and ‘UnDelete’ as described below.

10.1.1 Create New

Create New is used to create a new notification destination. When selected, a configuration screen is displayed with all of the configuration parameters needed for a new notification. See Section 10.2: Proxy Notifications Settings.

10.1.2 Edit

Edit is used to edit an existing notification. Selecting Edit on a line in the index will open the edit window for that line’s notification. See Section 10.2: Proxy Notifications Settings.

10.1.3 Delete

Delete is used to delete an existing notification. Selecting Delete on a line in the index will remove that line’s notification.

10.1.4 UnDelete

If a notification is deleted after it has been committed, the web interface marks it for deletion so that it will be permanently removed at the next commit. Until then, it will still be displayed in the index, but the only available action is ‘UnDelete’. If ‘UnDelete’ is selected, the corresponding notification’s deletion flag will be cleared and the notification will be returned to the list of available notifications.

10.2 Proxy Notifications Settings

If Create New or Edit is selected from the Proxy Notifications Index, a configuration page is displayed for the appropriate new or existing notification. See Figure 17.

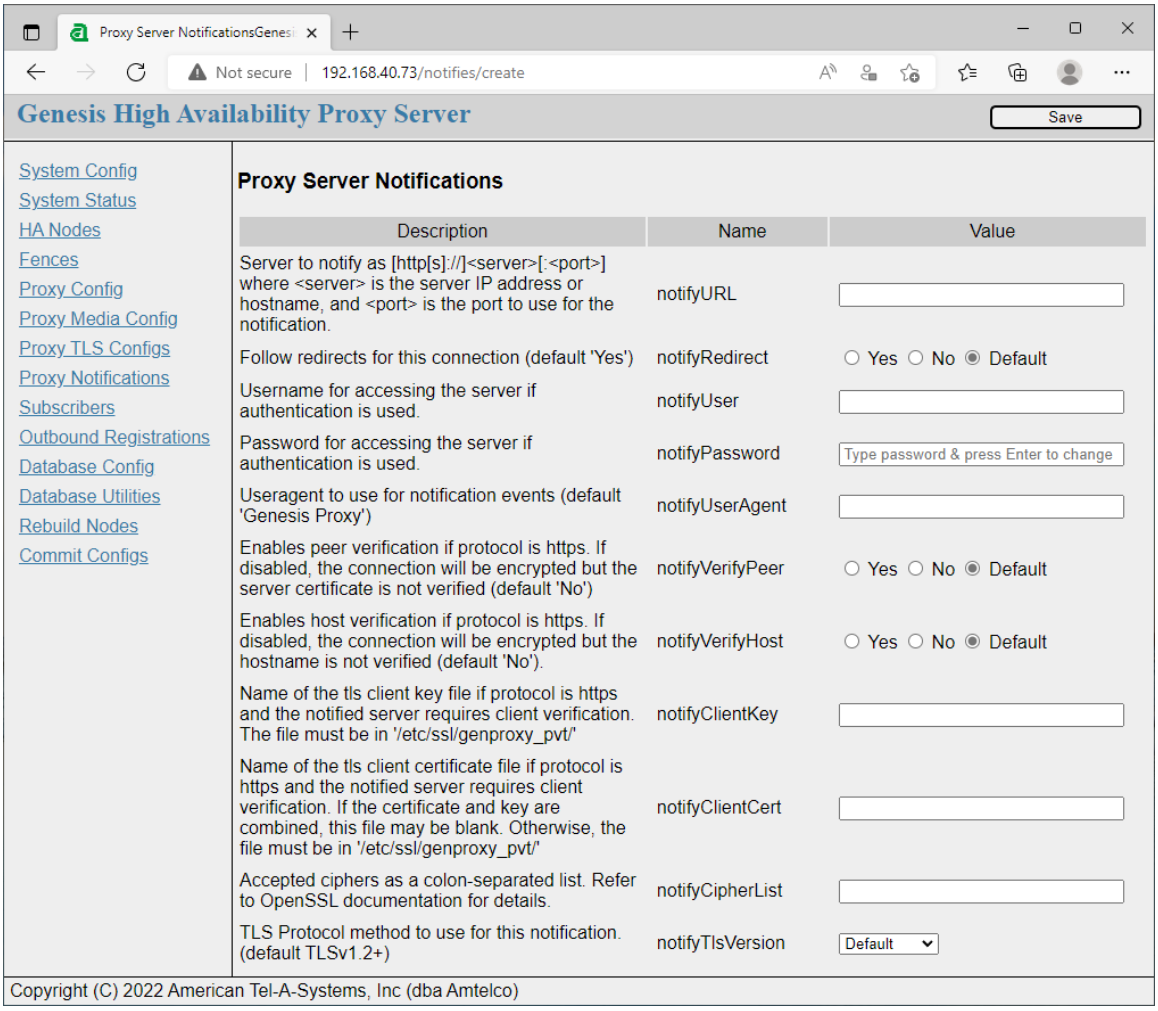

Figure 17: Notifications Settings

10.2.1 notifyURL

`notifyURL` is the URL for the server to send notifications to. The format is `[http[s]://]<server>[:<port>]`, where `<server>` is the server's IP address or hostname, and `<port>` is the port to use for the notification. If the port is not specified, port 80 will be used for http and port 443 for https.

10.2.2 notifyRedirect

`notifyRedirect` selects whether or not the notification client will follow redirects. The default value (`Yes`) should be acceptable for most situations. However, if TLS is not enabled, redirects to an https server will fail. In general, it is probably best to avoid situations where a redirect is expected.

10.2.3 notifyUser

notifyUser is the user for servers that require authentication. If authentication is not used, notifyUser can be blank. (Both BASIC and DIGEST authentication are supported.)

10.2.4 notifyPassword

notifyPassword is the password for servers that require authentication. If authentication is not used, notifyPassword can be blank.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

10.2.5 notifyUserAgent

notifyUserAgent is used to specify the User-Agent: header for the notification. If blank, 'Genesis Proxy' will be used.

10.2.6 notifyVerifyPeer

notifyVerifyPeer is used to select whether or not the server's certificate is verified for the notification when TLS (https) is used. If set to 'Yes', the server's certificate will be verified with the system's certificate list, or with the certificates specified by Section 7.1.10: notifyCaCert. (Default is 'No'.)

10.2.7 notifyVerifyHost

notifyVerifyHost is used to select whether or not the server's hostname is verified for the notification when TLS (https) is used. This setting requires a valid certificate from the server, so is only appropriate if notifyVerifyPeer is enabled.

10.2.8 notifyClientKey

notifyClientKey specifies the notification's private key file when TLS (https) is used and the notified server requires client verification. The key file must be in /etc/ssl/genproxy_pvt. The genproxy group must have read access to the file.

10.2.9 notifyClientCert

`notifyClientCert` specifies the notification's certificate file when TLS (https) is used and the notified server requires client verification. If the certificate and private key file are combined, this parameter may be blank. The file must be in `/etc/ssl/genproxy_pvt`. The `genproxy` group must have read access to the file.

10.2.10 notifyCipherList

`notifyCipherList` is used to specify the list of ciphers that are to be accepted when TLS (https) is used. Usually, the default cipher list is adequate, so there is no need to use this parameter.

10.2.11 notifyTlsVersion

`notifyTlsVersion` specifies the TLS version to use for the notification when TLS (https) is used. The default value (TLSv1.2+) should be acceptable for most installations.

11.0 Subscribers

The proxy server allows users to register and also incorporates authentication capabilities for inbound `REGISTER` and `INVITE` requests.

The `Subscribers` link provides an interface for defining these users.

11.1 Subscribers Index

When the `Subscribers` link is selected, the web interface displays an index of all defined subscribers. These subscribers can register with the proxy server. See Figure 18.

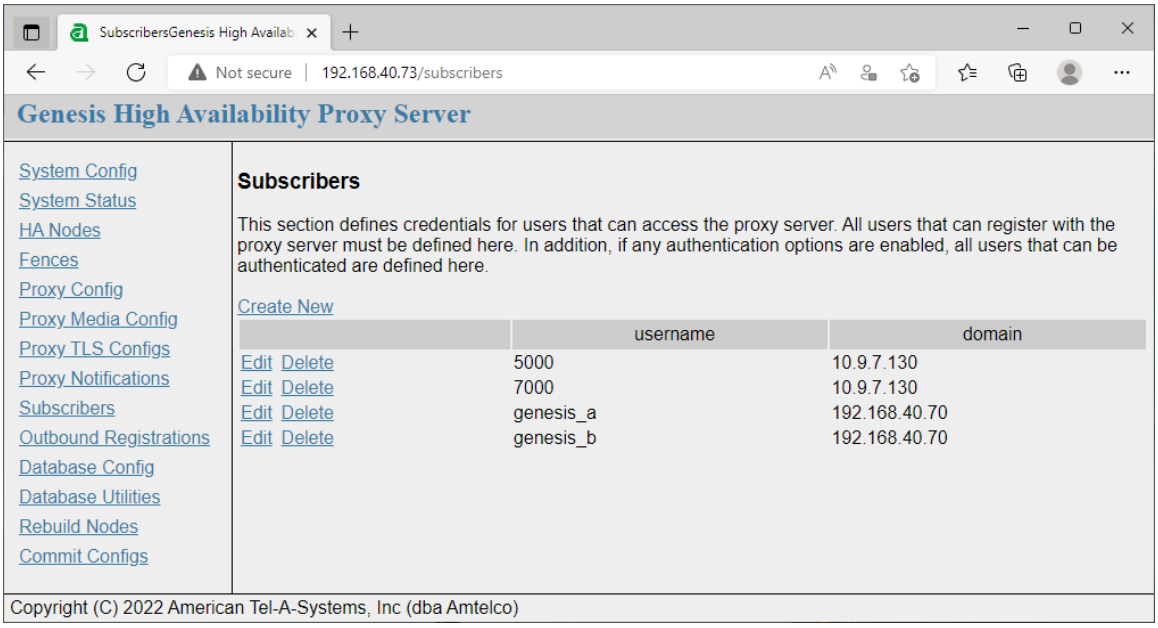

Figure 18: Subscribers Index

The `Subscribers Index` includes `username` and `domain` columns that uniquely identify the subscriber's address-of-record.

The index page includes several actions for maintaining subscribers. These are '`Create New`', '`Edit`', '`Delete`', and '`UnDelete`' as described below.

11.1.1 Create New

`Create New` is used to create a new subscriber. When selected, a configuration screen is displayed with all of the configuration parameters needed to allow the corresponding

subscriber to REGISTER and/or authenticate inbound REGISTER or INVITE requests. See Section `11.2: Subscribers Settings`.

11.1.2 Edit

`Edit` is used to modify an existing subscriber. Selecting `Edit` on a line in the index will open the edit window for that line's subscriber. See Section `11.2: Subscribers Settings`.

11.1.3 Delete

`Delete` is used to delete an existing subscriber. Selecting `Delete` on a line in the index will remove that line's subscriber.

11.1.4 UnDelete

If a subscriber is deleted after it has been committed, the web interface marks it for deletion so that it will be permanently removed at the next commit. Until then, it will still be displayed in the index, but the only available action is '`UnDelete`'. If '`UnDelete`' is selected, the corresponding subscriber's deletion flag will be cleared and the subscriber will be returned to the list of available subscribers.

11.2 Subscribers Settings

If `Create New` or `Edit` is selected from the `Subscribers Index`, a configuration page is displayed for the appropriate new or existing subscriber. See Figure 19.

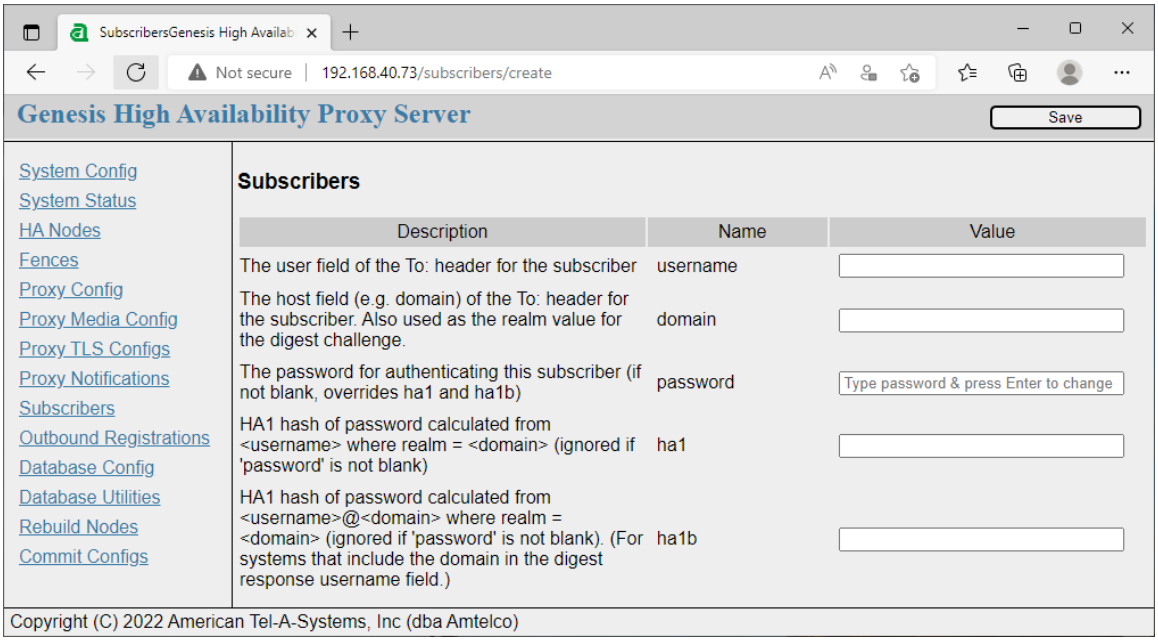

Figure 19: Subscribers Settings

11.2.1 username

`username` is the user field of the address-of-record for this subscriber (i.e. the user field of the `To:` header).

11.2.2 domain

`domain` is the host field of the address of record for this subscriber (i.e. the host field of the `To:` header).

11.2.3 password

`password` is the password that the proxy server will use if inbound registration and/or INVITE authentication are enabled via `enableRegAuth` and `enableInviteAuth`. If `password` is specified, the subscriber settings `ha1` and `ha1b` are ignored. Note: subscriber authentication does not expect the address of record to include the password.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

11.2.4 ha1

`ha1` is the HA1 password hash to use if inbound registration and/or INVITE authentication are enabled via `enableRegAuth` and `enableInviteAuth`. This field is only used if `password` is blank. The hash is the MD5 checksum calculated from: `<username>:<domain>:<password>` (where `<username>` and `<domain>` are the corresponding fields in this subscriber and `<password>` is the password assigned for the authentication).

11.2.5 ha1b

`ha1b` is the HA1B password hash to use if inbound registration and/or INVITE authentication are enabled via `enableRegAuth` and `enableInviteAuth`. This field is only used if `password` is blank. The hash is the MD5 checksum calculated from: `<username>@<domain>:<domain>:<password>` (where `<username>` and `<domain>` are the corresponding fields in this subscriber and `<password>` is the password assigned for the authentication).

12.0 Outbound Registrations

The proxy server includes registration capabilities so that it can register with other devices. This feature may be necessary when the system uses a SIP provide to provide VoIP services and that provide requires registration.

The `Outbound Registrations` link provides an interface for defining and editing parameters for registering with other devices.

12.1 Outbound Registrations Index

When the `Outbound Registrations` link is selected, the web interface displays an index of all SIP endpoints that can register with the proxy server. See Figure 20.

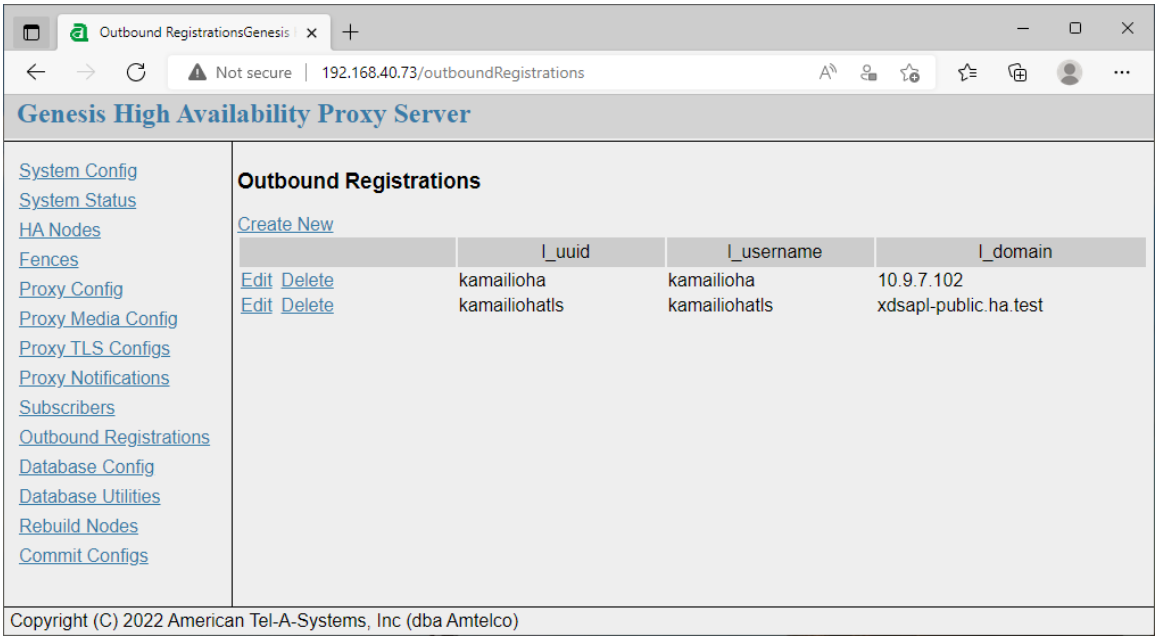

Figure 20: Outbound Registrations Index

The `Outbound Registrations Index` includes `l_uuid`, `l_username`, and `l_domain` columns. The `l_uuid` column uniquely identifies the outbound registration and the other two columns are included to provide a more user-friendly means of identifying the registration.

The index page includes several actions for maintaining Outbound Registrations. These are '`Create New`', '`Edit`', '`Delete`', and '`UnDelete`' as described below.

12.1.1 Create New

Create New is used to create a new Outbound Registration. When selected, a configuration screen is displayed with all of the configuration parameters needed to allow the proxy server to register with the corresponding SIP device. See Section 12.2: Outbound Registrations Settings.

12.1.2 Edit

Edit is used to modify an existing Outbound Registration. Selecting Edit on a line in the index will open the edit window for that line's Outbound Registration. See Section 12.2: Outbound Registrations Settings.

12.1.3 Delete

Delete is used to delete an existing Outbound Registration. Selecting Delete on a line in the index will remove that line's Outbound Registration.

12.1.4 UnDelete

If an Outbound Registrations is deleted after it has been committed, the web interface marks it for deletion so that it will be permanently removed at the next commit. Until then, it will still be displayed in the index, but the only available action is 'UnDelete'. If 'UnDelete' is selected, the corresponding Outbound Registration's deletion flag will be cleared and the Outbound Registration will be returned to the list of available Outbound Registrations.

12.2 Outbound Registrations Settings

If Create New or Edit is selected from the Outbound Registrations Index, a configuration page is displayed for the appropriate new or existing Outbound Registration. See Figure 21.

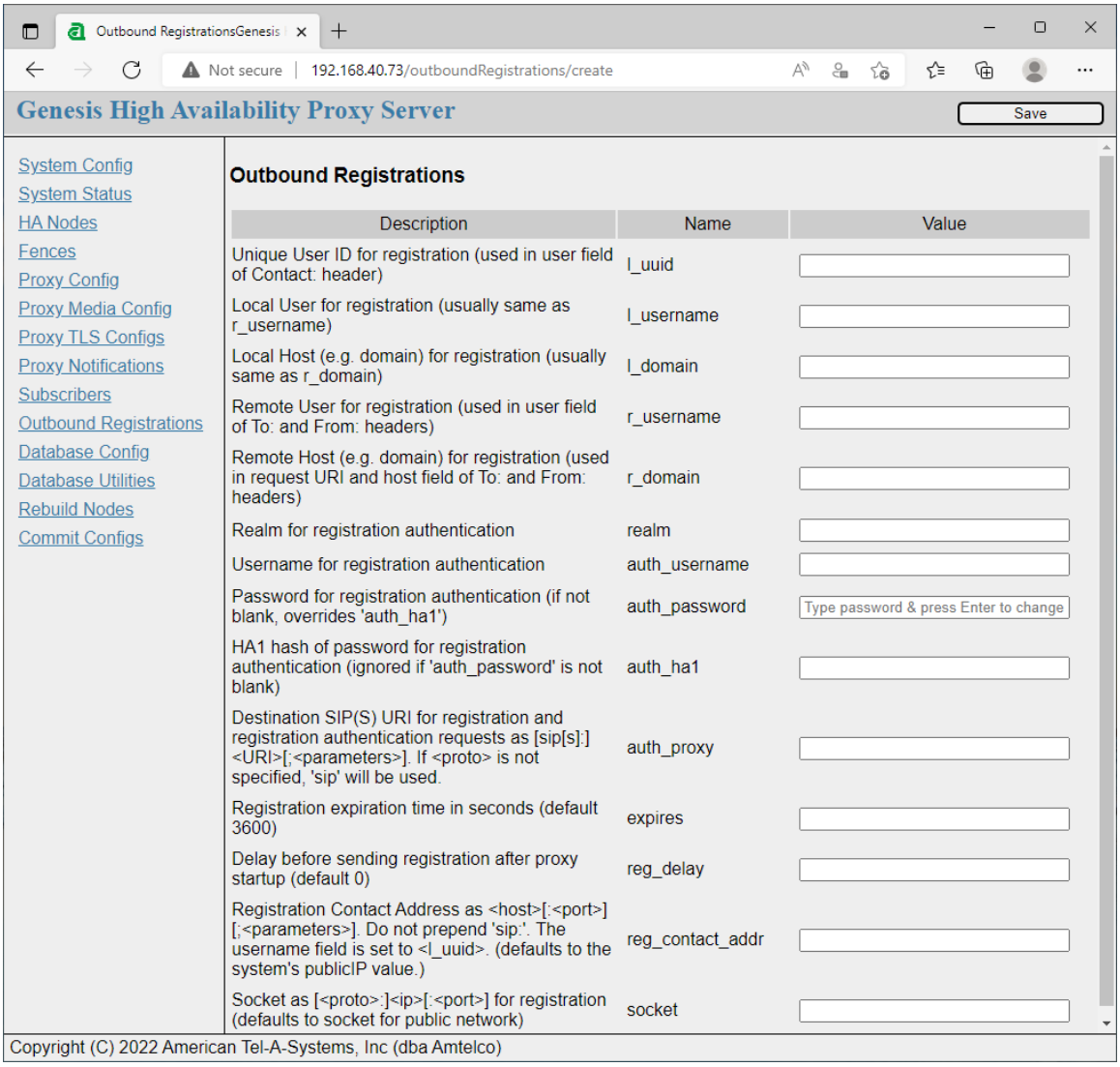

Figure 21: Outbound Registrations Settings

12.2.1 l_uuid

`l_uuid` is a local unique user id. It must be unique within the set of all outbound registration entries. `l_uuid` is used as the username portion of the `Contact:` URI for the registration.

12.2.2 l_username

`l_username` is used for identifying the registration in the proxy server. For most situations, set it to the same value as `r_username`.

12.2.3 l_domain

`l_domain` is used for identifying the registration in the proxy server. For most situations, set it to the same value as `r_domain`.

12.2.4 r_username

`r_username` is the user field of the address-of-record for the registration. It is used as the user field in the `To:` and `From:` headers of the registration. For most situations, set it to the same value as `l_username`.

12.2.5 r_domain

`r_domain` is the host field of the address-of-record for the registration. It is used as the host field in the `Request URI` and in the `To:` and `From:` headers of the registration. For most situations, set it to the same value as `l_domain`.

12.2.6 realm

`realm` is the realm used when authentication is required for registration. It is used to calculate the `ha1` hash of the authentication password and must match the realm that the registrar returns when requesting authentication.

12.2.7 auth_username

`auth_username` is the username that the proxy server will use when authentication is required for this registration.

12.2.8 auth_password

`auth_password` is the password that the proxy server will use when authentication is required for this registration If `auth_password` is specified, the outbound registration setting `auth_ha1` is ignored.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

12.2.9 auth_ha1

auth_ha1 is the HA1 password hash to use when authentication is required for this registration. This field is only used if auth_password is blank. The hash is the MD5 checksum calculated from: <auth_username>:<realm>:<password> (where <auth_username> and <realm> are the corresponding fields in this registration and <password> is the password the registrar expects for the authentication).

12.2.10 auth_proxy

auth_proxy is the destination SIP or SIPS URI that REGISTER requests will be sent to as [<proto>:]<URI>[;<parameters>], where <proto> is 'sip' or 'sips'. If <proto> is not specified, 'sip' will be assumed. The current implementation does include support for SIPS. However, use of SIPS is not recommended at this time.

In order to use TCP or TLS for the registration, <parameters> must be specified here. Using a <parameters> value of 'transport=tcp' will force the registration to use TCP as its transport or 'transport=tls' will force the registration to use TLS as its transport.

Note that specifying 'sips' as the protocol does not seem to have much effect. The protocol for the To:, From:, and Contact: headers is always set to 'sip'. If using TLS, this may not work with some endpoints.

12.2.11 expires

expires is the expiration time (in seconds) that will be inserted in the Expires: header of the REGISTER request. The default is 3600 seconds (1 hour).

12.2.12 reg_delay

reg_delay specifies the delay (in seconds) after the proxy server starts before it will send the REGISTER request. The default is 0, which will send the request as soon as possible after startup.

12.2.13 reg_contact_addr

reg_contact_addr provides the host, port, and parameters to include in the Contact: header of the REGISTER request. The format is: <host>[:<port>][;<parameters>]. The value must not be prepended with a protocol (e.g. 'sip' or 'sips'). If blank, the proxy server's publicIP will be used.

The <parameters> field can be used to force calls to the proxy to use TCP or TLS. E.g. 'transport=tls' will force TLS and 'transport=tcp' will force TCP.

12.2.14 socket

socket specifies which socket the proxy server should use for REGISTER requests. The format is [<proto>:]<ip>[:<port>]. If no socket is specified, REGISTER requests will be sent using the UDP socket for the public network.

In general, the <port> parameter is not required. The proxy server will select the appropriate port based on the protocol. In this case, if <proto> is blank, 'udp', or 'tcp', the privatePort, publicPort, or mappedPort specified on the Global System Configuration page will be used, depending on the socket <ip> value. Likewise, if <proto> is 'tls', the appropriate privatePortTLS, publicPortTLS, or mappedPortTLS will be used. If TLS is used, however, it is best to explicitly set the port. Otherwise the proxy server will log a warning every time it registers.

Conversely, if <proto> is not specified, but <port> is specified, the protocol will be set based on whether the port is the TLS port or not. If a port that supports TLS is specified, the protocol will be set to 'tls', otherwise it will be 'udp'. Note that if the socket is not TLS, the socket listens for both UDP and TCP. There is no need to individually select one or the other.

If the registration is for a device on the network associated with the system's privateIP, <ip> should be set to the privateIP value. If the registration is for a device on the network associated with the system's publicIP, <ip> should be set to the mappedIP value if mappedIP is set and to the publicIP value otherwise.

13.0 Database Configuration

The proxy server uses a database to store various configuration and operational parameters. The system supports two database options: MySQL Server and Microsoft SQL Server. If MySQL Server is used, MySQL is configured on all of the proxy server nodes in a high availability arrangement. If Microsoft SQL Server is used, database accesses will use a separate Microsoft SQL Server that is not an integral part of the proxy server system.

The preferred database option is Microsoft SQL Server. The database will be administered separately from the proxy server, but must be accessible by the proxy server. The database should be configured for high availability operation using Microsoft's Always On availability groups and appropriate configuration.

Users on MySQL Server are configured as `<username>@<domain>`. When the proxy server is configured to use MySQL Server, it will create three users for each node: '`amtelco`', '`amtelcoro`', and '`replication`'. The domain values will be the IP addresses of the nodes. The passwords for these users are set via the database configuration settings.

Different configuration options are used based on which database is selected. These are described below.

The `Database Config` link provides access to the database configuration settings. See Figure 22 for the Default/Microsoft SQL Server settings and Figure 23 for the MySQL Server settings.

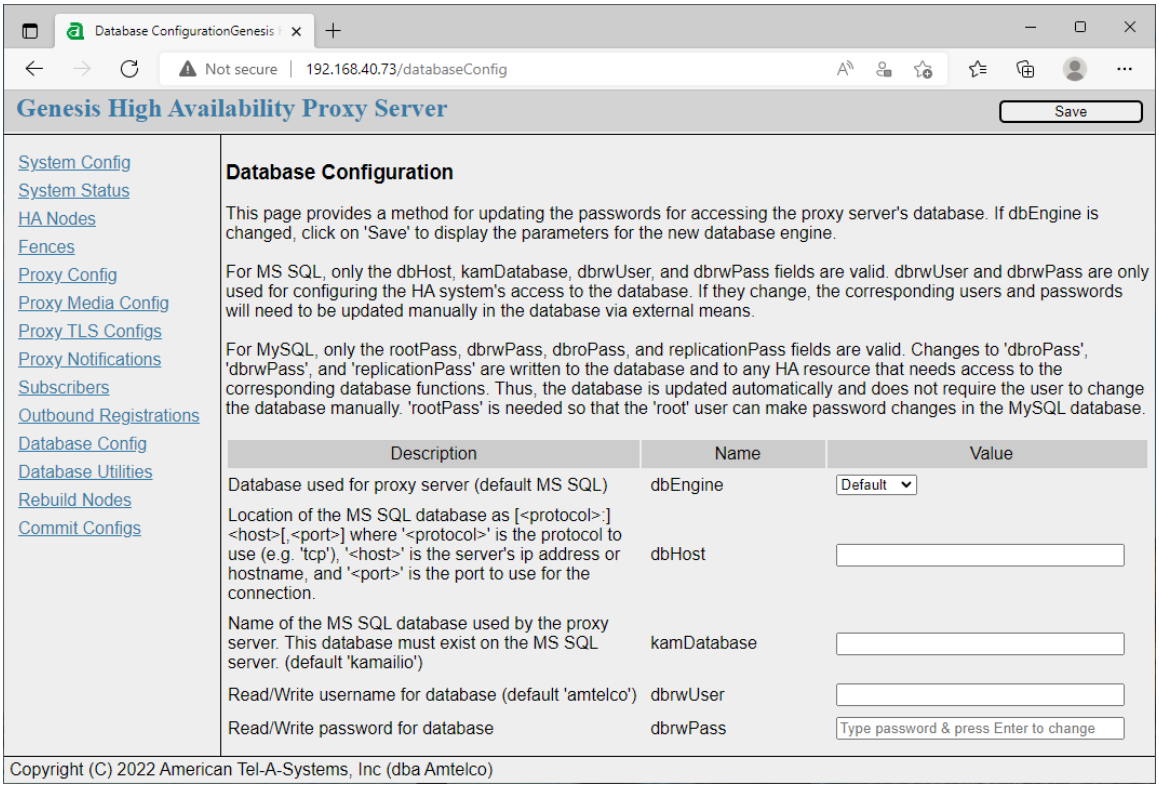

Figure 22: Database Configuration Settings -- Microsoft SQL Server

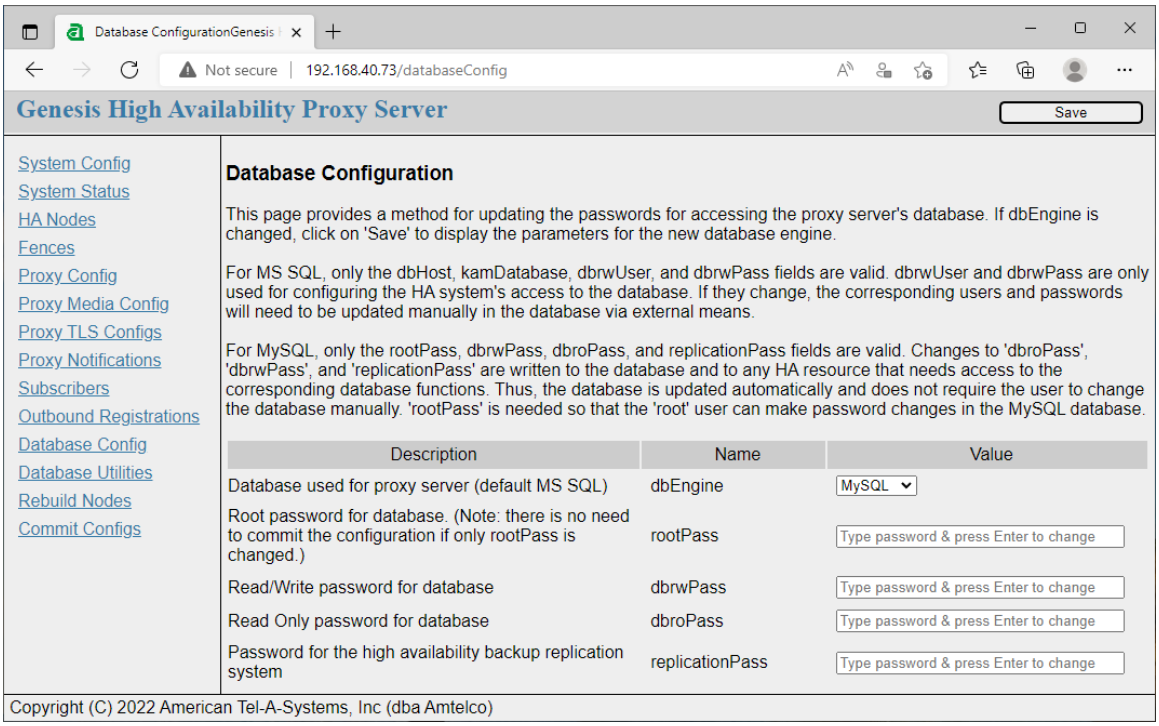

Figure 23: Database Configuration Settings -- MySQL Server

13.1.1 dbEngine

`dbEngine` selects which database engine to use, either MySQL Server or Microsoft (MS) SQL Server. The default and preferred option is Microsoft SQL Server.

The configuration options displayed for the Database Configuration depends on which database is selected. If `dbEngine` is changed, the new page will only be displayed after clicking on the `Save` button. Thus, to change the database, select the appropriate value for `dbEngine` and click on `Save`. The configuration page for the new database engine will be displayed.

13.1.2 rootPass

`rootPass` is only available when the database is MySQL Server. It is the password for the MySQL root user and must match the root password in the server. This password is necessary so that the web interface can update the other passwords used by the proxy server.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

13.1.3 dbHost

`dbHost` is only used for Microsoft SQL Server. It is the URL that provides access to Microsoft SQL Server via ODBC as `[<protocol>:]<host>[,<port>]`, where `<protocol>` is the protocol to use for accessing the server, `<host>` is the IP address or hostname of the server, and `<port>` is the port to use for the connection.

Microsoft's ODBC driver only supports '`tcp`' for its protocol, so there is no need to include the `<protocol>` parameter. If using Microsoft's default port (`1433`), `<port>` does not need to be specified.

13.1.4 kamDatabase

`kamDatabase` is only used for Microsoft SQL Server. It is the name of the database on Microsoft SQL Server that is used for proxy server functions. The database and its tables must have been built prior to committing the database configuration parameters.

13.1.5 dbrwUser

`dbrwUser` is only used for Microsoft SQL Server. It is the username that the proxy server will use to access the database specified by `kamDatabase`. This user must have read/write privileges to the database.

13.1.6 dbrwPass

`dbrwPass` is used for both MySQL Server and Microsoft SQL Server.

For Microsoft SQL Server, `dbrwPass` is the password for `dbrwUser`. It must match the Microsoft SQL Server password for that user.

For MySQL Server, `dbrwPass` is the password that will be used for the '`amtelco`' users in MySQL Server, which can read and write to the proxy server's database. This password will be updated on the MySQL Server when the system is committed.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

13.1.7 dbroPass

`dbroPass` is only used for MySQL Server. It is the password that will be used for the '`amtelcoro`' users in MySQL Server, which can only read from the proxy server's database. This password will be updated on the MySQL Server when the system is committed.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

13.1.8 replicationPass

`replicationPass` is only used for MySQL Server. It is the password that will be used for the '`replication`' users in MySQL Server, which are used to implement the

replication features required for high availability operation. This password will be updated on the MySQL Server when the system is committed.

The web interface does not send passwords to clients. Thus, password fields always show placeholder text, even if a password has previously been set. To change the password, type the new password into the field and press Enter or Tab. If a blank password is needed, select the field and press Enter without typing any test. If no password change is desired, do not type anything in the field and do not press Enter. (If a password is accidentally entered, refreshing or browsing away from the screen without clicking the Save button will eliminate the changed password.)

14.0 Database Utilites

The `Database Utilities` link provides access to a download link for retrieving the databases for the web interface and the proxy server. This feature is primarily intended for diagnostic purposes, not for backing up or restoring the database. (The backup and restore capabilities of Microsoft SQL Server are intended to be used for this purpose.)

When `Database Utilities` is selected, a screen will be displayed that provides a '`Download Database`' button (see Figure 24). Selecting this button will download a representation of the web interface and proxy server databases to the download location determined by the browser.

A Python utility script named '`dbShow`' is provided with the web server. This script can be used to display the databases in a terminal window (command prompt on Windows). Alternatively, the gzip compressed file can be uncompressed and viewed in a text editor. The utility script displays the most relevant information from the databases, but does omit some database columns. In some cases, it may be helpful to view the entire file rather than using the utility.

Figure 24: Database Utilities

15.0 Rebuild Nodes

The `Rebuild Nodes` link provides a way to completely rebuild the proxy server software on one or more nodes. This feature might be used if a node has failed and is in an unknown state or has had its operating system reinstalled.

When `Rebuild Nodes` is selected, a screen will be displayed that lists all of the nodes in the system except the current primary node. See Figure 25.

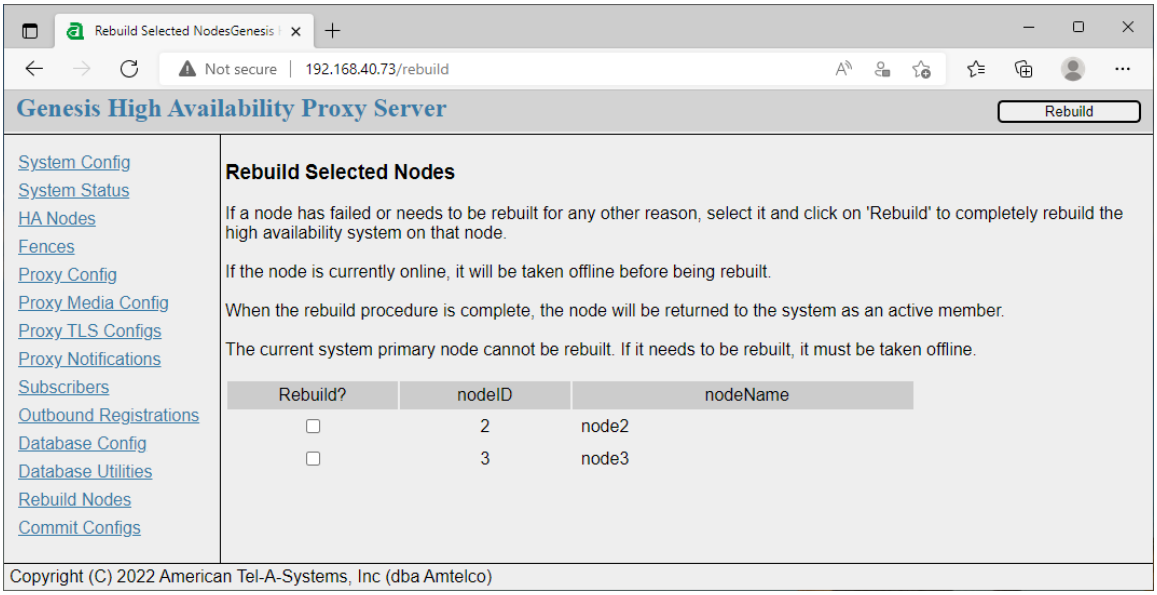

Figure 25: Rebuild Nodes

To rebuild a node, click on the checkbox in the `Rebuild?` column corresponding to that node. Multiple nodes may be selected, if desired. After making the necessary selections, click the `Rebuild` button in the upper right corner of the screen. The rebuild process will start and display progress messages as it proceeds. When the rebuild is complete, its success or failure will be indicated. See Figure 26.

If the current primary node needs to be rebuilt, it must be demoted or taken offline so another node becomes the primary node. This can be done by issuing the command '`sudo crm cluster stop`' from a terminal on that node.

If any parameters in the web interface have been modified but are not yet committed, nodes cannot be rebuilt. In this case, the `Rebuild Nodes` screen will only display a message indicating that the nodes cannot be rebuilt and will not list any nodes or include a `Rebuild` button.

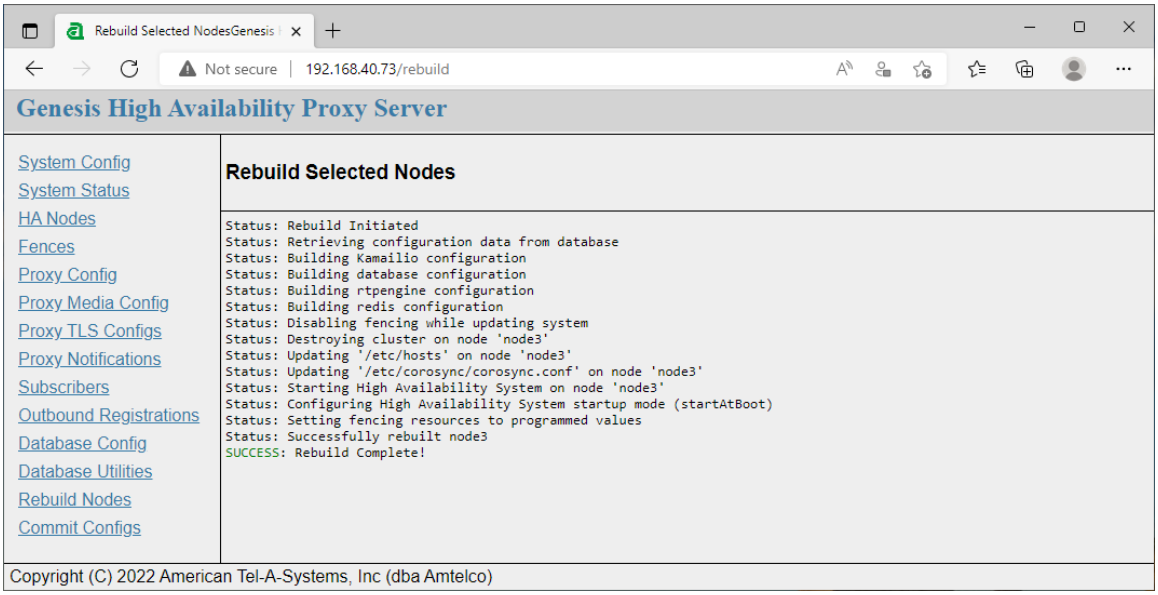

Figure 26: Rebuild Nodes Progress Page

16.0 Commit Configuration

The `Commit Config` link is used to commit all changes that have been made in the web interface to the proxy system. Changes made in the web interface are initially stored in a configuration database and have no effect on the proxy system until they are committed.

Some configuration parameters can be committed while the system is live and do not have any impact on current operation. Others may require some disruption in system operation. In some cases, the disruption is to switch primary systems, which can occur fairly quickly. However, some parameter changes require a complete shutdown and restart of the high availability software which typically takes a minute or more to complete.

When the `Commit Config` link is selected, the web interface analyzes the changes that have been made and indicates what effect the commit will have on the system's operation (see Figure 27). If the commit will require some down time, the commit should be done when call traffic is at a minimum. Ideally, even changes that don't cause down time should be made when there is minimal call traffic, but doing so is at the user's discretion.

If no configuration parameters have been changed since the previous commit, the commit process will completely rebuild the high availability proxy system. Doing so would be fairly unusual, but may be necessary under some circumstances.

Clicking on the `Commit` button in the upper right corner of this screen will start the commit process. As it proceeds, progress messages will be displayed. When complete, the success or failure of the commit will be indicated. See Figure 28.

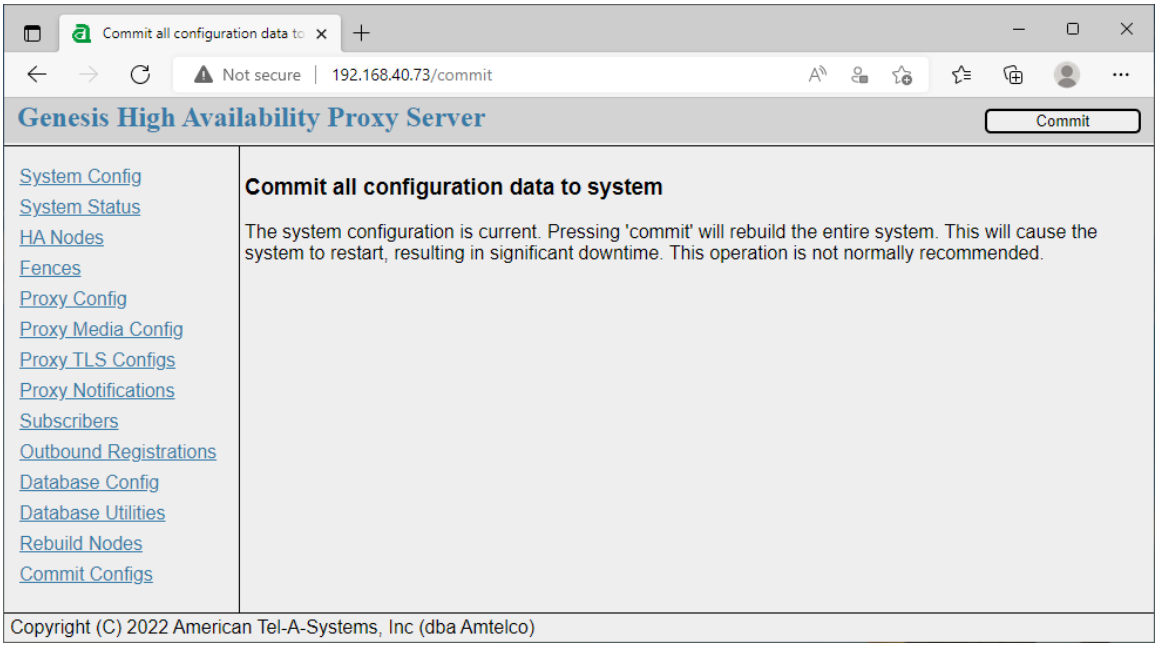

Figure 27: Commit Page

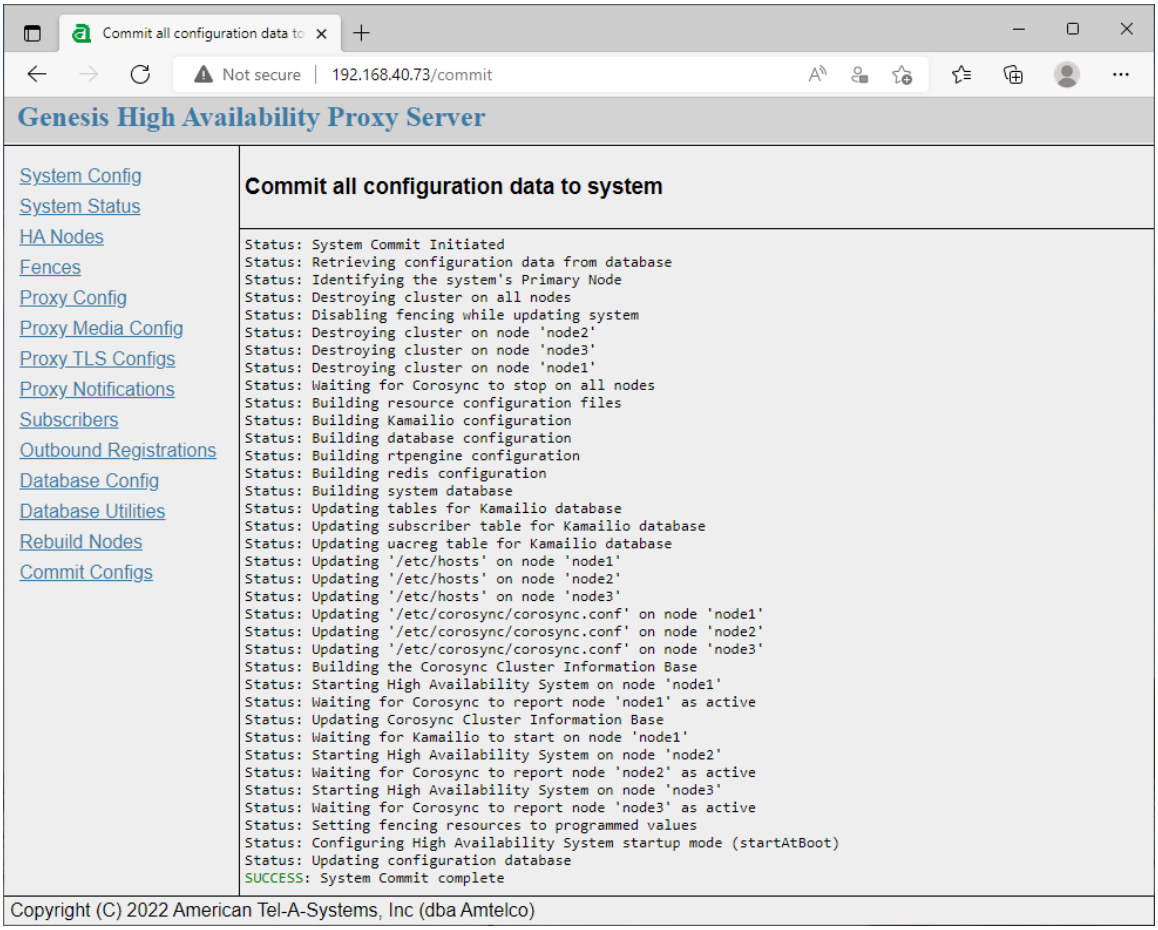

Figure 28: Commit Progress Page

Appendix A TLS and SRTP Notes

The proxy system supports the use of TLS and SRTP for VoIP communications. This appendix provides some guidance about the use of these protocols when the proxy server has TLS enabled.

SIP messaging configuration for TLS

Generally, TLS is used when the relevant URI begins with 'sips:' or has ';transport=tls' appended as a parameter. Some systems will attempt to force end-to-end TLS if 'sips:' is specified. Currently, the proxy server does not enforce this operation. Instead, if the destination is known to support only one protocol, and that protocol differs from the desired protocol, it will override the desired protocol and use the available one. Otherwise, the proxy server will use the desired protocol. There is no fallback to an alternate protocol in this case.

In some cases, use of "sips:" may be less likely to work for end-to-end communication. For this reason, it may be better to use "sip:" and set the transport parameter when TLS is used.

SRTP Options

The proxy server supports Secure RTP independent of whether TLS is used for SIP communications. Both DTLS and SDES are available. SDES is only appropriate if TLS is used for SIP messaging, since the keys are provided in the SDP of the SIP messaging.

If using DTLS, appropriate keys must be configured on the proxy system as well as at all affected endpoints.

DTLS does not require any specific configuration on the proxy system other than that required for TLS. It uses the same keys and certificates as TLS.

General Comments

- Certificate verification may not work for DTLS encrypted media, since IP addresses rather than domain names are often used in the SDP media section. It should be possible, however, to configure particular SIP endpoints for certificate verification. This is done by adding specific `Proxy TLS Configuration` server and client sections for the SIP <ip>:<port> of the endpoint. The two default sections can then be configured with `tlsVerify` set to '`Off`' (or '`Default`').
- TLS and SRTP both incur significant overhead, since all data must be encrypted and decrypted.

- It is very difficult to troubleshoot messaging and audio issues when encryption is used, since packet capturing programs do not have access to the encryption keys.

Genesis Considerations

Genesis should not be programmed to use both TLS (sips:) and non-TLS (sip:) for communicating with the proxy server.

It should be possible for Genesis to use non-TLS for SIP communications even if SRTP is required. If the far end is communicating with the proxy system via TLS, SDES keys will be exchanged securely over that link, so will still be secure. Use of DTLS is also secure regardless of the SIP communication protocol.

Media Encryption Notes

The proxy system includes a number of options for configuring media encryption. These are intended to allow for flexibility in setting the encryption mode for each endpoint in a SIP call. In particular, since Genesis will typically be communicating with the proxy server over a local network, but then to a SIP provider on the public internet, the proxy system allows the local connection to be unencrypted while ensuring that the public connection is encrypted. This eliminates the overhead that Genesis would incur if it is using media encryption.

The proxy server only modifes media encryption if TLS is enabled (`enableTLS` is '`Yes`') and NAT Traversal is enabled (`enableNAT` is '`Yes`').

The following paragraphs describe how the proxy system sets the encryption mode for outbound calls. Figure 29 is a flowchart of this algorithm.

1. If an INVITE is received that contains the header specified by `amtelcoMediaEncryption`, the proxy server will force the encryption on the outbound leg of the call to the specified mode. This header can enable RTCP-based feedback if '`_f`' is appended to the string. The following values are acceptable: "`sdes`", "`sdes_f`", "`dtls`", "`dtls_f`", "`none`", "`none_f`".
2. Otherwise, if the destination domain is in `rtpList`, set the outbound media to unencrypted. Set RTCP-based feedback according to `enableAvpf`.
3. Otherwise, if the destination domain is in `dtlsList`, use DTLS encryption. Set RTCP-based feedback according to `enableAvpf`.
4. Otherwise, if the destination domain is in `sdesList`, use SDES encryption. Set RTCP-based feedback according to `enableAvpf`.
5. Otherwise, if `changeMediaEncryption` is set to '`No`', use the same encryption on the outbound leg as on the inbound leg.
6. Otherwise, if the incoming SDP is configured to use encryption and the outgoing INVITE *is not* using TLS, set the outgoing RTP stream to unencrypted and set RTCP-based feedback according to `enableAvpf`.

7. Otherwise, if the incoming SDP is not configured to use encryption and the outgoing INVITE *is* using TLS, set the outgoing RTP stream to use the encryption specified by `defaultMediaEncryption` and set RTCP-based feedback according to `enableAvpf`.
8. Otherwise, use the same encryption on the outbound leg as on the inbound leg.

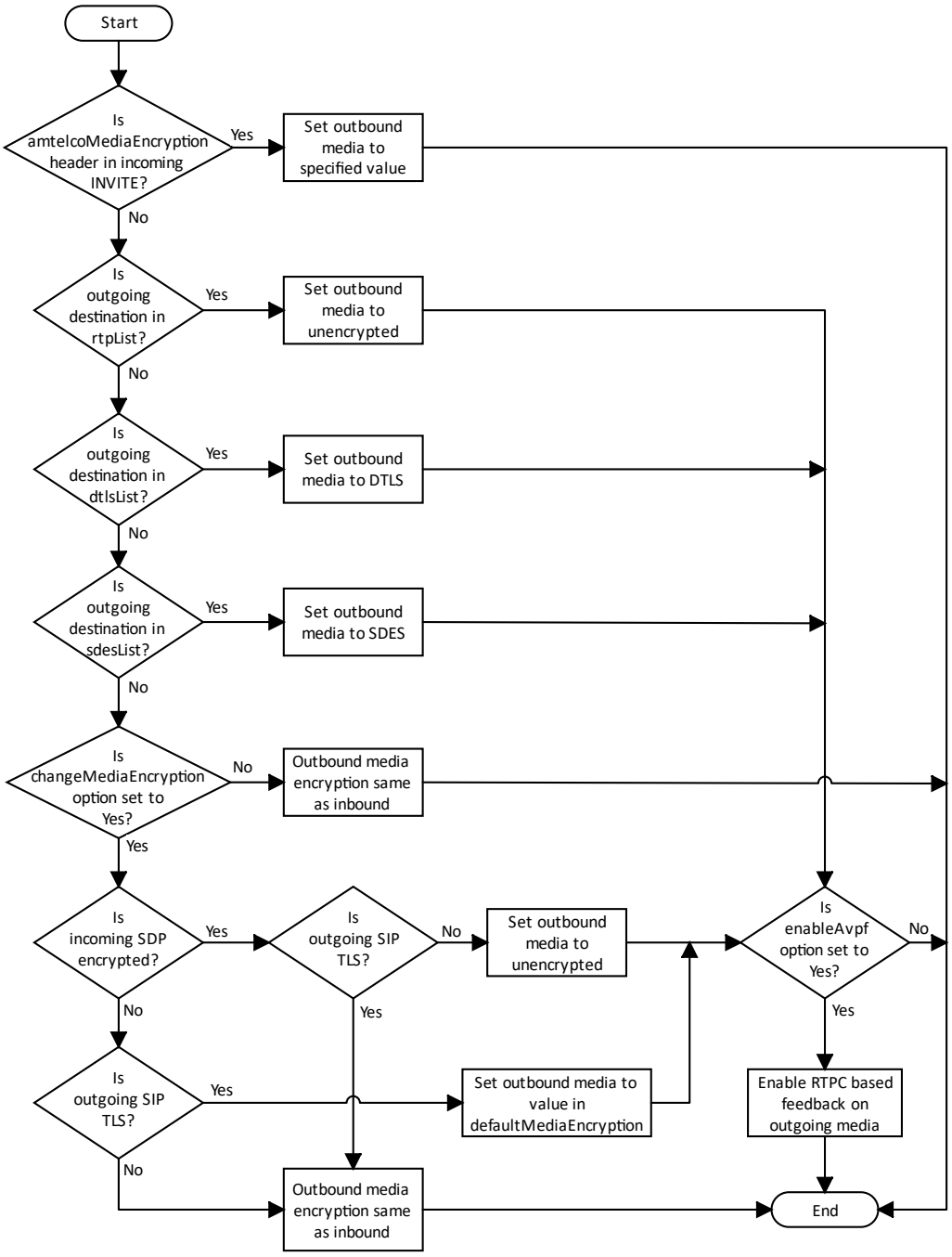

Figure 29: SDP Encryption Flowchart