SYBASE®

The Relay Server User Guide

August 2009

Version 11.0.1

Copyright and trademarks

Contents

About this book

Subject

This book describes how to set up and use the Relay Server.

Audience

This book is for all users of the Relay Server.

The Relay Server

Contents

Introduction to the Relay Server

The Relay Server enables secure, load-balanced communication between mobile devices and MobiLink, Afaria and iAnywhere Mobile Office servers through a web server. The Relay Server provides the following:

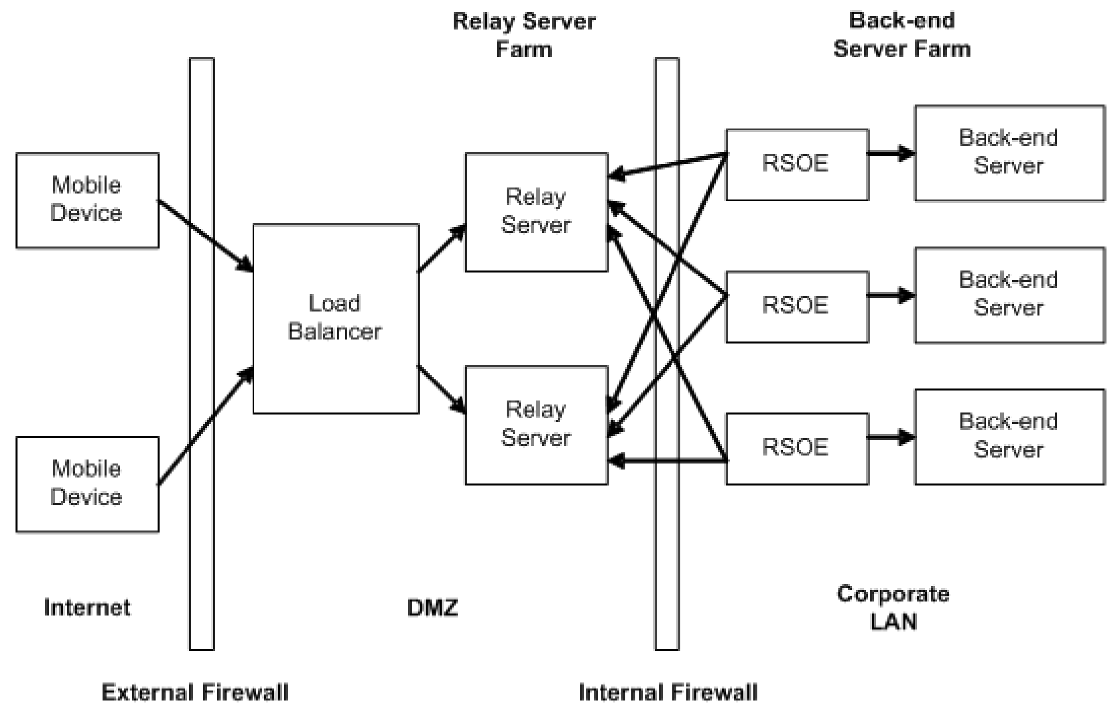
- A common communication architecture for mobile devices communicating with MobiLink, Afaria and iAnywhere Mobile Office servers.
- A mechanism to enable a load-balanced and fault-tolerant environment for MobiLink, Afaria and iAnywhere Mobile Office servers.
- A way to help communication between mobile devices and MobiLink, Afaria and iAnywhere Mobile Office servers in a way that integrates easily with existing corporate firewall configurations and policies.

Relay Server architecture

A Relay Server deployment consists of the following:

- Mobile devices running client applications and services that need to communicate with back-end servers running in a corporate LAN.
- Optional load balancer to direct requests from the mobile devices to a group of Relay Servers.
- One or more Relay Servers running in the corporate DMZ.
- Back-end servers running in a corporate LAN that are responsible for servicing client requests.
- One Relay Server Outbound Enabler (RSOE) per back-end server. The Outbound Enabler manages all communication between a back-end server and the Relay Server farm.

The following diagram shows the Relay Server architecture.



The Relay Server consists of a set of web extensions, a background process for maintaining state information, and a web server.

Because the Relay Server is a web extension running in a web server, all communication is performed using HTTP or HTTPS. Using HTTP easily integrates with existing corporate firewall configurations and policies. The Relay Server requires that the connection from the corporate LAN to the Relay Server be initiated from inside the corporate LAN. This provides a more secure deployment environment because it does not require inbound connections from the DMZ into the corporate LAN.

The Relay Server contains two web extensions: a client extension and a server extension. The client extension handles client requests made from applications running on mobile devices. The server extension handles requests made by the Outbound Enabler on behalf of a back-end server.

The Relay Server farm

A Relay Server farm is any number of Relay Servers with a front-end load balancer. It is possible to set up a Relay Server farm with a single Relay Server, in which case a load balancer is not required. In this case, mobile devices can connect directly to the Relay Server.

Back-end server farm

A back-end server farm is a group of homogeneous back-end servers. A client making a request through the Relay Server farm must specify the back-end server farm it is targeting.

Load balancer

The load balancer directs requests from the mobile devices to a Relay Server running in the Relay Server farm. The load balancer is not required if there is only one Relay Server.

Relay Server Outbound Enabler

The Relay Server Outbound Enabler runs on the same computer as the back-end server. Its primary function is to initiate an outbound connection to all Relay Servers in the Relay Server farm on behalf of the back-end server. There is one Outbound Enabler per back-end server. See "Outbound Enabler" on page 9.

Relay Server configuration file

A Relay Server configuration file is used to define both a Relay Server farm and the back-end server farms connecting to the Relay Server farm. The Relay Server configuration file is divided into sections:

- "Relay Server section" on page 5
- "Backend farm section" on page 6
- "Backend server section" on page 7
- "Options section" on page 7

Each section starts with a section tag. A section tag is formed by enclosing a keyword that identifies the section name in square brackets. For example, [relay_server] denotes the start of the Relay Server section.

The section tag is followed by several lines defining various properties related to the section being defined. A property is defined by specifying the property name on the left-hand side of an equal sign and its value on the right-hand side of the equal sign. For example, `property name = value`. All section and property names are case insensitive. Comments are marked with pound sign (#) character at the beginning of a line.

The configuration file should contain only 7-bit ASCII characters. The sections can be specified in any order.

Relay Server section

The Relay Server section is used to define a single Relay Server, so there must be a Relay Server section for each Relay Server in the farm. This section is identified by the relay_server keyword.

Relay Server section properties

The following properties can be specified in a Relay Server section:

- **enable** Specifies whether this Relay Server is to be included in the Relay Server farm. Possible values are:
  - **Yes** Indicates that this Relay Server is to be included in the Relay Server farm.
  - **No** Indicates that this Relay Server should not be included in the Relay Server farm.

  The default is Yes. This property is optional.
- **host** The hostname or IP address that should be used by the Outbound Enabler to make a direct connection to the Relay Server.
- **http_port** The HTTP port that should be used by the Outbound Enabler to make a direct connection to the Relay Server. A value of **0** or **off** disables HTTP connections. By default, this property is enabled and set to 80.
  - **0 or off** Disable HTTP access from Outbound Enabler.
  - **1 to 65535** Enable HTTP at the specified port.

- **https_port** The HTTPS port that should be used by the Outbound Enabler to make a direct connection to the Relay Server. A value of **0** or **off** disables HTTPS connections. By default, this property is enabled and set to 443.
  - **0 or off** Disable HTTPS access from Outbound Enabler.
  - **1 to 65535** Enable HTTPS at the specified port.
- **description** Enter a custom description to a maximum of 2048 characters. This property is optional.

Backend farm section

The backend farm section specifies the properties of a back-end server farm. A back-end server farm is a group of homogenous back-end servers. A client making a request through the Relay Server farm must specify the back-end server farm it is targeting. There is one backend farm section for each back-end server farm.

This section is identified by the backend_farm keyword.

Backend farm section properties

The following properties can be specified in a backend farm section:

- **enable** Specifies whether to allow connections from this back-end server farm. Possible values are:
  - **Yes** Allow connections from this back-end server farm.
  - **No** Disallow connections from this back-end server farm.

  The default is Yes. This property is optional.
- **id** The name assigned to the back-end server farm, to a maximum of 2048 characters.
- **client_security** Specifies the level of security the back-end server farm requires of its clients. The possible values are:
  - **on** Indicates that clients must connect using HTTPS.
  - **off** Indicates that clients must connect using HTTP.

  This property is optional. If no value is specified, clients can connect using either HTTP or HTTPS.
- **backend_security** Specifies the level of security required of an Outbound Enabler in the back-end server farm to connect to the Relay Server farm. The possible values are:
  - **on** Indicates that all connections from the back-end farm must by made using HTTPS.
  - **off** Indicates that all connections from the back-end farm must be made using HTTP.

  This property is optional. If no value is specified, either HTTP or HTTPS can be used to connect.
- **description** Enter a custom description to a maximum of 2048 characters. This property is optional.

Backend server section

The backend server section defines a back-end server connection. It specifies the information that is used by the Outbound Enabler when it connects to the Relay Server farm on behalf of a back-end server. There is a backend server section for each Outbound Enabler connecting to the Relay Server farm. The backend server section also assigns a back-end server to a back-end server farm.

This section is identified by the backend_server keyword.

Backend server section properties

The following properties can be specified in a backend server section:

- **enable** Specifies whether to allow connections from this back-end server. Possible values are:
  - **Yes** Allows connections from this back-end server.
  - **No** Disallows connections from this back-end server.

  The default is Yes. This property is optional.
- **id** The name assigned to the back-end server connection, to a maximum of 2048 characters.
- **farm** The name of the back-end server farm that this back-end server belongs to.
- **MAC** The MAC address of the network adapter used by the Outbound Enabler to communicate with the Relay Server. The address is specified using the IEEE 802 MAC-48 format. To get the MAC address in the correct format, look in the Relay Server Outbound Enabler console or log. This property is optional. If it is not specified, MAC address checking does not occur.
- **token** A security token that is used by the Relay Server to authenticate the back-end server connection, to a maximum of 2048 characters. This property is optional.
- **description** Enter a custom description to a maximum of 2048 characters. This property is optional.

Options section

The options section is used to specify properties that apply to each Relay Server in the farm. Only one options section is allowed.

This section is identified by the options keyword.

Options section properties

The following properties can be specified in an options section:

- **start** The method used to start the State Manager. The possible values are:
  - **auto** The State Manager is started automatically using the State Manager command line defaults.
  - **no** The State Manager is started externally as a Windows service.
  - **full path** Specify the full path to the State Manager executable (*rshost*).

  The default is auto. This property is optional.

- **shared_mem** Specifies the maximum amount of shared memory that the Relay Server uses for state tracking. The default is 10 megabytes. This property is optional.
- **verbosity** You can set verbosity to the following levels:
  - **0** Log errors only. Use this logging level for deployment. This is the default.
  - **1** Request level logging. All HTTP requests are written to the log file.

  Errors are displayed regardless of the log level specified, and warnings are displayed only if the log level is greater than 0.

Relay Server configuration file format

This is the basic format of a Relay Server configuration file:

```
#
# Options
#
[options]
# List of Relay Server properties that apply to all Relay Servers
option = value

#
# Define a Relay Server section, one for each
# Relay Server in the Relay Server farm
#
[relay_server]
# List of properties for the Relay Server
property = value

#
# Define a backend server farm section, one for each backend
# server farm
#
[backend_farm]
# List of properties for a backend server farm
property = value

#
# Define a backend server section, one for each
# Outbound Enabler connecting to the Relay Server farm
#
[backend_server]
# List of properties for the backend server connection
property = value
```

Outbound Enabler

The Outbound Enabler runs on the same computer as the back-end server. Its purpose is to:

- Open an outbound connection from the computer running in the corporate LAN to the Relay Server farm running in the DMZ.
- Forward client requests received from the Relay Server to the back-end server and forward back-end server responses back to the client via the Relay Server.

When the Outbound Enabler starts, it makes an HTTP request to retrieve the list of Relay Servers running in the farm. This is done using the server URL that maps to the web server extension component of the Relay Server. The server URL can map directly to a Relay Server or it can map to a load balancer. If the server URL maps to a load balancer, the load balancer forwards the request to one of the Relay Servers running in the farm. The Relay Server that receives the request from the Outbound Enabler returns the connection information for all Relay Servers in the farm. The Outbound Enabler then creates two outbound connections, called channels, to each Relay Server returned. One channel, called the up channel, is created using an HTTP request with an essentially infinite response. The response is a continuous stream of client requests from the Relay Server to the Outbound Enabler. The second channel, called the down channel, is created using an HTTP request with an essentially infinite content length. The request is formed by a continuous stream of server responses to client requests.

When the Outbound Enabler receives a client request on the up channel from one of the Relay Servers it has connected to, it forwards it to the back-end server that the Outbound Enabler is servicing. Once a response is received from the back-end server, it gets forwarded to the Relay Server from which it received the corresponding request using the down channel.

Outbound Enabler syntax

rsoe [*option*]+

rsoe @{ *filename* | *environment-variable* } ...

Parameters

Options The following options can be used with the Outbound Enabler. They are all optional.

rsoe options	Description
@data	Reads options from the specified environment variable or configuration file. If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file. See "File Hiding utility (dbfhide)" on page 10.
-f *farm*	The name of the farm that the back-end server belongs to.
-id *id*	The name assigned to the back-end server.
-cs *"connection-string"*	The host and port used to connect to the back-end server. The default is `"host=localhost;port=80"`.

rsoe options	Description
-cr *"connection-string"*	The Relay Server connection string. The format of the Relay Server connection string is a semicolon separated list of name-value pairs. The name-value pairs consist of the following: ● **host** IP address or hostname of the Relay Server. The default is localhost. ● **port** The port the Relay Server is listening on. This is required. ● **url_suffix** URL path to the server extension of the Relay Server. The default for Windows is */ias_relay_server/server/rs_server.dll*. The default for Linux is */srv/iarelayserver*. ● **https** 0 - HTTP (default) 1 - HTTPS For **https=1**, the following options can also be specified: ● **tls_type** RSA ● **certificate_name** Common name field of the certificate. ● **certificate_company** Organization name field of the certificate. ● **certificate_unit** Organization unit field of the certificate. ● **trusted_certificates** File containing a list of trusted root certificates.
-t *token*	The security token to be passed to the Relay Server.
-v *level*	Set the verbosity level to use for logging. The *level* can be **0**, **1**, or **2**: ● **0** Log errors only. Use this logging level for deployment. ● **1** Session level logging. This is a higher level view of a synchronization session. ● **2** Request level logging. Provides a more detailed view of HTTP requests within a synchronization session.
-d *seconds*	The Relay Server connection retry interval. The default is 5 seconds.
-s	Stop the Outbound Enabler.

File Hiding utility (dbfhide)

The File Hiding utility (dbfhide) uses simple encryption to obfuscate the contents of configuration files and initialization files.

Syntax

dbfhide *original-configuration-file encrypted-configuration-file*

Option	Description
original-configuration-file	Specifies the name of the original file.
encrypted-configuration-file	Specifies a name for the new obfuscated file.

The Relay Server and Outbound Enabler detect that a configuration file has been obfuscated using dbfhide and process it accordingly.

This utility does not accept the @data parameter to read in options from a configuration file.

Deployment considerations

The following considerations should be noted when using the Outbound Enabler:

- **Outbound Enabler as a Windows service** The Outbound Enabler may also be set up and maintained as a Windows service using the Service Utility. See "Relay Server State Manager" on page 12.
- **Authentication** You cannot use simple or digest authentication. The *rsoe.exe* does not support simple or digest authentication with web servers, regardless of the web server type or operating system.

Relay Server State Manager

The Relay Server State Manager is a process that is responsible for maintaining Relay Server state information across client requests and Outbound Enabler sessions. The State Manager is also responsible for managing the log file used by the Relay Server. The State Manager can either be started automatically by the Relay Server or started as a Windows service (on Windows only).

The default log file name is *ias_relay_server_host.log*. On Windows, this file is located in the directory specified by the TEMP environment variable. On Linux, the file is located in the directory specified by the TMP, TEMP, or TMPDIR environment variables. If none of those variables are set, a log file is created on the root.

Note
In all cases, the Apache user process must have write permissions to the *tmp* directory location you choose.

On a graceful shutdown, the State Manager renames the log file to a file of the form *<yymmdd><nn>.log* where *<yymmdd>* represents the date on which the log file was renamed and *<nn>* is the sequential version number of the log file for that day.

Starting the State Manager as a Windows service is the recommended method. Note that starting the State Manager manually on a command line is not supported.

It is possible to specify the options that are used by the Relay Server to start the State Manager. To change the options, set the **start** property in the options section of the Relay Server configuration file. For example:

```
[options]
start = "rshost -o c:\temp\myrshost.log"
```

Note that you must specify the name of the Relay Server State Manager executable (`rshost`) before the options.

Starting the Relay Server State Manager as a Windows service

For Windows only, the State Manager can be started as a Windows service by using the Service utility *dbsvc.exe*. The start property in the options section of the Relay Server configuration file should be set to **no**. See "Options section" on page 7.

The Service utility (dbsvc) is used to create, modify and delete services. For a full listing of usage information, run *dbsvc.exe* without any options.

To set up an auto started State Manager service named rs:

```
dbsvc -as -s auto -w rs "C:\inetpub\wwwroot\ias_relay_server\server
\rshost.exe" -q -qc -f c:\inetpub\wwwroot\ias_relay_server\server
\rstest.config -o c:\temp\rs.log
```

To start the service:

```
dbsvc.exe -u rs
```

To stop the service:

```
dbsvc.exe -x rs
```

To uninstall the service:

```
dbsvc.exe -d rs
```

Starting the Relay Server State Manager automatically

The State Manager process is started automatically when the first Outbound Enabler connects to the Relay Server. This is the default behavior when the start property in the options section of the Relay Server configuration file is not specified or is explicitly specified as auto. The default log file location is *%temp%\ias_relay_server_host.log*. See "Options section" on page 7.

Starting the Relay Server State Manager automatically with customized options

When auto start is desired but you want to override some default behavior such as verbosity level or log file location, you can use the start property in the options section of the Relay Server configuration file to explicitly specify your State Manager command line. The -f option cannot be used in this case and the configuration file must be named *rs.config* and be placed in the same directory as the server extension. See "Relay Server State Manager command line syntax" on page 13.

Note
Do not specify a log file location under the wwwroot directory. IIS does not allow a worker process to create a file under the published tree.

Relay Server State Manager command line syntax

rshost [*option*]+

Parameters

Options The following options can be used to configure the State Manager. They are all optional.

rshost options	Description
-f *filename*	File name of the Relay Server configuration file.
-o *filename*	File name to use for logging.
-oq	Prevent popup window on startup error.
-q	Run in minimized window.

rshost options	Description
-qc	Close window on completion.
-u	Update configuration of a running Relay Server.
-ua	Archive the log file to <yymmdd><nn>.log and truncate.

Deploying the Relay Server

The following is an overview of how to deploy the Relay Server to IIS on Windows:

1. Deploy the Relay Server components. See "Deploying the Relay Server components to IIS on Windows" on page 15.
2. Deploy the web server extensions and State Manager. See "Deploying the web server extensions and State Manager" on page 16.
   a. Create an application pool. See "Creating an application pool" on page 16.
   b. Enable the Relay Server web extensions and deploy the Relay Server configuration file. See "Deploying the web server extensions and State Manager" on page 16.
3. Deploy Relay Server configuration updates, as necessary. See "Updating a Relay Server configuration for IIS on Windows" on page 20.

The following is an overview of how to deploy the Relay Server to Apache on Linux:

1. Deploy the Relay Server components. See "Deploying the Relay Server components to Apache on Linux" on page 18.
2. Deploy the web extension files and State Manager. See "Deploying the web extension files and State Manager" on page 18.
3. Deploy Relay Server configuration updates, as necessary. See "Updating a Relay Server configuration for Apache on Linux" on page 21.

Deploying the Relay Server components to IIS on Windows

The Relay Server for Windows consists of the following executables:

- *rs_client.dll*
- *rs_server.dll*
- *rshost.exe*
- *dblgen11.dll*
- *dbsvc.exe*
- *dbfhide.exe*
- *dbicu11.dll*
- *dbicudt11.dll*
- *dbsupport.exe*
- *dbghelp.dll*

See also

- "Relay Server State Manager" on page 12
- "File Hiding utility (dbfhide)" on page 10

Deploying the web server extensions and State Manager

To deploy the Relay Server files

1. Create the following directories under the web site home directory that you use for the Relay Server:
   - *ias_relay_server*
   - *ias_relay_server\client*
   - *ias_relay_server\server*
2. Copy *rs_client.dll* to the *ias_relay_server\client* directory.
3. Create the Relay Server configuration file *rs.config*. See "Relay Server configuration file" on page 5.
4. Copy *rs_server.dll*, *rshost.exe* and *rs.config* to the *ias_relay_server\server* directory.
5. Ensure that the connection timeout property of the web site home directory is set to 60 seconds or more.

Creating an application pool

A dedicated application pool must be created for the *rs_server.dll* and *rs_client.dll* web server extensions. All worker recycling options need to be turned off as the Relay Server utilizes long running worker processes.

To create the application pool

1. Start IIS Manager Console.
2. Right-click **Application Pools** and create a new application pool, for example RS_POOL.
3. Edit the properties for the application pool you created:
   a. Select the **Recycling** tab and turn off all the recycling options.
   b. Select the **Performance** tab and do the following:
      i. Turn off **Shutdown Worker Processes After Being Idle**.
      ii. Set the number of worker processes to the total number of processing cores. You can further adjust this number depending on your usage and performance preferences. See the IIS performance notes about Web garden size for more information.

Enabling the Relay Server web extensions

The steps below describe the process to enable the Relay Server web extensions.

To edit the properties of ias_relay_server and enable the Relay Server web extensions

1. Select the **Directory** tab and do the following:
   a. Set execute permissions to **Scripts And Executables**.

   b. Click **Create** under **Application Settings**. Select the application pool you created in ???? as the associated application pool.

2. Select the **Directory Security** tab and do the following:

   a. Click **Edit** in **Authentication and Access Control**.

   b. Enable anonymous access and fill in the user name and password for an account belonging to the Administrators group.

      Alternatively, you may leave the setting as the built-in user **IUSR_%computername%** and execute the following command to grant permission to access the IIS metabase.

      ```
      C:\Windows\Microsoft.Net\Framework\<Version>\aspnet_regiis.exe -ga
      IUSR_%computername%
      ```

3. Under **Web Server Extensions** in the IIS manager, allow both *rs_server.dll* and *rs_client.dll* to be run as ISAPI.

4. Deploy the Relay Server configuration file by creating a Relay Server configuration file and copying it to the *ias_relay_server\server* directory.

See also

- "Relay Server configuration file" on page 5

Performance tips

Keep the following in mind when deploying the Relay Server to IIS on Windows:

- The Relay Server web extension does not rely on ASP.NET. Removing the ASP.NET ISAPI filter yields better performance. The filter gets turned on by default in a standard IIS install. To turn off the filter, do the following:

  1. Start IIS Manager Console.
  2. Edit the properties of **Default Web Site**.
  3. Under the **ISAPI Filters** tab, remove the ASP.NET filter.

- For better performance, you can turn off the IIS access log. To turn off the access log, do the following:

  1. Start IIS Manager Console.
  2. Edit the properties of the *ias_relay_server* directory under **Default Web Site**.
  3. Under the **Directory** tab, clear the **Log Visits** selection.

- In a production environment, Relay Server verbosity can be set to 0 via the Relay Server configuration file. This yields better performance under high loads.

- The Relay Server does not impose restrictions on the Web garden size. One worker process may serve requests from all Outbound Enablers as well as from all the clients. However, the number of threads that can be created in the process is limited by the process heap space left available for thread creation. The thread created by IIS has a 256k stack size. If your machine has adequate resources, experiment with a higher number of processes if you suspect you are hitting a concurrency limit when the server is loaded with thousands of concurrent requests.

Deploying the Relay Server components to Apache on Linux

The Relay Server for Linux consists of the following executables:

- *mod_rs_ap_client.so*
- *mod_rs_ap_server.so*
- *rshost*
- *dblgen11.res*
- *libdbtasks11_r.so*
- *libdbicudt11.so*
- *libdbicu11_r.so*
- *libdblib11_r.so*
- *dbsupport*
- *dbfhide*

See also

- "Relay Server State Manager" on page 12
- "File Hiding utility (dbfhide)" on page 10

Deploying the web extension files and State Manager

To deploy the Relay Server files

1. Copy the above files into your Apache install *modules* directory.
2. Create the Relay Server configuration file *rs.config*. See "Relay Server configuration file" on page 5.
3. Copy *rs.config* into the *modules* directory. The server module expects the *rshost* executable to be in the same directory where you copied the *rs.config* file.
4. Set the PATH and LD_LIBRARY_PATH environment variables to include the Apache *modules* directory.
5. Edit the Apache *conf/httpd.conf* file.
   a. Add the following lines to load the Relay Server client and server modules:

   ```
   LoadModule iarelayserver_client_module modules/mod_rs_ap_client.so

   LoadModule iarelayserver_server_module modules/mod_rs_ap_server.so
   ```

   The client and server modules are invoked using different URLs. The client module explicitly looks for the string *iarelayserver* in the URL path. That part of the URL need not change.

   b. Add the following line to create a <*location*> section for the client module:

```
<LocationMatch /cli/iarelayserver/* >
    SetHandler iarelayserver-client-handler
</LocationMatch>
```

c. Add the following line to create a <*location*> section for the server module:

```
<Location /srv/iarelayserver/* >
    SetHandler iarelayserver-server-handler
    RSConfigFile "/<apache-install>/modules/rs.config"
</Location>
```

> You must specify an `RSConfigFile` directive which specifies the location of the Relay Server configuration file, *rs.config*. The *rs.config* file must reside in the same directory where the `rshost` executable is deployed.

d. If the TimeOut directive is set, ensure it is set to at least 60 seconds.

6. On Linux, if any of the following environment variables are set globally when Apache spawns a process, then there is nothing further needed for the configuration of Apache: $TMP, $TMPDIR or $TEMP.

   If any of the above environment variables are not set globally, or if you want the default Relay Server log file to go in a specific temporary directory (for example, when the State Manager is started automatically but without customizations), then edit the file */<apache-dir>/bin/envvars* to set and then export TMP.

   For example, to edit $TMP in the envvars file, do the following:

```
set TMP="/tmp"
export TMP
```

   This sets the environment variable in the shell that Apache creates before it spawns its processes.

> **Note**
> In all cases, the Apache user process must have write permissions to the *tmp* directory location you choose.

Updating a Relay Server farm configuration

A Relay Server farm configuration is defined by the contents of the Relay Server configuration file. Each Relay Server in a Relay Server farm shares the same Relay Server configuration file, so when you update a Relay Server farm configuration you must update the Relay Server configuration file at each Relay Server in the farm. Updates include any of the following:

- Adding a new Relay Server to the Relay Server farm.
- Creating a new backend server farm and allowing it access to the Relay Server farm.
- Adding a new backend server to an existing backend server farm.
- Changing the properties of a Relay Server, backend server farm, or a backend server.
- Changing options.

One way to update a Relay Server configuration is to shutdown all Relay Servers, replace the Relay Server configuration file with the updated version, and restart all the Relay Servers. However, shutting down and restarting the Relay Servers means that users of the Relay Server may incur a service interruption.

The preferred method of updating a Relay Server configuration is to use the Relay Server State Manager to update the configuration while a Relay Server farm is running without interrupting service.

Updating a Relay Server configuration is done by launching a new instance of the Relay Server State Manager using the following command line format:

```
rshost -u -f <filename>
```

The –u option instructs the Relay Server State Manager to perform an update operation. The –f option specifies the name of the configuration file containing the updated configuration. See “Relay Server State Manager” on page 12.

Below is an overview of the steps required to update a Relay Server farm configuration:

1. Make your changes to the master copy of the Relay Server configuration file.
2. On each computer running an instance of a Relay Server that belongs to the Relay Server farm being updated, do the following:
   a. Replace the old configuration file with the updated configuration file.
   b. Run the Relay Server State Manager with the updated configuration file.

Updating a Relay Server configuration for IIS on Windows

To update a Relay Server configuration for IIS on Windows

1. For each computer that belongs to the Relay Server farm you are updating, copy the updated configuration file to the *ias_relay_server\server* directory under the Relay Server web site home directory. The configuration file must be called *rs.config* if auto start is used.

2. From the *ias_relay_server\server* directory, run the following command line to apply the configuration update:

   ```
   rshost -u -f rs.config
   ```

3. Repeat the previous steps for each computer in the Relay Server farm that is being updated.

Updating a Relay Server configuration for Apache on Linux

To update a Relay Server configuration for Apache on Linux

1. Copy the updated configuration file to the */modules* directory under the Apache install directory. The configuration file must be called *rs.config* if auto start is used.
2. From the */<Apache-install>/modules* directory, run the following command line to apply the configuration update:

   ```
   rshost -u -f rs.config
   ```

3. Repeat the previous steps for each computer in the Relay Server farm that is being updated.

Index