Contents

About this book ........................................................................................................ v

Introduction to the Relay Server .............................................................................. 1
  Relay Server architecture ......................................................................................... 1
  Affinity .................................................................................................................... 6
  Relay Server status page ....................................................................................... 7

Relay Server deployment .......................................................................................... 9
  Deploying the Relay Server components to Microsoft IIS 6.0 on Windows Server 2003 ........................................................................................................ 9
  Deploying the Relay Server components to Microsoft IIS 7.0, 7.5, or 8.0 ........ 13
  Deploying the Relay Server components to Apache on Linux .......................... 17

Relay Server State Manager ....................................................................................... 23
  Relay Server State Manager as a service .............................................................. 23
  Relay Server State Manager automatic start ...................................................... 24
  Relay Server State Manager (rshost) command line syntax ............................. 24

Relay Server configuration file .................................................................................. 27
  Relay Server section ............................................................................................... 27
  Backend farm section ............................................................................................ 28
  Backend server section ........................................................................................ 31
  Options section ..................................................................................................... 32
  Relay Server configuration file format .................................................................. 34

Outbound Enabler .................................................................................................... 35
  Outbound Enabler as a service ............................................................................... 43

Relay Server farm configuration updates ................................................................. 45
Relay Server

Updating a Relay Server configuration for Microsoft IIS on Windows .......... 46
Updating a Relay Server configuration for Apache on Linux ...................... 46

Relay Server plug-in for Sybase Central .................................................... 49
Working with Relay Server configuration files ......................................... 49
Managing Relay Servers and Relay Server farms ................................. 52
Managing backend servers and backend server farms .............................. 53

Relay Server logging and log administration .......................................... 57
Relay Server logging and SAP Passports .............................................. 57
Relay Server Record ............................................................................. 58
Outbound Enabler Record .................................................................. 63
Remote administration of the Relay Server log file ................................ 65

Sybase Hosted Relay Service ................................................................. 67
Subscribing to the Sybase Hosted Relay Service .................................. 67
Adding a server farm ........................................................................... 67

The Relay Server with MobiLink .......................................................... 69
Connecting a client to the Relay Server farm ...................................... 69
Setting up a relay server farm .............................................................. 70

Index .................................................................................................... 73
About this book

This book describes how to set up and use the Relay Server, which enables secure communication between mobile devices and Afaria, Mobile Office, MobiLink, SQL Anywhere, Unwired Server, and Sybase Unwired Platform servers through a web server.
Introduction to the Relay Server

The Relay Server enables secure, load-balanced communication between mobile devices and backend servers through a web server. Supported backend servers include Afaria, Mobile Office, MobiLink, SQL Anywhere, Unwired Server, and Sybase Unwired Platform. The Relay Server provides the following:

- A common communication architecture for mobile devices communicating with backend servers.
- A mechanism to enable a load-balanced and fault-tolerant environment for backend servers.
- A way to help communication between mobile devices and backend 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 backend 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.
- At least one backend server running in a corporate LAN that is responsible for servicing client requests.

The following backend servers are supported for use with the Relay Server:

- Afaria
- Mobile Office
- MobiLink
- SQL Anywhere
- Unwired Server
- Sybase Unwired Platform

Note

Relay Server is tested with specific backend servers and clients that communicate using well-defined HTTP requests and responses. Deployments that use custom HTTP traffic, including using SQL Anywhere as a Web server, must thoroughly test the traffic to ensure that it works with the Relay Server.

Refer to your license agreement or the SQL Anywhere Components by Platforms page for information about which backend servers are supported. See http://www.sybase.com/detail?id=1061806.

- There is usually only one, but there may be several, Relay Server Outbound Enablers (RSOE) per backend server. The Outbound Enabler manages all communication between a backend server and the Relay Server farm.

The following diagram shows the Relay Server architecture with a single Relay Server.
Note
Refer to your license agreement or the SQL Anywhere Components by Platforms page for information about which backend servers are supported. See http://www.sybase.com/detail?id=1061806.

The following diagram shows the Relay Server architecture for a more complex system with a Relay Server farm and a backend server farm.
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 backend server.

**Shared Memory and Security**

The Relay Server uses shared memory to transfer HTTP requests and responses between the client and server plug-ins. Secure deployments use HTTPS between the client and the Relay Server, and between the Outbound Enabler and the Relay Server. In this scenario, the Web Server decrypts the HTTPS into HTTP,
then the Relay Server re-encrypts the HTTP on its way to the Outbound Enabler. The brief interval during which the data is unencrypted in the Relay Server is sometimes called the Wireless Application Protocol gap or WAP Gap.

There are two ways to secure this data from rogue processes on the same computer. The primary approach is to use clients and backend servers supporting end-to-end encryption. Most MobiLink clients support end-to-end-encryption. The second approach, which is minimally recommended for all Relay Servers, is to harden the Web server and operating system (OS) for deployment to the DMZ using the standard techniques documented for each supported Web server and OS. This hardening should include steps to reduce the number of OS accounts on the Web server computer. Ideally, the hardening also restricts the computer/VM to only run the Relay Server and the Web server -- nothing else. The goal is to minimize the number of processes on the computer to a bare minimum, while hardening to prevent rogue agents from adding rogue programs.

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.

Backend server farm

A backend server farm is a group of homogeneous backend servers. A client making a request through the Relay Server farm must specify the backend 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 backend server. Its primary function is to initiate an outbound connection to all Relay Servers in the Relay Server farm on behalf of the backend server. There is usually only one, but there may be several, Relay Server Outbound Enablers (RSOE)s per backend server.

See also

- See “Outbound Enabler” on page 35.

Relay Server security

The Relay Server has built-in security features, but also relies on the security features provided by the web server. In combination with the web server, the Relay Server provides the following features for secure communications:

- Server-side certificates
- Client-side certificates
• backend server and farm configuration

• RSOE MAC address filtering and token authentication

• Client encryption technologies (Protocol-level encryption)

Server-side certificates
Using a server-side certificate, a client communicating with the Relay Server can verify the web server that is running the Relay Server is a trusted server. The client verifies the web server's public certificate with the root certificate stored on the client. If the certificates are verified, a key exchange occurs to establish the encrypted connection.

Client-side certificates
Using a client-side certificate, the web server can verify that a client communicating with the Relay Server is a trusted client. The web server verifies the client's public certificate with its root certificate stored in the certificate manager on the web server computer. If the certificates are verified, a key exchange occurs to establish the encrypted connection.

backend server and farm configuration
The rs.config file is used by the Relay Server to define the peer list of Relay Servers, if running in a farm environment, including the backend farm and backend server configurations. Each Relay Server in the environments needs to maintain a copy of the rs.config file.

The configuration of the backend farm and backend servers ensures the Relay Server only communicates with computers with which it has been configured. Any attempted communication with computers for which the Relay Server has not been configured are refused.

The backend farm can be configured to specify the level of communication security when accepting requests from the clients and the RSOE. There is a client_security and backend_security option that allows the backend farm to specify the type of communication that can be established. This option is specified as follows:

client_security=on|off On indicates the client must connect using HTTPS. Off indicates the client must connect using HTTP. This setting is optional. If no value is specified, the client can connect using HTTP or HTTPS.

backend_security=on|off On indicates the RSOE must connect using HTTPS. Off indicates the RSOE must connect using HTTP. This setting is optional. If no value is specified, the RSOE can connect using HTTP or HTTPS.

RSOE MAC address filtering and token authentication
The RSOE establishes the connection between the backend server and the Relay Server using three phases: 1) startup phase, 2) ready phase 3) working phase.

In the backend server section of the rs.config file, each server that exists in the backend farm is configured with an ID and associated farm name. The ID corresponds to the server name. The Relay Server has the ability to verify the MAC address of the computer running the RSOE to ensure the server communicating from within the internal firewall is trusted and allowed to establish a connection with the Relay Server.
The MAC property is the MAC address of the network adapter used by the RSOE. The address is specified in IEEE 802 MAC-48 format.

The backend server section also allows for the configuration of a security token that is used by the Relay Server to authenticate the backend server connection. The token must be provided upon startup of the RSOE when establishing the connection with the Relay Server.

**MobiLink security**

The MobiLink client uses HTTP or HTTPS to communicate with the Relay Server. With HTTPS communication, data is temporarily decrypted and re-encrypted as it is exchanged between the client and backend server. This is known as the WAP Gap. To ensure completely secure communication through the WAP Gap, it is recommended that you use the MobiLink end-to-end encryption feature to further protect data as it passes through the Relay Server. The MobiLink end-to-end encryption feature provides protocol-level encryption between the MobiLink, and UltraLite clients and the MobiLink server. MobiLink and UltraLite use RSA encryption. TLS security can be used in combination with end-to-end encryption.

**HTTP listening port of the outbound enabler**

For security purposes, when using the standalone outbound enabler, the HTTP listening port of the backend server should be explicitly bound only to the loopback IP address (127.0.0.1).

**See also**

- “Relay Server configuration file” on page 27
- Microsoft IIS http://www.sybase.com/detail?id=1059277
- Apache http://www.sybase.com/detail?id=1065869
- “End-to-end encryption” [SQL Anywhere Server - Database Administration]

**Affinity**

Some clients and back-end servers require that certain sequences of multiple HTTP requests go to the same back-end server or even to the same TCP socket connected to that back-end server. In Relay Server terminology, "affinity" is the association between a client and a back-end server across multiple HTTP requests. If a client does NOT require multiple requests to go to the same back-end server, then affinity is not required and you don't need to know anything more about it.

The Relay Server adds affinity information to HTTP responses, and respects affinity information sent up in HTTP requests. The affinity information is sent via HTTP cookies and/or headers. Clients wanting to issue multiple requests to the same back-end server via the Relay Server must send back the affinity information received in each HTTP response by injecting it into the next HTTP request. Back-end servers do not need to do anything to participate in Relay Server affinity. When non-persistent HTTP is used for a sequence of multiple requests for the same back-end, each request creates a new TCP socket connection. When there are multiple back-end servers in a farm, the client must maintain affinity information between the requests. The affinity information tells the Relay Server that the requests are related and are targeting the same back-end server.

When persistent HTTP is used, each request must be on the same TCP socket. Persistent HTTP is a more popular approach because of the performance advantage it provides by reducing the overhead due to
socket (and possibly TLS) connections. When persistent HTTP is used, the client must still maintain
affinity information between requests because the Relay Server still requires it to maintain the persistence
between the Outbound Enabler and the back-end server.

Client-side problems with managing affinity can cause failed requests and other strange behavior. If you
suspect an affinity issue with your client application, contact your client application vendor. If the Relay
Server administrator is familiar with the client application and has a thorough understanding of the chosen
mechanism for maintaining affinity across all involved clients of the back-end farm, the administrator
may selectively turn off unused affinity injection using the active_cookie or active_header properties in
the back-end farm section of the Relay Server configuration file. Explicitly turning off the
renew_overlapped_cookie property is NOT recommended. Doing so may lead to connectivity issues
when the client is using standard HTTP cookie reflection to maintain affinity information without
isolation between concurrent sessions.

See also
“Backend farm section properties” on page 28

Relay Server status page

The Relay Server status page provides the following information:

- Relay Server version and build number
- Host name
- Service start time in UTC
- Status capture time in UTC
- Status refresh interval or indication that a manual refresh is expected
- Overall availability
- A list of unavailable backend farms
- A list of partially available backend farms
- A list of available backend farms

Empty status page

An empty status page may indicate that there is a configuration issue preventing the extension from
generating the page. Make sure that the rshost process has been started and that the web server worker has
permission to access the shared memory created by the rshost process. To troubleshoot configuration
issues, start rshost manually and check the log file.

The frequency of the backend server status page auto refresh is set using the ias-rs-status-refresh-sec
query parameter, which forms part of the status url parameter. The status_url parameter is specified in the
format status_url=/your-status-url ias-rs-status-refresh-sec=n. If n is zero,
then auto refresh is turned off. So, to have the status page refresh every minute, you might use the
following: http://<host>/rs/client/rs_client.dll?ias-rs-status-refresh-
sec=60&ias-rs-farm=SimpleTestApp-farm.

The URL for a typical status page is http://MyHost:80/ias Relay_server/server/rs_server.dll for Microsoft
### Additional status pages

The following status pages are also available:

<table>
<thead>
<tr>
<th>Status page</th>
<th>User or location</th>
<th>URL format examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detailed status</td>
<td>Relay Server administrator</td>
<td>IIS: http://&lt;host&gt;/ias_relay_server/admin/rs_admin.dll</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Apache: http://&lt;host&gt;/srv/ias_relay_server/admin/rs_admin.dll</td>
</tr>
<tr>
<td>Overall status</td>
<td>Remote user</td>
<td>IIS: http://&lt;host&gt;/ias_relay_server/client/rs_client.dll</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Apache: http://&lt;host&gt;/srv/ias_relay_server/client/rs_client.dll</td>
</tr>
<tr>
<td>Backend farm status</td>
<td>Remote user</td>
<td>IIS: http://&lt;host&gt;/ias_relay_server/client/rs_client.dll?ias-rs-farm=&lt;App-farm&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Apache: http://&lt;host&gt;/srv/ias_relay_server/client/rs_client.dll?ias-rs-farm=&lt;App-farm&gt;</td>
</tr>
<tr>
<td>Backend server status</td>
<td>Remote user</td>
<td>IIS: http://&lt;host&gt;/ias_relay_server/client/rs_client.dll?ias-rs-farm=&lt;App-farm&gt;&amp;ias-rs-server=&lt;App-server&gt;</td>
</tr>
<tr>
<td>Overall status</td>
<td>Backend server administrator</td>
<td>IIS: http://&lt;host&gt;/ias_relay_server/server/rs_server.dll</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Apache: http://&lt;host&gt;/srv/ias_relay_server/server/rs_server.dll</td>
</tr>
<tr>
<td>Backend farm status</td>
<td>Backend server administrator</td>
<td>IIS: http://&lt;host&gt;/ias_relay_server/server/rs_server.dll?ias-rs-farm=&lt;App-farm&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Apache: http://&lt;host&gt;/srv/ias_relay_server/server/rs_server.dll?ias-rs-farm=&lt;App-farm&gt;</td>
</tr>
<tr>
<td>Backend server status</td>
<td>Backend server admini</td>
<td>IIS: http://&lt;host&gt;/ias_relay_server/server/rs_server.dll?ias-rs-farm=&lt;App-farm&gt;&amp;ias-rs-server=&lt;App-server&gt;</td>
</tr>
<tr>
<td></td>
<td>strator</td>
<td>Apache: http://&lt;host&gt;/srv/ias_relay_server/server/rs_server.dll?ias-rs-farm=&lt;App-farm&gt;&amp;ias-rs-server=&lt;App-server&gt;</td>
</tr>
</tbody>
</table>
Relay Server deployment

Deploying the Relay Server components to Microsoft IIS 6.0 on Windows Server 2003

Before running the Relay Server with IIS 6.0, you need to deploy Relay Server files to each computer in the Relay Server farm.

**Prerequisites**

The Relay Server components are installed as part of the SQL Anywhere 16 installation. The install process automatically deploys all the necessary files on the computer that is going to run the Relay Server.

By default, all files are installed to `%SQLANY16%` and are based on the bitness of the computer:

- `%SQLANY16%\Bin64` is used for DLLs and executables for administration.
- `%SQLANY16%\RelayServer\IIS\Bin64` is used for Relay Server-specific files under the appropriate folder (for example, `Admin`, `Client`, `Monitor`, or `Server`). The `Server` folder contains the `rshost.exe` and `rs.config` files.

**Context and remarks**

**Interactive quick setup feature**

An interactive quick setup feature, `rs-setup.bat`, is provided as an alternative to this procedure. `rs-setup.bat` is located in the `%SQLANY16%\RelayServer\IIS\quicksetup_iis6` directory and performs the following tasks:

1. Creates a demo application
2. Generates a quick reference guide
The Relay Server for Windows consists of the following executables:

- `rs_client.dll`
- `rs_server.dll`
- `rs_monitor.dll`
- `rshost.exe`
- `dblgen16.dll`
- `dbsvc.exe`
- `dbfhide.exe`
- `dbtool16.dll`
- `dblib16.dll`
- `dbicu16.dll`
- `dbicudt16.dll`
- `dbsupport.exe`
- `dbghelp.dll`

For information about which versions of IIS are supported, see [http://www.sybase.com/detail?id=1061806](http://www.sybase.com/detail?id=1061806).

Setup scripts for the Relay Server on IIS can be found in the `%SQLANY16%\RelayServer\IIS` directory.

**Task**

1. Create a virtual directory called `rs` under the Default Web Site in the Microsoft IIS Manager for use by the Relay Server. The physical location of the virtual directory is `%SQLANY16%\RelayServer\IIS\Bin64`.

2. Create the Relay Server configuration file `rs.config` using the following guidelines:

   - The file should have four sections:
     - Options section
     - Relay server section
     - Backend farm section
     - Backend server section

   - Each section starts with a section tag that encloses a keyword that identifies the section name in square brackets.

   - Add the appropriate properties to each section. A property is defined by specifying the property name on the left side of an equal sign and its value on the right side of the equal sign. For example, `propertyname = value`.

   - The configuration file should contain only 7-bit ASCII characters.

3. Create an application pool:
   a. Start Microsoft IIS Manager Console.
   b. Right-click **Application Pools** and create a new application pool, for example RS_POOL.
   c. Edit the properties for the application pool you created.
i. Click the **Recycling** tab and turn off all the recycling options.

ii. Click the **Performance** tab and:
   A. Turn off **Shutdown Worker Processes After Being Idle**.
   B. 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. For more information, see the Microsoft IIS performance notes about Web garden size.

4. Set the Connection timeout property of the Default Web Site to a minimum of 60 seconds. By default this value should be 120 seconds, which is sufficient.

5. Edit the properties of rs and enable the Relay Server web extensions using the IIS Manager Console:
   a. Click the **Directory** tab and do the following:
      i. Set execute permissions to **Scripts And Executables**.
      ii. Click **Create** under **Application Settings**. Select the application pool you created in step 3 as the associated application pool.
   b. Click the **Directory Security** tab and do the following:
      i. Click **Edit** in **Authentication and Access Control**.
      ii. 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 run the following command to grant permission to access the Microsoft IIS metabase.

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

   c. Under **Web Server Extensions** in the Microsoft IIS manager, add **rs_server.dll**, **rs_client.dll**, and **rs_monitor.dll** as a new Web service extension. The extension name should be ISAPI and the DLLs need to have the extension status set to Allowed.

6. Deploy the Relay Server configuration file by creating a Relay Server configuration file and copying it to the `%SQLANY16%\ RelayServer\ IIS\ BinXX\ server` directory.

7. Ensure optimum performance by reviewing the performance tips.

8. Define a service that automatically starts the Relay Server State Manager when the computer is started using a command line similar to the following:

   ```
   dbsvc -as -t rshost -w RelayServer "%SQLANY16%\ RelayServer\ IIS\ BinXX\Server\ rshost.exe" -q -qc -f "%SQLANY16%\ RelayServer\ IIS\ BinXX\Server\ rs.config" -o "c:\temp\ias_relay_server.log"
   ```

   **Note**
   It is recommended that you start the State Manager as a service. However, it can also be started automatically by the Relay Server.

9. Update the Relay Server configuration for Microsoft IIS 6.0 on Windows:
a. For each computer that belongs to the Relay Server farm you are updating, copy the updated configuration file to the `%SQLANY16%\RelayServer\IIS\BinXX\Server` directory under the Relay Server web site home directory.

b. From the `%SQLANY16%\RelayServer\IIS\BinXX\Server` directory, run the following command line to apply the configuration update:

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

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

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>After configuring the Relay Server with IIS, it is recommended that you restart the IIS server or the computer.</td>
</tr>
</tbody>
</table>

**Results**

The Relay Server configuration file is deployed to all computers in the Relay Server farm.

**See also**

- “Relay Server configuration file” on page 27
- “Performance tips” on page 12
- “Relay Server State Manager” on page 23
- “Relay Server State Manager as a service” on page 23
- “Relay Server State Manager (rshost) command line syntax” on page 24
- “File Hiding utility (dbfhide)” on page 42

**Performance tips**

Keep the following in mind when deploying the Relay Server to Microsoft 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 Microsoft IIS install. To turn off the filter, do the following:
  1. Start Microsoft 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 Microsoft IIS access log. To turn off the access log, do the following:
  1. Start Microsoft IIS Manager Console.
  2. Edit the properties of the **rs** 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 Microsoft IIS has a 256k stack size. If your computer 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 Microsoft IIS 7.0, 7.5, or 8.0

Before running the Relay Server with IIS 7.0, 7.5, or 8.0, you need to deploy Relay Server files to each computer in the Relay Server farm.

Prerequisites

The Microsoft IIS ISAPI Extensions feature is installed.

The Relay Server components are installed using the SQL Anywhere install. By default, all files are installed to %SQLANY16%:

- %SQLANY16%\Bin64 is used for DLLs and executables for administration.
- %SQLANY16%\RelayServer\IIS\Bin64 is used for Relay Server-specific files under the appropriate folder (for example, Admin, Client, Monitor, or Server). The Server folder contains the rshost.exe and rs.config files.

Context and remarks

Interactive quick setup feature

An interactive quick setup feature, rs-setup.bat, is provided as an alternative to this procedure. rs-setup.bat is located in the %SQLANY16%\RelayServer\IIS\quicksetup_iis7or8 directory and performs the following tasks:

1. Installs IIS7 or IIS8 and turns on the required IIS7 or IIS8 features
2. Configures IIS7 or IIS8 for the Relay Server
3. Creates a demo application
4. Generates a quick reference guide
The Relay Server for Windows consists of the following executables:

- rs_client.dll
- rs_server.dll
- rs_monitor.dll
- rshost.exe
- dblgen16.dll
- dbsvc.exe
- dbfhide.exe
- dbtool16.dll
- dblib16.dll
- dbicu16.dll
- dbicudt16.dll
- dbsupport.exe
- dbghelp.dll

For information about which versions of IIS are supported, see http://www.sybase.com/detail?id=1061806.

The IIS 7/8 setup scripts for Relay Server can be found in the %SQLANY16%\RelayServer\IIS\quicksetup_iis7or8 directory.

**Task**

1. Backup the IIS configuration file applicationHost.config located in the c:\Windows\System32\inetsrv\config folder.

2. To add an application pool for the Relay Server, edit the applicationHost.config file to add the following code to the <system.applicationHost> » <applicationPools> section.

```xml
<add name="RelayServer" queueLength="65535" managedRuntimeVersion=""
     managedPipelineMode="Integrated">
  <processModel identityType="LocalSystem" idleTimeout="00:00:00"
                  maxProcesses="20" pingingEnabled="false"
                  pingInterval="00:00:30" pingResponseTime="00:01:30" />
  <recycling disallowOverlappingRotation="true">
    <periodicRestart time="00:00:00">
      <schedule>
        <clear />
      </schedule>
    </periodicRestart>
  </recycling>
  <failure rapidFailProtection="false" />
  <cpu resetInterval="00:00:00" />
</add>
```

**Note**
The rest of the steps refer the %SQLANY16%\RelayServer\IIS\BinXX directory as %rs_dir% in the applicationHost.config file. However, environment variable expansion is not fully supported in every section in the IIS configuration file, so the %rs_dir% variable needs to be fully expanded when you add it into the applicationHost.config file.
3. To add the Relay Server application to the default site, edit the `applicationHost.config` file to add the following code to the `<system.applicationHost> » <sites> » <site name="Default Web Site">` section.

```xml
<application path="/rs" applicationPool="RelayServer">
  <virtualDirectory path="/" physicalPath="%rs_dir%"/>
</application>
```

4. To add the Relay Server ISAPI extensions, edit the `applicationHost.config` file to add the following code to the `<system.webServer> » <security> » <isapiCgiRestriction>` section.

```xml
<add path="%rs_dir%\Admin\rs_admin.dll" allowed="true" />
<add path="%rs_dir%\Client\rs_client.dll" allowed="true" />
<add path="%rs_dir%\Monitor\rs_monitor.dll" allowed="true" />
<add path="%rs_dir%\Server\rs_server.dll" allowed="true" />
```

5. To enable access to the Relay Server extensions, edit the `applicationHost.config` file to add the following code to the `<configuration>` section.

```xml
<location path="Default Web Site/rs">
  <system.webServer>
    <security>
      <authentication>
        <anonymousAuthentication userName="" />
      </authentication>
      <requestFiltering>
        <requestLimits maxAllowedContentLength="2147483647" />
      </requestFiltering>
    </security>
    <handlers accessPolicy="Execute, Script" />
  </system.webServer>
</location>
```

**Note**
The Relay Server is set up for anonymous access based on these instructions. Proper security needs to be configured for IIS and the Relay Server based on the business requirements.

6. To enforce HTTPS access to the Relay Server administration extension for security reasons, edit the `applicationHost.config` file to add the following code to the `<configuration>` section.

```xml
<location path="Default Web Site/rs/Admin">
  <system.webServer>
    <security>
      <access sslFlags="Ssl" />
    </security>
  </system.webServer>
</location>
```

7. Save these changes to the `applicationHost.config` file.

8. Set the Connection timeout property of the Default Web Site to a minimum of 60 seconds. By default this value should be 120 seconds, which is sufficient.

9. Create the Relay Server configuration file `rs.config` using the following guidelines:
The file should have four sections:

- Options section
- Relay server section
- Backend farm section
- Backend server section

Each section starts with a section tag that encloses a keyword that identifies the section name in square brackets.

Add the appropriate properties to each section. A property is defined by specifying the property name on the left side of an equal sign and its value on the right side of the equal sign. For example, `propertyname = value`.

The configuration file should contain only 7-bit ASCII characters.

10. Copy the `rs.config` file to the `%SQLANY16%\RelayServer\IIS\BinXX\Server` directory.

11. Ensure optimum performance by reviewing the performance tips.

12. Define a service that automatically starts the Relay Server State Manager when the computer is started using a command line similar to the following:

```
dbsvc -a <administrator> -p <password> -t rhost -w RelayServer "%rs_dir%\Server\rhost.exe" -q -qc -f "%rs_dir%\Server\rs.config" -o "c:\temp\lias_relay_server.log"
```

**Note**
It is recommended that you start the State Manager as a service. However, it can also be started automatically by the Relay Server.

13. Update the Relay Server configuration for Microsoft IIS on Windows:

a. For each computer that belongs to the Relay Server farm you are updating, copy the updated configuration file to the `%SQLANY16%\RelayServer\IIS\BinXX\Server` directory under the Relay Server web site home directory.

b. From the `%SQLANY16%\RelayServer\IIS\BinXX\Server` directory, run the following command line to apply the configuration update:

```
rhost -u -f rs.config
```

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

**Results**

The Relay Server configuration file is deployed to all computers in the Relay Server farm.
Deploying the Relay Server components to Apache on Linux

Before running the Relay Server with Apache, you need to deploy Relay Server files to each computer in the Relay Server farm.

Prerequisites

The Relay Server components are installed using the SQL Anywhere install. On Linux, the Relay Server files are installed to the /opt/sqlanywhere16 directory as part of the SQL Anywhere installation.

Context and remarks

Interactive quick setup feature

An interactive quick setup feature is provided as an alternative to this procedure. The quick setup feature:

1. Configures the web server for Relay Server
2. Creates a demo application
3. Generates a quick reference guide

The quick setup is comprised of two main steps:

1. Configure the Apache web server for Relay Server. This step can be accomplished by running ap-setup.sh script in the install-dir/relayserverserver/quicksetup_apache directory.

2. Create and start Relay Server test services This step can be accomplished by running rs-test-setup.sh script in the install-dir/relayserverserver/quicksetup_apache directory.
The Relay Server for Apache consists of the following executables:

- `mod_rs_ap_client.so`
- `mod_rs_ap_server.so`
- `rshost`
- `dblgen16.res`
- `libdbtasks16.so`
- `libdbtasks16_r.so`
- `libdbicudt16.so`
- `libdbicu16_r.so`
- `libdblib16_r.so`
- `dbsupport`
- `dbfhide`
- `libdblib16.so`
- `mod_rs_ap_monitor.so`
- `mod_rs_ap_admin.so`

**Task**

1. Create the Relay Server configuration file `rs.config`.
2. Copy `rs.config` into the `install-dir/relayserver/apache??/bin64` directory.
3. Edit the Relay Server configuration file `rs.config` using the following guidelines.

   - The file should have four sections:
     - Relay server section
     - Backend farm section
     - Backend server section
     - Options section

   - Each section starts with a section tag that encloses a keyword that identifies the section name in square brackets.

   - Add the appropriate properties to each section. A property is defined by specifying the property name on the left side of an equal sign and its value on the right side of the equal sign. For example, `property name = value`.

   - The configuration file should contain only 7-bit ASCII characters.

4. The LD_LIBRARY_PATH environment variable needs to include the Apache `install-dir/lib64` and `install-dir/relayserver/apache??/bin64` directories. Edit the `/<apache-dir>/bin/envvars` file to set and then export LD_LIBRARY_PATH.

5. Edit the Apache `conf/httpd.conf` file.

   a. Add the following lines to load the Relay Server client and server modules:

   ```
   LoadModuleiarelayserver_client_module install-dir/relayserver/
apache??/bin64/mod_rs_ap_client.so
   LoadModuleiarelayserver_server_module install-dir/relayserver/
apache??/bin64/mod_rs_ap_server.so
   ```
b. Add the following line to load the SQL Anywhere Monitor support module:

```
LoadModuleiarelayserver_monitor_module install-dir/relayserver/
apache??/bin64/mod_rs_ap_monitor.so
```

c. Add the following line to load the Remote Administration support module:

```
LoadModuleiarelayserver_admin_module install-dir/relayserver/
apache??/bin64/mod_rs_ap_admin.so
```

d. Add the following line to create a `<LocationMatch>` section for the client module:

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

e. Add the following line to create a `<LocationMatch>` section for the server module:

```
<LocationMatch/srv/iarelayserver/*>
    SetHandleriarelayserver-server-handler
    RSConfigFile"/install-dir/relayserver/apache??/bin64/rs.config"
</LocationMatch>
```

Note
---
You must specify an `RSConfigFile` directive which specifies the location of the Relay Server configuration file, `rs.config`.

f. Add the following line to create a `<LocationMatch>` section for the SQL Anywhere Monitor module:

```
<LocationMatch/mon/iarelayserver/*>
    SetHandleriarelayserver-monitor-handler
</LocationMatch>
```

g. Add the following line to create a `<LocationMatch>` section for the Remote Administration module:

```
<LocationMatch/admin/iarelayserver/*>
    SetHandleriarelayserver-admin-handler
</LocationMatch>
```

h. 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**
The Apache user process must have write permissions to the specified tmp directory.

7. To update the Relay Server configuration while it is started:
   a. Copy the updated configuration file to the `install-dir/relayserver/apache??/bin64` directory.
   b. From the `install-dir/relayserver/apache??/bin64` directory, run the following command line to apply the configuration update:

   ```
rhost -u -f rs.config
   ```

   c. If the Relay Server is set up as a farm with more than one server, repeat the previous steps for each computer in the Relay Server farm.

**Results**

The Relay Server configuration file is deployed to all computers in the Relay Server farm.

For information about which versions of Apache on Linux are supported, see [http://www.sybase.com/detail?id=1061806](http://www.sybase.com/detail?id=1061806).

**See also**
- “Relay Server configuration file” on page 27
- “Performance tips” on page 12
- “Relay Server State Manager” on page 23
- “Relay Server State Manager as a service” on page 23
- “Relay Server State Manager (rshost) command line syntax” on page 24
- “File Hiding utility (dbfhide)” on page 42

**Concurrent connections**

### Relay Server version

This ONLY applies to Relay Server 12.0.x and up on Apache Web server. This does NOT apply to Relay Server 11.0.x on Apache Web server.

The Apache Web server controls concurrent connections (simultaneous requests) using the max clients directive. By default, the max clients directive is set to 256. If more than 256 concurrent connections are established with the Apache Web server, connections over the 256 limit are queued normally based on the listen backlog directive. By default, the listen backlog directive is set to 511.

For the Apache Web server to handle more than 256 concurrent connections, the max clients directive needs to be set in the `httpd.conf` file. If the max client directive is increased, the server limit directive must also be modified to increase the number of Apache processes in the Web server.
Relay Server 12.0.x and up incorporate a semaphore manager to manage semaphore use by the Relay Server. As a result, there is no need to increase the "semaphore sets" in the system when changing the max clients and server limit directives.

To increase the number of concurrent connections, add the following lines to `httpd.conf`

```plaintext
ServerLimit 1000
MaxClients 1000
```

Other Apache directives that can be adjusted in busy Web servers include:

```plaintext
MaxSpareServers
MinSpareServers
StartServers
```
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 service.

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 in the `/tmp` directory.

**Note**
The Apache user process must have write permissions to the specified `tmp` directory.

On a graceful shutdown, the State Manager renames the log file to a file of the form `<ymmd><nn>.log` where `<ymmd>` 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 service is the recommended method. 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"
```

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

Relay Server State Manager as a service

The State Manager can be started as a service by using the Service utility (dbsvc).

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

**To set up an auto-started State Manager service named RelayServer on Windows**

```
dbsvc -as -s auto -t rshost -w RelayServer "%SQLANY16%\RelayServer\IIS\BinXX\Server\rshost.exe" -q -qc -f "%SQLANY16%\RelayServer\IIS\BinXX\Server\rs.config" -o "c:\temp\ias_relay_server.log"
```

**To set up an auto-started State Manager service named RelayServer on Unix**

```
dbsvc -y -a <apache-user> -t rshost -w RelayServer -q -qc -f /<your-director>/rs.config -os 100K -ot /tmp/rs.log
```
Remarks
The syntax of dbsvc on Windows is different than Unix. In Unix, you do not specify the full path of the executable as the first parameter after -w option argument.

Use full paths only.

In Unix, use a user account (if possibly the same) so that Apache-user processes can attach to the State Manager shared memory and be able to read it and write to it.

Start the service
\[ \text{dbsvc.exe -u rs} \]

To stop the service
\[ \text{dbsvc.exe -x rs} \]

To uninstall the service
\[ \text{dbsvc.exe -d rs} \]

See also
- “Options section” on page 32.

Relay Server State Manager automatic start
The State Manager process is started automatically when the first Outbound Enabler connects to the Relay Server. The default log file location is \%temp\%ias_relay_server_host.log.

Relay Server State Manager (rhost) command line syntax
\[ \text{rhost [option]+} \]

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

<table>
<thead>
<tr>
<th>rhost options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-f filename</td>
<td>Indicates the name of the Relay Server configuration file.</td>
</tr>
<tr>
<td>-o filename</td>
<td>Indicates the name of the file to use for logging. The default log file location for Windows is %temp%ias_relay_server_host.log when the -o option is not specified.</td>
</tr>
<tr>
<td>rshost options</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>-os size</td>
<td>Controls the size of the log file and provides additional information in the log file banner. When -os is specified, the old log is renamed using the <code>&lt;yymmdd&gt;&lt;nn&gt;.olg</code> format. The log banner is rewritten to the new active log file, with the addition of the computer name, processor architecture, build target and operating system information.</td>
</tr>
<tr>
<td>-oq</td>
<td>Prevents a popup window if there is a startup error.</td>
</tr>
<tr>
<td>-q</td>
<td>Runs in a minimized window.</td>
</tr>
<tr>
<td>-qc</td>
<td>Closes the window on completion.</td>
</tr>
<tr>
<td>-u</td>
<td>Updates the configuration of a running Relay Server.</td>
</tr>
<tr>
<td>-ua</td>
<td>Archives the log file to <code>&lt;yymmdd&gt;&lt;nn&gt;.log</code> and truncates the file.</td>
</tr>
</tbody>
</table>
Relay Server configuration file

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

- Relay Server section
- Backend farm section
- Backend server section
- Options section

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 side of an equal sign and its value on the right 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 configuration files can be created, imported and deployed using the Relay Server plug-in for Sybase Central.

See also
- “Relay Server plug-in for Sybase Central” on page 49
- “Relay Server farm configuration updates” on page 45

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 backend server farm. A backend server farm is a group of homogeneous backend servers. A client making a request through the Relay Server farm must specify the backend server farm it is targeting. There is one backend farm section for each backend 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:

- **active_cookie**  Specifies whether or not a cookie is set to retain client-server affinity.
  - yes  This is the default setting. To maintain client-server session affinity, the Relay Server injects a standard HTTP set-cookie command with a proprietary cookie name in the response.
  - no  An active cookie is not set. Use this option when the backend farm is serving a sessionless browser application. For example, when the backend farm is providing a sessionless SQL Anywhere web service.

For best results, set this control as follows:

<table>
<thead>
<tr>
<th>Backend server type</th>
<th>active_cookie setting</th>
<th>active_header setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobiLink</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
For a MobiLink server back end, setting both active_cookie and active_header to yes should always work. However, setting both to yes may put redundant session information in every HTTP request/response in a session. To potentially save on cumulative on-the-wire byte charges, you may be able to set either active_cookie to no. You should test all network scenarios to make sure the chosen settings work in all cases.

- **active_header**  
  Specifies whether or not a header is set to maintain client-server session affinity.
  - yes (Default.) This is the default setting. To maintain client-server session affinity, the Relay Server injects a proprietary header in the response in case intermediaries tamper with the active_cookie.
  - no A proprietary header is not set. Setting this option cuts down on traffic volume if the backend farm is serving only browser applications, or if the active_cookie is working well for all the clients of this backend farm.

- **renew_overlapped_cookie**  
  In the case where a client is sharing a standard cookie across multiple concurrent connections, timeout errors can occur. This issue appears in the Relay Server log.
  - yes (Default.) When renew_overlapped_cookie is set to yes, the Relay Server detects the cookie overlap for the farm that has this property explicitly turned on and renews the overlapping cookie by creating a new affinity binding. The request with the renewal is still routed to the same backend server, but not the same backend connection as the ongoing request that it overlaps with, but rather a new backend connection is created instead.
  - no This should be set to no if this behavior is not desired.

- **backend_security**  
  Specifies the level of security required of an Outbound Enabler in the backend server farm to connect to the Relay Server farm. The possible values are:
  - on Indicates that all connections from the backend farm must by made using HTTPS.
  - off Indicates that all connections from the backend farm must be made using HTTP.

  This property is optional. If no value is specified, either HTTP or HTTPS can be used to connect.

- **client_security**  
  Specifies the level of security the backend 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.

- **description**  
  Enter a custom description to a maximum of 2048 characters. This property is optional.
- **enable**  
  Specifies whether to allow connections from this backend server farm. Possible values are:
  
  - **Yes**  
    Allow connections from this backend server farm.
  
  - **No**  
    Disallow connections from this backend server farm.

  The default is Yes. This property is optional.

- **id**  
  The name assigned to the backend server farm, to a maximum of 2048 characters.

- **forward_x509_identity**  
  The SAP NetWeaver Gateway provides several means of authenticating clients, including X.509 certificate forwarding through trusted intermediaries. When this property is set to yes, Relay Server can extract forwarded client identity information from a trusted forwarder and forward it to the SAP NetWeaver Gateway or Web Dispatcher using HTTP headers. The default setting is no.

- **forwarder_certificate_issue**  
  In the case where a chain of SAP intermediaries exists, the client identity headers may already be present in the request. However, not all clients may be granted permission to act as forwarders. The default behavior, therefore, is to replace the existing headers with the identity of the forwarder. To grant permission for a forwarder to forward other client identities, you can set `forwarder_certificateIssuer=match-string` and `forwarder_certificateSubject=match-string`, where `match-string` is checked against a serialized form of the corresponding compound name field in the certificate. You can use `?` to match any character and `*` to match any string. Use `\` as the leading escape character for `?`, `*`, or `\` if they need to be matched literally.

  For example:

  ```
  forwarder_certificateIssuer = 'CN = quicksigner, OU = security department, O = my org, L = my city, S = my state, C = my country'
  ```

- **forwarder_certificate_subject**  
  In the case where a chain of SAP intermediaries exists, the client identity headers may already be present in the request. However, not all clients may be granted permission to act as forwarders. The default behavior, therefore, is to replace the existing headers with the identity of the forwarder. To grant permission for a forwarder to forward other client identities, you can set `forwarder_certificateIssuer=match-string` and `forwarder_certificateSubject=match-string`, where `match-string` is checked against a serialized form of the corresponding compound name field in the certificate. You can use `?` to match any character and `*` to match any string. Use `\` as the leading escape character for `?`, `*`, or `\` if they need to be matched literally.

  For example:

  ```
  forwarder_certificateSubject = 'CN = mySapWD???.my.com, OU = Sybase, O = SAP, *
  ```

- **max_client_buffer**  
  On occasion, shared memory resources can become exhausted due to issues with excessive buffering of server responses, which can include large numbers of clients, slow-reading clients, or large HTTP responses. `max_client_buffer = memory size`, allows you to specify a limit on the memory buffer size for each client. The default value is 1 MB. The maximum value is 4 GB.

- **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 logging. All HTTP requests are written to the log file.

○ 2  Request logging. Provides a more detailed view of HTTP requests.

○ 3 or higher  Detailed logging. Used primarily for Technical Support.

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

See also
“Affinity” on page 6

Backend server section

The backend server section defines a backend 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 backend server. There is a backend server section for each Outbound Enabler connecting to the Relay Server farm. The backend server section also assigns a backend server to a backend server farm.

The following backend servers are supported for use with the Relay Server:

● Afaria
● Mobile Office
● MobiLink
● Mobile Office
● SQL Anywhere
● Unwired Server
● Sybase Unwired Platform

Note
Refer to your license agreement or the SQL Anywhere Components by Platforms page for information about which backend servers are supported. See http://www.sybase.com/detail?id=1061806.

This section is identified by the backend_server keyword.

Backend server section properties

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

● description  Enter a custom description to a maximum of 2048 characters. This property is optional.
• **enable**  Specifies whether to allow connections from this backend server. Possible values are:
  
  ○ **Yes**  Allows connections from this backend server.
  
  ○ **No**  Disallows connections from this backend server.

The default is Yes. This property is optional.

• **farm**  The name of the backend server farm that this backend server belongs to.

• **id**  The name assigned to the backend server connection, to a maximum of 2048 characters.

• **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 backend server connection, to a maximum of 2048 characters. 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 logging. All HTTP requests are written to the log file.
  
  ○ **2**  Request logging. Provides a more detailed view of HTTP requests.
  
  ○ **3 or higher**  Detailed logging. Used primarily for Technical Support.

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

---

**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:
● **shared_mem**  Specifies the maximum amount of shared memory that the Relay Server uses for state tracking. You may want to change this setting when one or more of the following conditions occurs:

- Improved speed in the network between the Relay Server and the Outbound Enabler
- Significant growth in the number of backend farms
- Significant growth in the size of the backend farm
- Significant growth in the number of clients
- Significant growth in HTTP response size
- Addition of slower clients or slower networks

The default value is 10 MB. The maximum setting is 4 GB. This property is optional.

● **up_pad_size**  Inserts a pad of maximum transmission unit (MTU) size to improve latency when sparse numbers of small requests are being uploaded via the RSOE channel. The default value is 1460, which is the optimum MTU size of (typically) 1500, less 40 bytes for TCP and IP overhead. To turn padding off, set up_pad_size to 0.

To determine the optimum value for the MTU on Windows you can use the ping utility and the -f and -l options (the -f option indicates that data is not to be fragmented; the -l option specifies the data size). The goal is to set the MTU so that there is no data fragmentation. For example:

```
ping -f -l 1494 127.0.0.1
Pinging 127.0.0.1 with 1494 bytes of data:
Packet needs to be fragmented but DF set.
...
ping -f -l 1492 127.0.0.1
Pinging 127.0.0.1 with 1492 bytes of data:
Reply from 127.0.0.1: bytes=1492 time<1ms TTL=128
```

In this example, the maximum data size that can be pushed without fragmentation is 1492. The ping has eight bytes of ICMP overhead and the MTU is therefore 1500.

To determine the optimum value for the MTU on Unix, use `ifconfig` or `ping` with the -M hint option (to select the Path MTU Discovery strategy) and the -s packetsize option (specifies the number of data bytes to be sent. Refer to the man page of the ping utility for additional information.

● **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 logging. All HTTP requests are written to the log file.
- **2**  Request logging. Provides a more detailed view of HTTP requests.
- **3 or higher**  Detailed logging. Used primarily for Technical Support.

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:

```plaintext
# # 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 backend 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 backend server and forward backend 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 backend server that the Outbound Enabler is servicing. Once a response is received from the backend server, it gets forwarded to the Relay Server from which it received the corresponding request using the down channel.

Note
The following backend servers are supported for use with the Relay Server:

- Afaria
- Mobile Office
- MobiLink
- Mobile Office
- SQL Anywhere
- Unwired Server
- Sybase Unwired Platform

Refer to your license agreement or the SQL Anywhere Components by Platforms page for information about which backend servers are supported. See http://www.sybase.com/detail?id=1061806.

Outbound Enabler syntax

\texttt{rsoe \{ option \}+}
rsoe @{ filename | environment-variable } ...

Parameters

Options  The following options can be used with the Outbound Enabler. Options that have defaults are optional. At a minimum, the Outbound Enabler needs to supply the connection string for the Relay Server (-cr), the farm (-f) and server (-id) names. If a security token is configured, it must also be specified (-t).

<table>
<thead>
<tr>
<th>rsoe options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@data</td>
<td>Reads options from the specified environment variable or configuration file. 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 42.</td>
</tr>
<tr>
<td>rssoe options</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| `-cr "connection-string"` | Specifies 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.  
    See “host” [MobiLink - Client Administration].  
  - **port**    The port the Relay Server is listening on. This is required.  
    See “port” [MobiLink - Client Administration].  
  - **http_userid** Userid for authentication. Optional. You should consult your web server (or proxy) documentation to determine how to set up HTTP authentication.  
    See “http_userid” [MobiLink - Client Administration].  
  - **http_password** Password for authentication. Optional. You should consult your web server (or proxy) documentation to determine how to set up HTTP authentication.  
    See “http_password” [MobiLink - Client Administration].  
  - **http_proxy_userid** Userid for proxy authentication. Optional. You should consult your web server (or proxy) documentation to determine how to set up HTTP authentication.  
    See “http_proxy_userid” [MobiLink - Client Administration].  
  - **http_proxy_password** Password for proxy authentication. Optional. You should consult your web server (or proxy) documentation to determine how to set up HTTP authentication.  
    See “http_proxy_password” [MobiLink - Client Administration].  
  - **proxy_host** Specifies the host name or IP address of the proxy server. Optional.  
    See “proxy_host” [MobiLink - Client Administration].  
  - **proxy_port** Specifies the port number of the proxy server. Optional.  
    See “proxy_port” [MobiLink - Client Administration]. |
<table>
<thead>
<tr>
<th>rsOE Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url_suffix</td>
<td>URL path to the server extension of the Relay Server. Required.</td>
</tr>
<tr>
<td>https</td>
<td>0 - HTTP (default)</td>
</tr>
<tr>
<td></td>
<td>1 - HTTPS</td>
</tr>
</tbody>
</table>

For https=1, the following options can also be specified:

- **certificate_name**   Common name field of the certificate.
- **certificate_company** Organization name field of the certificate.
- **certificate_unit**   Organization unit field of the certificate.
- **identity**           Provides the credentials to establish mutually-authenticated TLS between the Outbound Enabler and the backend server. Mutual authentication is required for the backend server.
- **identity_password**  Provides the credentials to establish mutually-authenticated TLS between the Outbound Enabler and the backend server. Mutual authentication is required for the backend server.
- **fips**               Choose whether to use FIPS-certified encryption implementations for TLS encryption and end-to-end encryption.
- **trusted_certificates** A file containing a list of trusted root certificates.

To verify the backend server, and only the backend server, set this property to `backend_server_public_cert_filename`.

```
trusted_certificates=backend_server_public_cert_filename
```

For Windows, if `trusted_certificates` is not set, the operating system certificate store is used.

For more information, see “MobiLink client network protocol options” [MobiLink - Client Administration].
<table>
<thead>
<tr>
<th>rsoe options</th>
<th>Description</th>
</tr>
</thead>
</table>
| -cs "connection-string" | Specifies the backend server connection string. The format of the 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 backend server. The default is localhost.  
    See “host” [MobiLink - Client Administration].  
  ○ **port** The port the backend server is listening on. This is required. The default is 0.  
    See “port” [MobiLink - Client Administration].  
  ○ **https** 0 - HTTP (default)  
    1 - HTTPS  
    By default, MobiLink starts up the TCPIP communication protocol. When starting MobiLink for use with the RSOE, be sure to start the communication protocol required by your RSOE configuration. For example, if you specify HTTPS as the backend security, then MobiLink must be started with HTTPS.  
    When the https=1 parameter is included in the -cs option, the default port changes to 443.  
    See “-x mlsrv16 option” [MobiLink - Server Administration]  
    For **https=1**, the following options can also be specified:  
    • **identity** The path and file name of the identity file that is to be used for server authentication. Provides the credentials to establish mutually-authenticated TLS between the Outbound Enabler and the backend server. Mutual authentication is required for the backend server.  
      See “identity” [MobiLink - Client Administration].  
    • **identity_password** An optional parameter that specifies a password for the identity file. When this option is specified, the identity option must also be specified. Provides the credentials to establish mutually-authenticated TLS between the Outbound Enabler and the backend server. Mutual authentication is required for the backend server.  
      See “identity_password” [MobiLink - Client Administration]. |
<table>
<thead>
<tr>
<th>rsoe options</th>
<th>Description</th>
</tr>
</thead>
</table>
| ● trusted_certificates | A file containing a list of trusted root certificates. To verify the backend server, and only the backend server, set this property to `backend_server_public_cert_filename`. 

`trusted_certificates=backend_server_public_cert_filename`

For Windows, if `trusted_certificates` is not set, the operating system certificate store is used. See “trusted_certificates” [MobiLink - Client Administration]. |
| ○ status_url | Enables backend status requests. This option can be set in the RSOE configuration file (for example): `-cs "host=localhost;port=80;status_url=/getstatus/`. The frequency of the backend server liveness ping is set using the `-d` option. 

If `status_url` is specified, the RSOE sends a simple HTTP GET request as follows: 

```
GET /<your-status-url> HTTP/1.1
Host: localhost:80
User-Agent: IAS_OE_BE_Status
Connection: close
```

The RSOE parses the backend server’s HTTP response and looks for `AVAILABLE=<accept-value>` in the BODY of the HTTP response, where `<accept-value>` is one of: TRUE, FALSE, T, F, YES, NO, Y, N, ON, OFF, 1, or 0. If the RSOE receives `AVAILABLE=FALSE|F|NO|N|OFF|0`, it assumes that the backend server is not willing to accept more client requests and terminates its channels to the Relay Server. If the RSOE receives `AVAILABLE=TRUE|T:YES|Y|ON|1`, it re-establishes its channels with the Relay Server and resumes sending client requests to the backend server. |
<p>| -d seconds | Sets the frequency of the backend server liveness ping and backend server status request. The default is 5 seconds. |
| -dl | Use this option to display log messages in the Relay Server Outbound Enabler console. By default, log messages are not displayed for verbosity levels 1 and 2. |
| -f farm | Specifies the name of the farm that the backend server belongs to. |
| -id id | Specifies the name assigned to the backend server. |
| -o file | Specifies the file to log output messages to. |
| -oq | Prevents the appearance of the error window when a startup error occurs. |</p>
<table>
<thead>
<tr>
<th>rsoe options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-os size</td>
<td>Sets the maximum size of the message log files. The minimum size limit is 10 KB.</td>
</tr>
<tr>
<td>-ot file</td>
<td>Truncates the specified log file and logs messages to it.</td>
</tr>
<tr>
<td>-q</td>
<td>Run with a minimized window on startup.</td>
</tr>
<tr>
<td>-qc</td>
<td>Shuts down the window on completion.</td>
</tr>
<tr>
<td>-s</td>
<td>Stops the Outbound Enabler.</td>
</tr>
<tr>
<td>-t token</td>
<td>Sets the security token to be passed to the Relay Server.</td>
</tr>
</tbody>
</table>
| -uc          | Starts the rsoe in shell mode. This is the default. Applies to Unix and Mac OS X.  
You should only specify one of -uc, -ui, -um, or -ux. When you specify -uc, this starts the rsoe in the same manner as previous releases of the software. |
| -ud          | Instructs the rsoe to run as a daemon. This option applies to Unix platforms only. |
| -ui          | Starts the rsoe in shell mode if a usable display is not available. This option is for Linux with X window server support.  
When -ui is specified, the server attempts to find a usable display. If it cannot find one, for example because the X window server isn't running, then the rsoe starts in shell mode. |
| -ux          | For Linux, opens the rsoe messages window where messages are displayed.  
When -ux is specified, the rsoe must be able to find a usable display. If it cannot find one, for example because the DISPLAY environment variable is not set or because the X window server is not running, the rsoe fails to start.  
To run the rsoe messages window in quiet mode, use -q.  
On Windows, the rsoe messages window appears automatically. |
**rsoe options**

<table>
<thead>
<tr>
<th>-v level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the verbosity level to use for logging. The <em>level</em> can be 0, 1, 2, or higher (higher levels are used primarily for Technical Support):</td>
</tr>
<tr>
<td>○ 0</td>
<td>Log errors only. Use this logging level for deployment.</td>
</tr>
<tr>
<td>○ 1</td>
<td>Session level logging. This is a higher level view of a synchronization session.</td>
</tr>
<tr>
<td>○ 2</td>
<td>Request logging. Provides a more detailed view of HTTP requests.</td>
</tr>
<tr>
<td>○ 3 or higher</td>
<td>Detailed logging. Used primarily for Technical Support.</td>
</tr>
</tbody>
</table>

Levels 1 and 2 are only written to the log file and are not displayed. To have all log messages displayed, use the -dl option.

**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
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>original-configuration-file</td>
<td>Specifies the name of the original file.</td>
</tr>
<tr>
<td>encrypted-configuration-file</td>
<td>Specifies a name for the new obfuscated file.</td>
</tr>
</tbody>
</table>

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

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

**Integrated Outbound Enabler (recommended for MobiLink)**

By using the oe protocol for the -x option for mlsrv16, you can use an integrated Outbound Enabler instead of the stand-alone Outbound Enabler invoked with the rsoe command. Using the integrated Outbound Enabler has the following advantages:

- Reduced use of system resources, especially sockets.
- Provides a single, integrated log file. Lines printed to the MobiLink server log from the integrated Outbound Enabler has the prefix `<OE>`.
- Deployment is simplified.
● Liveness checks between the Outbound Enabler and the MobiLink server are eliminated.

For more information about how to use the integrated Outbound Enabler, see “-x mlsrv16 option” [MobiLink - Server Administration].

Deployment considerations

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

● **Outbound Enabler as a service**  The Outbound Enabler may also be set up and maintained as a service using the Service utility.

● **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.

See also

● “Outbound Enabler as a service” on page 43

### Outbound Enabler as a service

The Outbound Enabler can be started as a service by using the Service utility (dbsvc). The Service utility is used to create, modify and delete services. For a full listing of usage information, run dbsvc without any options.

**To set up an auto-started RSOE service named oes (Outbound Enabler service) on Windows**

```plaintext
dbsvc -as -s auto -t rsoe -w oes "%SQLANY16%\BinXX\rsoe.exe" -cr "host=relayserver.sybase.com;port=80 " -cs "host=localhost;port=80 " -f FarmName -id ServerName -t token
```

**To set up an auto-started RSOE service named oes (Outbound Enabler service) on Unix**

```plaintext
dbsvc -y -a <some-user-account> -t rsoe -w oes @/<full-dir-path>/oe.config
```

**Remarks**

The syntax of dbsvc on Windows is different than Unix. In Unix, you do not specify the full path of the executable as the first parameter after -w option argument.

Use full paths only.

On Unix, specify the Outbound Enabler parameters in a command file only. Do not use command-line options in the setup dbsvc command.

**Start the service**

```plaintext
dbsvc.exe -u oes
```

**To stop the service**

```plaintext
dbsvc.exe -x oes
```
To uninstall the service

dbsvc.exe -d oes

See also

- “SQL Anywhere web services high availability and scale-out solutions” [SQL Anywhere Server - Database Administration]
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.

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.

**See also**

- “Relay Server State Manager” on page 23
Updating a Relay Server configuration for Microsoft IIS on Windows

You may need to occasionally update Relay Server configuration files to add or change Relay Servers or Relay Server farms and change server and farm properties and options.

Prerequisites

A relay server configuration file for an existing Relay Server farm.

Task

1. For each computer that belongs to the Relay Server farm you are updating, copy the updated configuration file to the %SQLANY16%\RelayServer\IIS\Bin\XX\Server directory under the Relay Server web site home directory.

2. From the %SQLANY16%\RelayServer\IIS\Bin\XX\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.

Results

The Relay Server configuration is updated.

Updating a Relay Server configuration for Apache on Linux

You may need to occasionally update Relay Server configuration files to add or change Relay Servers or Relay Server farms and change server and farm properties and options.

Prerequisites

A relay server configuration file for an existing Relay Server farm.

Task

1. Copy the updated configuration file to the /modules directory under the Apache install directory.

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.
Results

The Relay Server configuration is updated.
Relay Server plug-in for Sybase Central

The Relay Server plug-in for Sybase Central provides an easy way to work with the Relay Server. Use the Relay Server plug-in to:

- Create, import, and deploy Relay Server configuration files.
- View Relay Server configuration file properties.
- Add Relay Servers, Relay Server farms, backend servers and backend server farms.
- View and edit Relay Servers, Relay Server farms, backend servers and backend server farms.

Working with Relay Server configuration files

You can use Sybase Central to work with Relay Server configuration files. From Sybase Central you can:

- Create a Relay Server configuration file.
- Open a Relay Server configuration file.
- Import a Relay Server configuration file.
- Deploy a Relay Server configuration file.

Creating a Relay Server configuration file

Create a Relay Server configuration file to define a Relay Server farm and the backend server farms connecting to that Relay Server farm.

Prerequisites

There are no prerequisites for this task.

Task

1. In the Folders view of Sybase Central, right-click Relay Server 16 and click New » Configuration File.
2. Browse to the directory where you want the configuration file saved on the computer running Sybase Central. This is not the same as the deployment location.
3. In the File Name field, type the name of the configuration file. Normally this would be rs.config.
4. Ensure the .config extension is selected in the Save As Type field.
5. Click Save.
Results

A Relay Server farm is automatically created.

Next

You can now add the necessary Relay Servers and backend servers.

Opening a Relay Server configuration file

Open a Relay Server configuration file to edit configuration settings.

Prerequisites

There are no prerequisites for this task.

Task

1. In the Folders view of Sybase Central, right-click Relay Server 16 and click Open Configuration File.

2. Browse to the directory where the configuration file is located, click the file and click Open.

Results

The configuration file is opened.

Next

You can now edit the file.

Importing a Relay Server configuration file

Import a Relay Server configuration file to reuse settings from another Relay Server setup.

Prerequisites

The configuration file must already exist.

Context and remarks

If the Relay Server requires HTTPS communication, the root certificate for the server needs to be stored in the Java Key and Certificate Management Utility. This can be accomplished using the Java Keytool utility. Sybase Central accesses the Java Key and Certificate Management Utility when the root certificate is required for communication.
Task

1. In the Folders view of Sybase Central, right-click Relay Server 16 and click Import Configuration File.

2. Enter the URL for the existing Relay Server.

3. If the Relay Server requires authentication, enter the User Name and Password and click OK.

Results

The Relay Server configuration file is imported.

Next

You can now edit the file.

Deploying a Relay Server configuration file

Deploy a Relay Server configuration file to configure your Relay Server setup.

Prerequisites

There are no prerequisites for this task.

Context and remarks

If the Relay Server requires HTTPS communication, the root certificate for the server needs to be stored in the Java Key and Certificate Management Utility. This can be accomplished using the Java Keytool utility. Sybase Central accesses the Java Key and Certificate Management Utility when the root certificate is required for communication.

Task

1. In the Folders view, right-click the Relay Server configuration file you want to deploy and click Deploy.

2. Enter the URL for the Relay Server.

3. If the Relay Server requires authentication, enter the User Name and Password and click OK.

4. The Server List page shows existing Relay Servers. To deploy the configuration file to one or more of the Relay Servers, select the server(s) from the list and click Add.

Results

The Relay Server configuration file is deployed.

To remove a Relay Server from the list, select the server(s) and click Remove.
Managing Relay Servers and Relay Server farms

You can use Sybase Central to manage Relay Servers and Relay Server farms. From Sybase Central you can:

- Add Relay Servers to a Relay Server farm.
- View or edit Relay Server properties.
- View or edit Relay Server farm properties.

Adding Relay Servers to a farm

You can have one or more Relay Servers running in a Relay Server farm.

Prerequisites

The Relay Server farm must already exist.

Task

1. In the Folders pane under the configuration file you want to work with, right-click the Relay Server farm you want to add Relay Servers to and click New » Relay Server.
2. Ensure Enable this Relay Server is selected.
3. Enter the path information for the Host you want to connect to. Click Ping to check that a connection can be established to the specified host.
4. Select the communication protocol to use. This can be HTTP or HTTPS.
5. Specify the port(s) to be used for the selected protocol(s).
6. If desired, type a description of the Relay Server in the Description field.
7. Click Apply to continue adding Relay Servers or OK to add the Relay Server and close the Create Relay Server window.

Results

The Relay Server is added to the farm.

Next

You can now view or edit Relay Server properties to make changes to your Relay Server setup.

Viewing or editing Relay Server properties

View or edit Relay Server properties to make changes to your Relay Server setup.
Prerequisites

There are no prerequisites for this task.

Task

1. In the Folders pane, click the Relay Server farm that contains the Relay Server you want to work with. The Relay Servers in that farm are listed in the right pane.

2. Right-click the Relay Server you want to edit or view, and click Properties.

3. Make the necessary changes to the Relay Server properties and click Apply or OK.

Results

The changes are applied to the Relay Server.

Next

You can view or edit Relay Server properties to make changes to your Relay Server farm setup.

Viewing or editing Relay Server farm properties

View or edit Relay Server properties to make changes to your Relay Server farm setup.

Prerequisites

There are no prerequisites for this task.

Task

1. In the Folders pane, right-click the Relay Server farm you want to work with and click Properties.

2. Make the necessary changes to the Relay Server farm properties and click Apply or OK.

Results

Any changes you have made to the Relay Server farm properties are saved.

Managing backend servers and backend server farms

You can use Sybase Central to manage backend servers and backend server farms. From Sybase Central you can:

- Create a backend server farm and add backend servers to it.
Creating a backend server farm

Create a backend server farm. A backend server farm is a group of homogeneous backend servers to which clients make requests through the Relay Server farm.

Prerequisites

There are no prerequisites for this task.

Task

1. In the left pane, right-click the Relay Server configuration file you want to work with, and click New » Backend Server Farm.

2. Ensure Enable This Backend Server Farm is selected.

3. Type the name associated with the new backend server farm.

4. In Client Security choose a protocol for clients to use to connect to the backend server farm.

5. In Backend Security choose a protocol for the Relay Server Outbound Enabler (rsoe) to use to connect to the backend server farm.

6. For client-server affinity, choose the type of server you are using. For a MobiLink HTTP server with standalone Outbound Enabler or a MobiLink server with an embedded Outbound Enabler, click MobiLink. For a typical SQL Anywhere web service click SQL Anywhere. Advanced custom settings are available by clicking Custom as the server type. Once the Custom server type is selected, you have full control over the following affinity settings:
   a. Check the Active Cookie option if you want the Relay Server to use a standard HTTP set-cookie command to retain client-server affinity.
   b. Check the Active Header option if you want the Relay Server to use a proprietary header to retain client-server affinity.

7. Optionally, type a description of the backend server farm in the Description field.

8. Click Apply to continue adding backend server farms or OK to add the backend server farm and close the Create Backend Server Farm window.

Results

The backend server farm is created.

Next

You can view or edit backend server farm properties to manage your server farm setup.
Viewing or editing backend server farm properties

View or edit backend server farm properties to manage your server farm setup.

Prerequisites

There are no prerequisites for this task.

Task

1. In the Folders pane, right-click the backend server farm you want to work with and click Properties.
2. Make the necessary changes to the backend server farm properties and click Apply or OK.

Results

Your changes to the backend server properties are saved.

Next

You can view or edit server properties to modify backend server behavior.

Adding servers to a backend server farm

Add servers to a backend server farm to manage workload.

Prerequisites

There are no prerequisites for this task.

Task

1. In the Folders pane, right-click the backend server farm you want to work with and click New » Backend Server.
2. Ensure Enable This Backend Server is selected.
3. Type the name associated with the new backend server.
4. To enforce MAC address checking, click the Enforce MAC Address Checking checkbox.
5. If you selected MAC address checking, enter the RSOE MAC address 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. Multiple MAC addresses separated by an exclamation mark (!) are reported by the Outbound Enabler if multiple adapters are currently active on your backend server computer.
the most permanent one for the Relay Server to check against. The `ipconfig /all` command on Windows provides a detailed listing of your network adapters together with associated MAC addresses.

6. Specify the security token that is used by the Relay Server to authenticate the backend server connection. You can use a maximum of 2048 characters.

7. If desired, type a description of the backend server in the **Description** field.

8. Click **Apply** to continue adding backend servers or **OK** to add the backend server and close the **Create Backend Server** window.

**Results**

The servers are added to the backend server farm.

**Next**

You can view or edit backend server properties to make changes to your Relay Server farm setup.

### Viewing or editing backend server properties

View or edit server properties to modify backend server behavior.

**Prerequisites**

There are no prerequisites for this task.

**Task**

1. In the **Folders** pane, click the backend server farm that contains the backend server you want to work with. The backend servers in that farm are listed in the right pane.

2. Right-click the backend server you want to edit or view, and click **Properties**.

3. Make the necessary changes to the backend server properties and click **Apply** or **OK**.

**Results**

The changes to the backend server properties are saved.
Relay Server logging and log administration

The Relay Server log file displays the following types of messages:

- **Information**  
  Basic information about your current session.

- **Warning**  
  Warning messages about actions that have occurred.

- **Error**  
  Error messages about actions that have failed.

The default log file name and directory for Windows is `%temp%\ias_relay_server_host.log`. To change the log file name, use the `rshost -o filename` option.

In addition, Relay Server logging supports the following features:

- The Relay Server and Outbound Enabler logs report timestamps in millisecond resolution.  
  Timestamps are reported in RFC 822 local differential format (+/- hhmm).

- When the Relay Server and Outbound Enabler handle HTTP requests that carry the SAP Passport header, the Relay Server and the Outbound Enabler increase the request handling verbosity level to match the trace level requirement contained in the Passport and also add a suffix to the associated log lines to list the key information of the Passport.

**Verbosity**

You can set Relay Server log verbosity to the following levels:

- **0**  
  Log errors only. Use this logging level for deployment. This is the default.

- **1**  
  Request logging. Summaries of HTTP requests are written to the log file in an abbreviated format.

- **2**  
  Request logging. Provides a more detailed view of HTTP requests.

- **3 or higher**  
  Detailed logging. Used primarily for Technical Support.

Errors are displayed regardless of the log level specified, and warnings are displayed if the log level is 1 or higher.

**See also**

“Relay Server State Manager (rhost) command line syntax” on page 24

Relay Server logging and SAP Passports

SAP Passports are used to trace requests from the client through to the backend server. The Passport may contain a directive to increase logging verbosity as it flows through each server. Both the Relay Server and the Relay Server Outbound Enabler respect this directive. In cases where the Relay Server or Outbound Enabler verbosity level is set higher than the level implied by the Passport, the higher level takes effect. This means a user cannot use a Passport with a low trace level to suppress payload logging.
when the Relay Server/Outbound Enabler administrator has set up high verbosity logging in the Relay Server/Outbound Enabler configuration.

SAP Passport version 2 and version 3 are supported.

<table>
<thead>
<tr>
<th>Passport trace level</th>
<th>RS/OE verbosity level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>1</td>
<td>Access level logging</td>
</tr>
<tr>
<td>Medium</td>
<td>4</td>
<td>Debug logging with packet logging (plus request header logging on the Relay Server side)</td>
</tr>
<tr>
<td>High</td>
<td>5</td>
<td>Debug logging with entire payload</td>
</tr>
</tbody>
</table>

The Relay Server and Outbound Enabler log lines associated with a request involving an SAP Passport have suffixes as shown in the following table:

<table>
<thead>
<tr>
<th>SAP Passport version</th>
<th>Log line suffix</th>
<th>Log line example</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>#SAP-PPK#V2#&lt;Transaction-uuid&gt;</td>
<td>I. 2012-07-19 15:06:32.720 &lt;15852.9340.F0B0S5R0&gt; Relaying content-length body #SAP-PPK#V2#8fa46833ea42b94a8181b5bc8da3a33c</td>
</tr>
<tr>
<td>3</td>
<td>#SAP-PPK#V3#&lt;Transaction-uuid&gt;#&lt;Root-context-uuid&gt;#&lt;Connection-uuid&gt;#&lt;Connection-counter&gt;</td>
<td>I. 2012-07-19 15:06:32.909 &lt;15852.14180.F0B0S6R0&gt; Relaying headers #SAP-PPK#V3#a65762116e4e48b6b1fccdc428785d49#dd880e77415f4811bd0b08bcd75cfb74#a6b60d0f83c340bd8e2458ef6bd29774#</td>
</tr>
</tbody>
</table>

**Relay Server Record**

This topic has been updated for build 1823.

The Relay Server Record (RSR) consists of a concise summary of Relay Server processing and includes information about requests, timing, affinity information, request status, and data volumes. You can use the RSR to diagnose relay failures and study performance characteristics.

The Relay Server Record is generated as part of the Relay Server log when verbosity is set to 1 or higher.

The RSR is a single line in the Relay Server log containing many values that together summarize an HTTP request. To assist in interpreting Relay Server Records, the Relay Server log file contains a header
describing the RSR values. The header line is composed of symbols that are also described in the file. The symbols use the following convention:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>b:</td>
<td>Byte count</td>
</tr>
<tr>
<td>c:</td>
<td>Other count</td>
</tr>
<tr>
<td>i:</td>
<td>ID or numeric code</td>
</tr>
<tr>
<td>m:</td>
<td>Time measured in milliseconds</td>
</tr>
<tr>
<td>x:</td>
<td>Hex number</td>
</tr>
<tr>
<td>name:string</td>
<td>Variable length named string values. This may include: Relay Server error names, Relay Server error parameters, and SAP Passport information.</td>
</tr>
<tr>
<td>oe</td>
<td>Element reported by the Outbound Enabler</td>
</tr>
<tr>
<td>up</td>
<td>Element associated with the request (excluding the response) from the Relay Server to the backend</td>
</tr>
<tr>
<td>rtp</td>
<td>Element associated with round-trip transport and processing of the last up and first down packet</td>
</tr>
<tr>
<td>dn</td>
<td>Element associated with the request (excluding the response) from the backend to the Relay Server</td>
</tr>
<tr>
<td>in</td>
<td>Time spent waiting for input to read</td>
</tr>
<tr>
<td>out</td>
<td>Time spent waiting for input to write</td>
</tr>
<tr>
<td>A.B</td>
<td>This notation indicates that B is a child component, sub-process, or an aspect of A.</td>
</tr>
<tr>
<td>pkt/packet</td>
<td>Refers to the packet created by the Relay Server and/or the Outbound Enabler for communication over the up/down channel</td>
</tr>
</tbody>
</table>

The symbols are compounded together (as field names) to refer to particular kinds of data. For example, **m:up.out** corresponds to the sum of time (m:) spent in the up channel (up) writing packets (.out) within the relaying period. The dot notation (.) also indicates that **out** is a sub-process of **up**.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>x:sfp</td>
<td>Session fingerprint (a component of affinity information)</td>
</tr>
<tr>
<td>i:oe.sidx</td>
<td>Session index assigned by the Outbound Enabler</td>
</tr>
<tr>
<td>Field name</td>
<td>Data type</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>flag[0]</td>
<td>Affinity decision; Values: n=new, c=continue, h=homed, r=renew x=expired. <em>new</em> indicates a new affinity; <em>continue</em> indicates a subsequent request in an established affinity session; <em>homed</em> indicates a new affinity with the designated backend server; <em>renew</em> indicates that there is a collision, so the affinity information is renewed and a new section is opened up; <em>expired</em> indicates that the Relay Server has signaled the client to expire the affinity cookie.</td>
</tr>
<tr>
<td>flag[1]</td>
<td>Request persistence; Values: p=persistent, n=non-persistent</td>
</tr>
<tr>
<td>flag[2]</td>
<td>Request transfer encoding; Values: k=chunked, l=content-length</td>
</tr>
<tr>
<td>flag[3]</td>
<td>Response persistence; Values: p=persistent, n=non-persistent, u=unknown</td>
</tr>
<tr>
<td>flag[4]</td>
<td>Response transfer encoding; Values: k=chunked, l=content-length, u=unknown</td>
</tr>
<tr>
<td>b:up</td>
<td>Request size in bytes</td>
</tr>
<tr>
<td>b:dn</td>
<td>Response size in bytes</td>
</tr>
<tr>
<td>c:up: pkt</td>
<td>Number of upward request packets sent to the Outbound Enabler</td>
</tr>
<tr>
<td>m:up</td>
<td>Request relaying period involving interleaving reads from the client, writes to the up channel, and request packetization</td>
</tr>
<tr>
<td>m:up: in</td>
<td>Sum of time spent waiting for request payload from the client within the request relaying period</td>
</tr>
<tr>
<td>m:up: out</td>
<td>Sum of time spent writing packets to the up channel within the relaying period</td>
</tr>
<tr>
<td>m:rtp</td>
<td>Round-trip processing period from the sending of the last request packet to the receiving of the first response packet between the Relay Server and the backend server, including the backend processing time</td>
</tr>
<tr>
<td>m:oe: rtp</td>
<td>Round-trip processing period from the sending of the last request packet to the receiving of the first response packet between the Outbound Enabler and the backend server, including the backend processing time</td>
</tr>
<tr>
<td>m:rtp: kpi</td>
<td>Round-trip relay processing and transport time of the last request packet and the first response packet between the Relay Server and the Outbound Enabler, calculated as <em>(m:rtp - m:oe.rtp)</em></td>
</tr>
<tr>
<td>c:dn: pkt</td>
<td>Number of downward response packets received from the Outbound Enabler</td>
</tr>
<tr>
<td>m:dn</td>
<td>Response relaying period involving interleaved reads from the down channel and writes to the client</td>
</tr>
<tr>
<td>m:dn: in</td>
<td>Sum of time spent waiting for response packets from the down channel within the response relaying period</td>
</tr>
<tr>
<td>m:dn: out</td>
<td>Sum of time spent waiting for writing response payload to the client within the response relaying period</td>
</tr>
<tr>
<td>Field name</td>
<td>Data type</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>m:oe.dn</td>
<td>Response receiving period observed by the Outbound Enabler. This period overlaps with time counted in m:dn.in and m:dn.out</td>
</tr>
<tr>
<td>m:close</td>
<td>Elapsed time between the end of m:dn and exiting the rs_client extension</td>
</tr>
<tr>
<td>b:dn.maxLQ</td>
<td>Peak memory usage of the local response packet queue of this request</td>
</tr>
<tr>
<td>c:dn.maxSQ</td>
<td>Peak number of response packets in the shared memory response packet queue of this request</td>
</tr>
<tr>
<td>i:dn.stts</td>
<td>HTTP response status</td>
</tr>
<tr>
<td>i:err</td>
<td>Error ID</td>
</tr>
<tr>
<td>i:warn</td>
<td>Warning ID</td>
</tr>
<tr>
<td>m:appTO</td>
<td>Application timeout of the request</td>
</tr>
<tr>
<td>err</td>
<td>Name of the error</td>
</tr>
<tr>
<td>warn</td>
<td>Name of the warning</td>
</tr>
<tr>
<td>oe.err</td>
<td>Name of the Outbound Enabler error</td>
</tr>
<tr>
<td>oe.err.p0</td>
<td>First error parameter from the Outbound Enabler</td>
</tr>
<tr>
<td>oe.err.p1</td>
<td>Second error parameter from the Outbound Enabler</td>
</tr>
<tr>
<td>oe.err.p2</td>
<td>Third error parameter from the Outbound Enabler</td>
</tr>
<tr>
<td>up.ua</td>
<td>User agent header of the request</td>
</tr>
<tr>
<td>up.uq</td>
<td>URL query parameters in name=value pairs. This can be useful for tagging information within the request when SAP Passports are not available.</td>
</tr>
<tr>
<td>up.AfHdr</td>
<td>Affinity</td>
</tr>
<tr>
<td>up.cookie</td>
<td>Cookie header in request</td>
</tr>
<tr>
<td>N/A</td>
<td>Composite request prefix (located at the end of each line):</td>
</tr>
<tr>
<td></td>
<td>&lt;processId.threadId.F(BEFarmIdx)B(BEServerIdx)S(BESessionNum)R(RequestIdx)&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;BEFarmName&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;BeServerName&gt;</td>
</tr>
</tbody>
</table>

For example:
<11436.4592.F0B0S0R0> <RSTEST02.F0> <S0>
A Relay Server Record looks similar to the following sample:

```
I. 2012-11-09 02:35:11.296-0500 RSR header: x:sfp i:oe.sidx_flag
  b:up  b:dn   c:up.pkt  m:up   m:up.in  m:up.out  m:rtp
  m:oe.rtp m:rtp.kpi  c:dn.pkt  m:dn   m:dn.in  m:dn.out  m:dn oe
  | m:close  b:dn.maxLQ c:dn.maxSQ i:dn.stts  i:err  i:warn
  m:appTO ...other-variable-length-elements...
I. 2012-11-09 02:35:38.192-0500 RSR row: d9d72990 0 npkp
  1024477 1031252 | 444 2366 2305 0 |
  1001 999 2 | 107 17764 17760 |
  17741 24 | 0 953152 87 |
  0 60000 <err:RSE_NO_ERROR> <warn:RSW_NO_WARNING> <oe.err:N/A>
  <oe.err.p0:> <oe.err.p1:> <oe.err.p2:> <up.userAgent:RSTestClient> <up.uq:>
  <up.AfHdr:> <up.cookie:> <11436.4592.F0B0S0R0> <RTEST02.F0> <S0>
I. 2012-11-09 02:36:02.278-0500 RSR row: d9d72990 0 cpkp
  1024534 1031211 | 428 2440 2379 0 |
  1000 999 1 | 124 17763 17741 |
  17741 22 | 0 951657 109 |
  0 60000 <err:RSE_NO_ERROR> <warn:RSW_NO_WARNING> <oe.err:N/A>
  <oe.err.p0:> <oe.err.p1:> <oe.err.p2:> <up.userAgent:RSTestClient> <up.uq:>
  <up.AfHdr:ias-rs-sessionid="kCnX2QAAAAAAAAAAUzAA";> <up.cookie:>
  <11436.4592.F0B0S0R0> <RTEST02.F0> <S0>
I. 2012-11-09 02:36:26.515-0500 RSR row: d9d72990 0 cnknk
  1024529 1031230 | 400 2440 2379 0 |
  1000 999 0 | 153 17924 17899 |
  17899 22 |
  0 60000 <err:RSE_NO_ERROR> <warn:RSW_NO_WARNING> <oe.err:N/A>
  <oe.err.p0:> <oe.err.p1:> <oe.err.p2:> <up.userAgent:RSTestClient> <up.uq:>
  <up.AfHdr:ias-rs-sessionid="kCnX2QAAAAAAAAAAUzAA";> <up.cookie:>
  <11436.4592.F0B0S0R0> <RSTEST02.F0> <S0>
I. 2012-11-09 02:36:31.830-0500 RSR row: 63685111 4294967295 upkuu
  1024000 0 |
  0 0 0 2379 0 |
  0 0 2439 |
  0 0 0 0 0 0 0 1001 |
  0 0 0 0 0 0 |
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
See also

“Affinity” on page 6

Outbound Enabler Record

You can use the Outbound Enabler Record (OER) to diagnose relay failures and study performance. The OER consists of a concise summary of the Outbound Enabler processing of a single request and includes information about requests, timing, important debugging information, request status, and data volumes. The Outbound Enabler Record is generated as part of the Outbound Enabler log when verbosity is set to 1 or higher. There is one OER data line per HTTP request.

The Outbound Enabler Record is comprised of a set of data types (represented by symbols) and values. The symbols and their associated data types are listed in the following table:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>b:</td>
<td>Byte count</td>
</tr>
<tr>
<td>c:</td>
<td>General numeric counter</td>
</tr>
<tr>
<td>i:</td>
<td>ID or numeric code</td>
</tr>
<tr>
<td>m:</td>
<td>Time measured in milliseconds</td>
</tr>
<tr>
<td>x:</td>
<td>Hex number</td>
</tr>
<tr>
<td>name:string</td>
<td>Variable length named string values. This may include: user agent information, affinity information, cookie information, Outbound Enabler errors, Outbound Enabler error parameters, the name of the Relay Server, Relay Server error names, and the label.</td>
</tr>
<tr>
<td>up</td>
<td>Element associated with the relaying request (excluding the response) to the backend</td>
</tr>
<tr>
<td>rtp</td>
<td>Element associated with round-trip transport and processing of the last up and first down packet</td>
</tr>
<tr>
<td>dn</td>
<td>Element associated with the relaying request (excluding the response) from the backend to the Relay Server</td>
</tr>
</tbody>
</table>

The symbols are compounded together (as field names) to refer to particular kinds of data. For example, $b\text{:up.maxReqQ}$ corresponds to the byte count (b:) of the peak memory usage of the per-request upward (up) packet queue (maxReqQ).
### Field name | Data type
---|---
i:oidx | Relay Server index in the Relay Server farm
i:ridx | Temporary request index assigned by the Relay Server
i:sidx | Application session index assigned by the Outbound Enabler to the first request
x:snum | Session serial number assigned by the Relay Server to the first request
x:sfp | Session fingerprint assigned by the Relay Server on the first request
m:up | Request relaying period starting from the sending of the first request packet to the sending of the last request packet
m:rtp | Round-trip processing period from the sending of the last request packet to the receiving of the first response packet between the Outbound Enabler and the backend server, including the backend processing time
m:dn | Response relaying period starting from the arrival of the first response packet to the arrival of the last response packet
m:close | Idle period from the arrival of the last response packet to the garbage collection or reuse of the affinity context
b:up.maxReqQ | Peak memory usage of the per-request upward packet queue
b:dn.maxShrQ | Peak memory usage of the shared downward packet queue during the response relaying period of this request
i:err | Error ID
err.p0 | First error parameter from the Outbound Enabler
err.p1 | Second error parameter from the Outbound Enabler
err.p2 | Third error parameter from the Outbound Enabler

An Outbound Enabler Record looks similar to the following sample:

```
I. 2012-11-09 02:35:12.821-0500 <OEHost> OER header:  i:oidx  i:ridx
  i:sidx  x:snum  x:sfp  |  m:up  m:rtp  m:dn  m:close  
  b:up.maxReqQ  b:dn.maxShrQ  |  i:err  ...other-variable-length-elements...
I. 2012-11-09 02:35:41.070-0500 <Backend-0000> OER row:  0
  0  0  0  d9d72990  |  2364  999  23
  20620  49723  65455  |  0 <err:OEE_NO_ERROR> <err.p0:>
  <err.p1:> <err.p2:>
I. 2012-11-09 02:36:05.153-0500 <Backend-0000> OER row:  0
  0  0  0  d9d72990  |  2364  999  23
  20615  56968  65455  |  0 <err:OEE_NO_ERROR> <err.p0:>
  <err.p1:> <err.p2:>
I. 2012-11-09 02:36:08.614-0500 <Backend-0000> OER row:  0
  0  0  0  d9d72990  |  2364  999
```
Remote administration of the Relay Server log file

Remote administration of the Relay Server log file is provided through the AdminChannel class in rstool.jar.

Command line usage:

```java
java ianywhere.ml.rs.AdminChannel [{options}]...
```

where options can be one or more of the following

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-url rsAdminUrl</td>
<td>Points to the rs_admin extension</td>
</tr>
<tr>
<td>-uid user -pwd password</td>
<td>Provides credentials for HTTP authentication to access rs_admin</td>
</tr>
<tr>
<td>-ping</td>
<td>Ping the rs_admin extension</td>
</tr>
<tr>
<td>-getRSConfig</td>
<td>Retrieve the Relay Server configuration</td>
</tr>
<tr>
<td>-setRSConfig</td>
<td>Update and backup the Relay Server configuration</td>
</tr>
<tr>
<td>-hello</td>
<td>Negotiate administration protocol version and ping</td>
</tr>
<tr>
<td>-archiveRSLog</td>
<td>Truncate and archive the current online Relay Server log file</td>
</tr>
<tr>
<td>-xol outFile none</td>
<td>begin-Time none</td>
</tr>
<tr>
<td>-xrl outFile none</td>
<td>begin-Time none</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>-?</td>
</tr>
</tbody>
</table>
For example:

```
java.exe -cp rstool.jar ianywhere.ml.rs.AdminChannel -url https://rs.my.com/rs16/admin/rs_admin.dll -uid me -pwd changit -hello
java.exe -cp rstool.jar ianywhere.ml.rs.AdminChannel -url https://rs.my.com/rs16/admin/rs_admin.dll -uid me -pwd changit -xrl rr.xrl none none 1 "RSR (element|header|row)"
```
Sybase Hosted Relay Service

The Sybase Hosted Relay Service is a farm of Relay Servers hosted by Sybase. It is intended to ease the development of mobile applications that use MobiLink data synchronization and to simplify the evaluation process for developers, especially where data is sent using public wireless networks. In particular, you do not need to ask your IT department to install anything or open any holes in your corporate firewall for inbound connections. All communication between MobiLink and the hosting service uses HTTP(S) via an outbound connection initiated by MobiLink.

The Sybase Hosted Relay Service is not intended for production deployments. Before deploying your production application, you must first install the Relay Server in your own corporate infrastructure.

Subscribing to the Sybase Hosted Relay Service

To use the Sybase Hosted Relay Service you must first subscribe to it.

Prerequisites

There are no prerequisites for this task.

Task

1. Go to http://relayservice.sybase.com/account. This takes you to the Sybase Hosted Relay Service home page.

2. Create an account by clicking Register.

3. You are asked to specify a Subscription ID (choose one that is unique to your organization) and Password, provide contact information for your self and your organization, and agree to the Hosted Relay Service Terms of Service. Click Submit.

Results

Once you have successfully registered, an email is sent to you confirming your registration.

Next

You can now log in to the Sybase Hosted Relay Serve using your subscription ID and password and add a server farm.

Adding a server farm

Before you can use the Sybase Hosted Relay Service you need to add a farm.

Prerequisites

There are no prerequisites for this task.
**Task**

1. Click the type of farm you want to add. Choose from:
   - Add New MobiLink Farm
   - Add New SQL Anywhere Web Read-Write Farm
   - Add New SQL Anywhere Web Read Farm
   - Add New SQL Anywhere Web Read-Only Farm
   - Add New SQL Remote Message Server Farm
   - Add New Afaria Farm
   - Add New SAP Sybase Mobile Office Farm
   - Add New Sybase Unwired Platform Farm

2. Enter a unique **Farm Name** to describe the server farm.

3. Provide a unique name for each server in the farm. You can specify a maximum of two servers.

4. Click **Create Farm**. A confirmation page is displayed if the farm was successfully added.

5. Click **Configuration Instructions** to learn more about using the service. The instructions are based on the information you provided.

6. Click **Log Out** when you are done.

**Results**

The server farm is added.
The Relay Server with MobiLink

The following sections provides information about using the Relay Server with MobiLink.

See also

- For information about which operating systems and browsers are supported for the Relay Server, see http://www.sybase.com/detail?id=1002288
- For information about deploying the Outbound Enabler, see “MobiLink server deployment” [MobiLink - Server Administration]

Connecting a client to the Relay Server farm

Once a Relay Server farm has been properly configured, a client connects to the Relay Server farm using the following URL:

http://<Relay Server client extension URL>/<farmname>

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Relay Server client extension URL&gt;</td>
<td>For Microsoft IIS on Windows, &lt;domain name&gt;&lt;relay-server.sybase.com&gt;/rs/client/rs_client.dll For Apache on Linux, &lt;domain name&gt;/cli/iarelayserver Use relayserver.sybase.com as the &lt;domain name&gt; if you are using the publicly available Sybase Hosted Relay Service. For information about subscribing to the service as well as instructions for setting up your backend servers, see “Subscribing to the Sybase Hosted Relay Service” on page 67.</td>
</tr>
<tr>
<td>&lt;farmname&gt;</td>
<td>Identifies the backend farm (a group of backend servers) that Relay Server forwards the client request to.</td>
</tr>
</tbody>
</table>

SQL Anywhere MobiLink client connection example

A SQL Anywhere MobiLink client should specify the following options to connect to server farm F1:

- e "ctp=http; adr='host=relayserver.sybase.com; url_suffix=/rs/client/rs_client.dll/F1"

For HTTPS, change http to https.
UltraLite/UltraLiteJ MobiLink client connection example

An UltraLite/UltraLiteJ MobiLink client should set the following properties in the ULSyncParms class to connect to server farm F1:

- Set the stream type to HTTP or HTTPS.
- Set the stream parameters to the following:

  "host=relayserver.sybase.com;url_suffix=/rs/client/rs_client.dll/F1"

Setting up a relay server farm

Before MobiLink clients can connect to a farm, you must configure and deploy the configuration file with the appropriate settings.

Prerequisites

For the purposes of this scenario, use the Microsoft IIS version of the Relay Server.

Context and remarks

Suppose company ABC has developed a mobile application and now wants to set up the deployment runtime to service the mobile application. Initially, the mobile deployment consists of 10000 devices and grows in the future. The customer therefore wants a fault tolerant and load-balanced environment that is able to handle the load today and be easily extended to handle more mobile deployments in the future. Based on the data synchronization characteristics of the mobile application, the customer has determined that the following configuration is needed:

- 2 MobiLink servers
- 2 Relay Servers
- 1 load balancer

Each Relay Server is deployed on its own computer. Two computers, with host names rs1.abc.com and rs2.abc.com are used.

Each MobiLink server is deployed on its own computer. The two MobiLink servers are assigned names ml1 and ml2 and belong to the backend server farm called abc.mobilink.

The load balancer is addressable using the host name www.abc.com.

For maximum security, HTTPS is used by all clients and Outbound Enablers connecting to the Relay Servers. It is assumed that all web servers are equipped with a certificate from a well known Certificate Authority (CA), and the backend server computers all have the corresponding trusted root certificates in their standard certificate store.

Task

1. The first step is to create the Relay Server configuration file.
The filename containing the configuration must be called \textit{rs.config}. For this particular scenario, the following configuration file is used:

```plaintext
# # Options
# [options]
verbosity = 1

# # Define the Relay Server farm
# [relay_server]
host = rs1.abc.com
[relay_server]
host = rs2.abc.com

# # Define the MobiLink backend server farm
# [backend_farm]
id = abc.mobilink
client_security = on
backend_security = on

# # List MobiLink servers that are connecting to the Relay Server farm
# [backend_server]
farm = abc.mobilink
id = ml1
token = mltoken1
[backend_server]
farm = abc.mobilink
id = ml2
token=mltoken2
```

2. Deploy the configuration file \textit{rs.config} along with the Relay Server components to the two computers that are running the Relay Server.

3. Start MobiLink server on the two computers that are running the MobiLink servers using the Integrated Outbound Enabler.

On the computer running MobiLink server with id ml1:

```
mlsrv16 -x oe<config=oe1.txt> -zs ml1 <other ML options>
```

where oe1.txt = `-f abc.mobilink -id ml1 -t mltoken1 -cr
"host=www.abc.com;port=443;https=1"

On the computer running MobiLink server with id ml2:

```
mlsrv16 -x oe<config=oe2.txt> -zs ml2 <other ML options>
```

where oe2.txt = `-f abc.mobilink -id ml2 -t mltoken2 -cr
"host=www.abc.com;port=443;https=1"
See “-x mlsrv16 option” [MobiLink - Server Administration].

**Results**

Once all servers and Outbound Enablers are running, MobiLink clients are able to connect to the farm using the following connection information:

- **HTTPS** protocol
- **host** www.abc.com
- **url_suffix** /rs/client/rs_client.dll/abc.mobilink

**Next**

Connect MobiLink clients to the farm.

**See also**

- “End-to-end encryption” [SQL Anywhere Server - Database Administration]
Index

Symbols

-cr option
  Outbound Enabler syntax [Relay Server], 35
-cs option
  Outbound Enabler syntax [Relay Server], 35
-d option
  Outbound Enabler syntax [Relay Server], 35
-dl option
  Outbound Enabler syntax [Relay Server], 35
-f option
  Outbound Enabler syntax [Relay Server], 35
  Relay Server command line syntax, 24
-id option
  Outbound Enabler syntax [Relay Server], 35
-o option
  Outbound Enabler syntax [Relay Server], 35
  Relay Server command line syntax, 24
-oq option
  Outbound Enabler syntax [Relay Server], 35
  Relay Server command line syntax, 24
-os option
  Outbound Enabler syntax [Relay Server], 35
  Relay Server command line syntax, 24
-ot option
  Outbound Enabler syntax [Relay Server], 35
-q option
  Outbound Enabler syntax [Relay Server], 35
  Relay Server command line syntax, 24
-qc option
  Outbound Enabler syntax [Relay Server], 35
  Relay Server command line syntax, 24
-s option
  Outbound Enabler syntax [Relay Server], 35
-t option
  Outbound Enabler syntax [Relay Server], 35
-u option
  Outbound Enabler syntax [Relay Server], 35
  Relay Server command line syntax, 24
-ua option
  Relay Server command line syntax, 24
-uc option
  Outbound Enabler syntax [Relay Server], 35
-ud option
  Outbound Enabler syntax [Relay Server], 35
-ux option
  Outbound Enabler syntax [Relay Server], 35
-v option
  Outbound Enabler syntax [Relay Server], 35
@data option
  Outbound Enabler syntax [Relay Server], 35

A
active_cookie
  Relay Server configuration file property - backend farm, 28
active_header
  Relay Server configuration file property - backend farm, 29
Afaria server
  Relay Server, supported, 31
affinity
  Relay Server, 6
Apache
  concurrent connections, increasing for Relay Server, 20
deploying the Relay Server, 17
application pool
  creating, 12
architectures
  Relay Server, 1

B
backend farm section
  Relay Server configuration file, 28
backend server farm
  Relay Server, 4
backend server section
  Relay Server configuration file, 31
backend_security
  Relay Server configuration file property - backend farm, 29

C
client
  connecting to a Relay Server farm, 69
client_security
  Relay Server configuration file property - backend farm, 29
configuration files
  Relay Server, 27
  Relay Server format, 34
connections
   Relay Server, increasing concurrent for Apache, 20

D
deploying
   Relay Server, 9
description
   Relay Server configuration file property - backend
     farm, 29
   Relay Server configuration file property - backend
     server, 31

E
enable
   Relay Server configuration file property - backend
     farm, 30

F
farm (see Relay Server farm)
forward_x509_identity
   Relay Server configuration file property - backend
     farm, 30
forwarder_certificate_issue
   Relay Server configuration file property - backend
     farm, 30
forwarder_certificate_subject
   Relay Server configuration file property - backend
     farm, 30

H
hosted Relay Server
   adding a server farm, 67
   logging in, 67
   subscribing, 67
HTTP load balancer
   Relay Server, 4

I
ias-rs-status-refresh
   Outbound Enabler syntax [Relay Server], 35
id
   Relay Server configuration file property - backend
     farm, 30
IIS 7.x
   deploying Relay Server to, 13
IIS 8.x
   deploying Relay Server to, 13
deploying, Microsoft IIS 6.0, 12
deploying, Microsoft IIS 7.x, 17
relays (see Relay Server)
ren_overlapped_cookie
    Relay Server configuration file property - backend farm, 29
rhost (see Relay Server State Manager)
rhost.exe
    location, 9, 13
    Relay Server configuration update, Linux, 46
    Relay Server configuration update, Windows, 46
    Relay Server deployment, 9
    Relay Server State Manager, 45
    Relay Server State Manager command line syntax, 24
rsoe (see Outbound Enabler)
rsoe.exe
    syntax, 35
rsoe.exe
    start service, 43

S
SAP passport support
    Relay Server, 57
security
    Relay Server, 4
server farm
    Relay Server, 4
services
    running the State Manager as a service, 23
SQL Anywhere server
    Relay Server, supported, 31
state management
    Relay Server, 23
State Manager
    options, 24
    Relay Server command line syntax, 24
status page
    refreshing, 7
    Relay Server, 7
Sybase hosted relay service
    about, 67
    adding a server farm, 67
    logging in, 67
    subscribing, 67
Sybase Unwired Platform server
    Relay Server, supported, 31
synchronizing through a web server
    Relay Server, 1
syntax
    Outbound Enabler, 35
    Relay Server Outbound Enabler, 35
    Relay Server State Manager command line, 24

U
Unwired server
    Relay Server, supported, 31
url
    Relay Server Status Page, 7

V
verbosity
    Relay Server, 57
    Relay Server configuration file property - backend farm, 30
virtual memory
    Relay Server, 30

W
web extensions
    Relay Server, 1