PUBLIC

SQL Anywhere - Relay Server

Document Version: 17.01.0 – 2021-10-15

Relay Server

THE BEST RUN

Content

1 Relay Server

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

In this section:

1.1 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 SAP Afaria, SAP Mobile Office, MobiLink, SAP SQL Anywhere, SQL Remote, SAP Mobile Server, and SAP Mobile 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.
- Communication between mobile devices and backend servers in a way that integrates easily with existing corporate firewall configurations and policies.

In this section:

1.1.1 Relay Server Architecture

A typical Relay Server deployment is comprised of mobile devices, one or more Relay Servers, and one or more backend servers.

A Relay Server deployment consists of the following components:

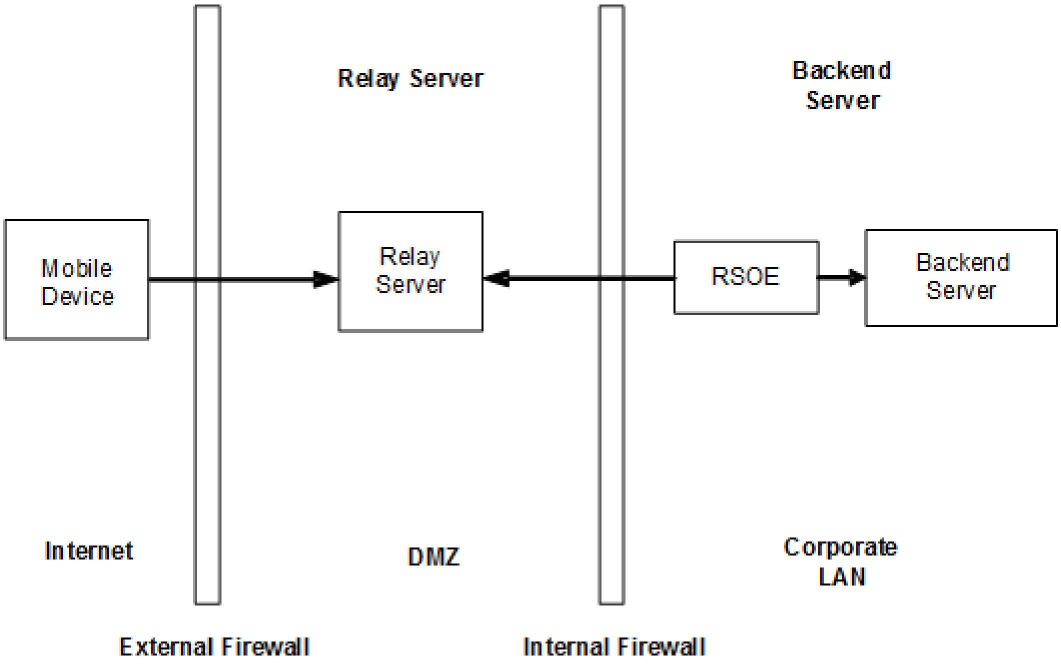
- Mobile devices running client applications and services that communicate with backend servers running in a corporate LAN.
- An optional load balancer to direct requests from the mobile devices to a group of Relay Servers.
- One or more Relay Servers running in the corporate DMZ.
- At least one backend server running in a corporate LAN that services client requests. The following backend servers are supported for use with the Relay Server:
  - SAP Afaria
  - SAP Mobile Office
  - MobiLink
  - SAP Mobile Platform
  - SAP SQL Anywhere
  - SQL Remote

    - SAP Mobile Server

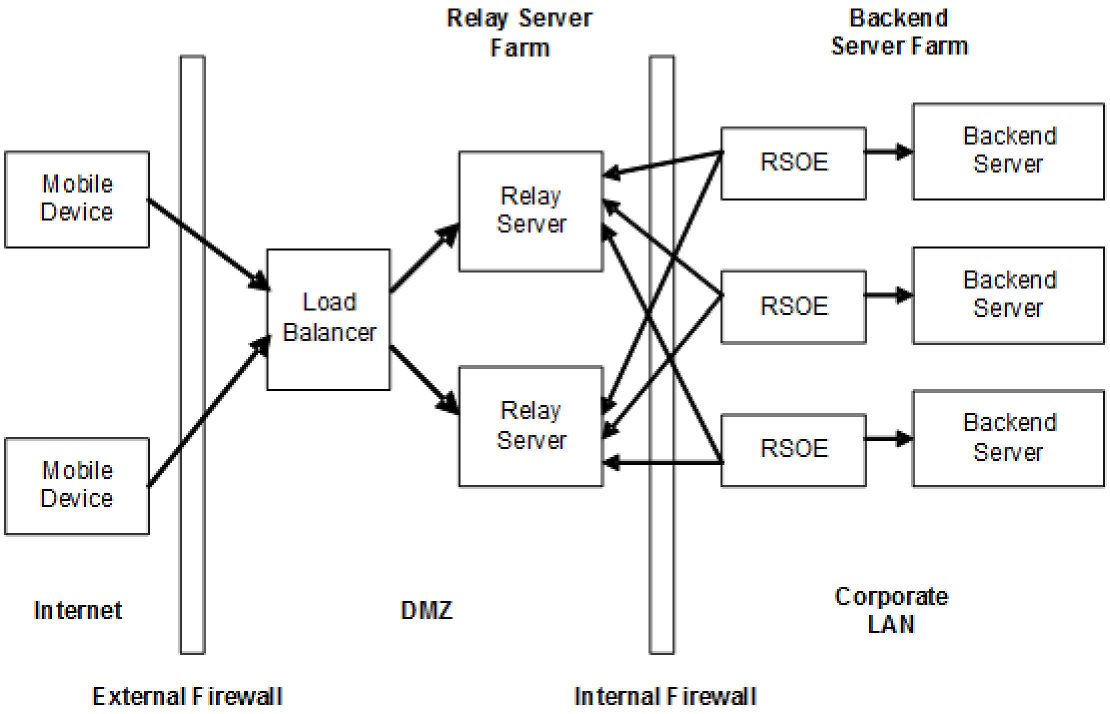
> **i Note**
>
> Relay Server is tested with specific backend servers and clients that communicate using well-defined HTTP requests and responses. Deployments that use custom HTTP traffic, including using SQL Anywhere as a web server, must thoroughly test the traffic to ensure that the deployment works with the Relay Server.
>
> Refer to your license agreement or the SAP SQL Anywhere Supported Platforms and Engineering Support Status page for information about which backend servers are supported.

- There is usually only one Relay Server Outbound Enabler (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.



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 web server, three web extensions (an administration extension, a client extension, and a server extension), and a background process for maintaining state information (Apache only). 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.

Because the Relay Server is a web extension running in a web server, all communication is performed using HTTP or HTTPS. Using HTTP provides straightforward integration 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 configuration provides a more secure deployment environment because it does not require inbound connections from the DMZ into the corporate LAN.

Shared Memory and Security (Apache Only)

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, and 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 that support 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 for deployment to the DMZ by using the standard techniques documented for each supported web server and operating system. This hardening should include steps to reduce the number of operating system accounts on the web server computer. The hardening also restricts the computer/VM to run only the Relay Server and the web server. 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.

In this section:

Related Information

Supported Platforms

1.1.1.1 The Relay Server Farm

A Relay Server farm consists of one to any number of Relay Servers.

In the case of multiple Relay Servers, using a front-end load balancer is typical. You can also use other load-balancing solutions like round robin.

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 (RSOE) runs on the same computer as the backend server. Its initiates an outbound connection to all Relay Servers in the Relay Server farm on behalf of the backend server. There is usually only one RSOE per backend server.

Related Information

The Outbound Enabler [page 33]

1.1.1.2 Relay Server Security

The Relay Server has built-in security features, but also relies on security features provided by the web server to support secure communications.

The Relay Server, in combination with the web server, supports transport layer security for server, client, and RSOE authentication, HTTP authentication, RSOE proxy authentication, RSOE MAC address filtering, RSOE token authentication, and client encryption technologies (protocol-level encryption).

Server-side Certificates and Transport Layer Security

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

Client-side Certificates and Transport Layer Security

Using a client-side certificate, the web server can verify that a client or RSOE communicating with the Relay Server is trusted. 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, then a key exchange occurs to establish the encrypted connection.

Backend Server and Farm Configuration

The Relay Server uses the `rs.config` file to define the peer list of Relay Servers when it runs in a farm environment, including the backend farm and backend server configurations. Each Relay Server in the Relay Server farm needs an identical copy of the `rs.config` file.

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

HTTP Authentication

The Relay Server fully inherits HTTP authentication options that are offered by the web server. For example, basic, digest, Windows 8, and client certificate mapping can be used between the client or RSOE and the web server.

RSOE MAC Address Filtering and Token Authentication

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 that 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.

RSOE Proxy Authentication

The RSOE supports HTTP proxy authentication use a user ID and password.

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. To ensure completely secure communication through the WAP Gap, use the MobiLink end-to-end encryption feature to protect data as it passes through the Relay Server. The MobiLink end-to-end encryption feature provides protocol-level encryption between SQL Anywhere and UltraLite clients and the MobiLink server via RSA. End-to-end-encryption can be use with or without transport layer security..

HTTP Listening Port of the Backend Server

For security purposes, when using the Outbound Enabler, explicitly bind the HTTP listening port of the backend server to the loopback IP address (127.0.0.1) only to minimize the surface area of attacks on the backend server.

Related Information

The Relay Server Configuration File [page 20]
End-to-end Encryption
Configuring the Relay Server with Microsoft IIS Using SSL
Configuring the Relay Server with Apache using SSL

1.1.2 Affinity

On a network using the Relay Server, *affinity* refers to the association between a client and a backend server across multiple HTTP requests. Affinity is only required when a client requires multiple requests to go to the same backend server.

The Relay Server adds affinity information to HTTP responses and respects affinity information that is sent up in HTTP requests. The affinity information is sent via HTTP cookies. Clients that must issue multiple requests to the same backend server via the Relay Server must send back the affinity information received in each HTTP response by injecting it into the next HTTP request. Backend 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 backend server, each request creates a new TCP socket connection. When there are multiple backend 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 backend server.

If your application is not RESTful and therefore has an affinity requirement, then the client requires support for standard HTTP cookie reflection. The Relay Server uses a non-persistent HTTP cookie to maintain affinity.

Related Information

Configuring a Backend Farm (Command Line) [page 26]

1.1.3 Relay Server Status Page

The Relay Server status page provides information about services, hosts, and the availability of backend farms.

Relay Server status includes:

- Host name
- The description specified in `rs.config`
- 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

Both the Detailed status page and Backend server status pages provide backend server status information.

> ℹ Note
>
> An empty status page may indicate that there is a configuration issue preventing the extension from generating the page.

Set the frequency of the backend server status page auto refresh by using the ias-rs-status-refresh-sec = n parameter. If n is zero, then auto refresh is turned off. To have the status page refresh every minute, use the following URL with a Microsoft IIS Relay Server:

```
http://host/rs/client/rs.dll?ias-rs-status-refresh-sec=60&ias-rs-
farm=SimpleTestApp-farm
```

The URL for a typical Microsoft IIS status page is http://`host`/rs/server/rs.dll.

The URL for a typical Apache status page is http://`host`/srv/iarelayserver.

Additional Status Pages

The following status pages are also available:

Status page	User or location	URL format examples
Detailed status	Relay Server administrator	IIS: http://`host`/rs/admin/rs.dll Apache: http://`host`/admin/iarelayserver
Overall status	Remote user	IIS: http://`host`/rs/client/rs.dll Apache: http://`host`/cli/iarelayserver

Status page	User or location	URL format examples
Backend farm status	Remote user	IIS: http://`host`/rs/client/rs.dll?ias-rs-farm=`App-farm` Apache: http://`host`/cli/iarelayserver?ias-rs-farm=`App-farm`
Backend server status	Remote user	IIS: http://`host`/rs/client/rs.dll?ias-rs-farm=`App-farm`&ias-rs-server=`App-server` Apache: http://`host`/cli/iarelayserver?ias-rs-farm=`App-farm`&ias-rs-server=`App-server`
Overall status	Backend server administrator	IIS: http://`host`/rs/server/rs.dll Apache: http://`host`/srv/iarelayserver
Backend farm status	Backend server administrator	IIS: http://`host`/rs/server/rs.dll?ias-rs-farm=`App-farm` Apache: http://`host`/srv/iarelay-server?ias-rs-farm=`App-farm`
Backend server status	Backend server administrator	IIS: http://`host`/rs/server/rs.dll?ias-rs-farm=`App-farm`&ias-rs-server=`App-server` Apache: http://`host`/srv/iarelay-server?ias-rs-farm=`App-farm`&ias-rs-server=`App-server`

1.2 Relay Server Deployment

Relay Server can be deployed on IIS 7.0, 7.5, 8.0, or 8.5, and Apache on Linux.

1.3 Deploying the Relay Server Components to Microsoft Windows Server (Command Line)

Configure and deploy Relay Server files to each computer in the Relay Server farm.

Prerequisites

Choose a deployment platform:

- Windows Server 2008 (IIS 7.0 without WebSocket support for Agentry traffic)
- Windows Server 2008 R2 (IIS 7.5 without WebSocket support for Agentry traffic)
- Windows Server 2012 (IIS 8.0 with WebSocket support for Agentry traffic)
- Windows Server 2012 R2 (IIS 8.5 with WebSocket support for Agentry traffic)

The Relay Server components are installed using the SQL Anywhere install. By default, all files are installed to *%SQLANY17%*:

- *%SQLANY17%*`\Bin64` contains supporting DLLs and executables for administration.
- *%SQLANY17%*`\RelayServer\IIS\Bin64` contains `rs.dll`, the `rs.config` configuration file, and the log folder.
- *%SQLANY17%*`\java\rstool.jar` contains the administration tool for the Relay Server.

Context

The *%SQLANY17%*`\RelayServer\IIS\iis7_plus_setup.bat` setup script performs the following tasks:

1. Installs Microsoft IIS 7 or Microsoft IIS 8 and turns on the required features.
2. Backs up your Microsoft IIS configuration.
3. Configures Microsoft IIS 7 or Microsoft IIS 8 for the Relay Server.

Web server administrators can customize the script as necessary.

The Relay Server for Windows consists of the following executables:

- `rs.dll`
- `dblgen`*17*`.dll`
- `dbfhide.exe`

- `dbicu`*17*`.dll`
- `dbicudt`*17*`.dll`
- `dbsupport.exe`
- `dbghelp.dll`

Procedure

1. Run `iis7_plus_setup.bat`.
2. Update the Relay Server configuration for Microsoft IIS on Windows. For each computer that belongs to the Relay Server farm you are updating, copy the updated configuration file to the *%SQLANY17%*`\RelayServer\IIS\Bin64\Server` directory under the Relay Server web site home directory.

Results

The Relay Server configuration file is deployed to the specified Relay Server(s).

Related Information

The Relay Server Configuration File [page 20]
Supported Platforms
File Hiding Utility (dbfhide)

1.4 Deploying the Relay Server Components to Apache on Linux (Command Line)

Configure and deploy Relay Server files to each computer in the Relay Server farm before running the Relay Server with Apache.

Prerequisites

The SQL Anywhere install includes Relay Server components. Relay Server files are installed to the */opt/sqlanywhere17* directory.

Context

> **i Note**
>
> The interactive quick setup feature configures the Apache web server for Relay Server and is an alternative to this procedure. Run the `ap-setup.sh` script in the `/relayserver/quicksetup_apache` directory.

Relay Server for Apache consists of the following executables:

- `mod_rs_ap_client.so`
- `mod_rs_ap_server.so`
- `rshost`
- `dblgen`*17*`.res`
- `libdbtasks`*17*`.so`
- `libdbtasks`*17*`_r.so`
- `libdbicudt`*17*`.so`
- `libdbicu`*17*`_r.so`
- `dbsupport`
- `dbfhide`
- `mod_rs_ap_admin.so`

Procedure

1. Create the Relay Server configuration file `rs.config`.
2. Copy `rs.config` into the `install-dir /relayserver/apache??/bin64` directory. `??` is 22 for Apache 2.2.x and 24 for Apache 2.4.x.
3. Edit the Relay Server configuration file `rs.config` using the following guidelines.
   - The file has four sections, and each section starts with a section tag that is enclosed in square brackets:
     - Relay server section
     - Backend farm section
     - Backend server section
     - Options section
   - Add the appropriate properties to each section. Define each property as a keyword=value pair.
   - The configuration file only supports 7-bit ASCII characters.
4. Add the following directories to the LD_LIBRARY_PATH environment variable:
   - `install-dir/lib64`
   - `install-dir/relayserver/apache??/bin64`

   Edit the `/apache-dir/bin/envvars` file to set and then export LD_LIBRARY_PATH.
5. Source `/apache-dir/bin/envvars` before running client- or server-facing Apache instances.
6. Configure the server-facing Apache instance:

a. Copy the Apache `conf/httpd.conf` file and name the copy `httpd.conf.server`.

b. Add the following line to `conf/httpd.conf` to load the Relay Server server module:

```
LoadModule iarelayserver_server_module install-dir/relayserver/apache??/
bin64/mod_rs_ap_server.so
```

> **i Note**
>
> All modules are invoked using different URLs and all modules explicitly look for the string `iarelayserver` in the URL path. That part of the URL need not change.

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

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

> **i Note**
>
> The Remote Administration module configuration is optional. It can reside in the server- or client-facing configuration file.

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

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

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

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

f. Update the Listen directive and ensure the server instance listens on a different port or adapter than the client instance. This port must also be accessible to the computer running the Outbound Enabler.

g. Update the ErrorLog directive and change the name of the error log file so that the server-facing instance logs to a separate file.

h. Update the CustomLog directive and change the name of the access log file so that the server-facing instance logs to a separate file.

i. Add a PidFile directive so that the client pid file and server pid file cannot overwrite each other. For example:

```
PidFile "logs/httpd_server.pid"
```

j. If Apache is set up for HTTPS, then make another copy of `conf/extra/httpd-ssl.conf` (for example `httpd-ssl.conf.server`) and change the SSL listening port. If both Apache instances include the same file, then a socket bind error occurs on the Apache instance that is started second.

k. Set the Timeout directive to 60 seconds. The Timeout value must be greater than both the max_junction_idle_sec value and the value expected in the application's timeout logic.

l. Disable the Apache reqtimeout_module.

7. Configure the client-facing Apache instance.

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

    b. Add the following line to load the Relay Server client module:

    ```
    LoadModule iarelayserver_client_module install-dir/relayserver/apache??/
    bin64/mod_rs_ap_client.so
    ```

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

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

    d. Set the Timeout directive to 60 seconds. The Timeout value must be greater than the value expected in the application's timeout logic.

8. On Linux, if the $TMP, $TMPDIR, or $TEMP environment variable is set globally when Apache spawns a process, then Apache configuration is complete.

   If any of the above environment variables are not set globally, or if you want to place the default Relay Server log file in a specific temporary directory, then edit the file `/apache-dir/bin/envvars` to set and then export $TMP.

   If you start Apache by using the httpd command line directly, then source `/apache-dir/bin/envvars` first.

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

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

   > **i Note**
   >
   > The Apache user process requires write permissions to the specified `tmp` directory.

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

9. To update the Relay Server configuration while it is started:

    a. Copy the updated configuration file to `install-dir/relayserver/apache??/bin64`.

    b. From the `install-dir/relayserver/apache??/bin64` directory, run the following command line to apply the configuration update:

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

    c. If the Relay Server is set up as a farm with more than one server, then 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.

Related Information

The Relay Server Configuration File [page 20]

Relay Server State Manager (Linux) [page 18]
Relay Server State Manager Service (Linux) [page 18]
Relay Server State Manager (rshost) Command-line Syntax (Linux) [page 19]
File Hiding Utility (dbfhide)

1.5 Relay Server State Manager (Linux)

The Relay Server State Manager is a process that maintains Relay Server state information across client requests and Outbound Enabler sessions. The State Manager also manages the Relay Server log file.

The State Manager runs as a service.

The default Relay Server log file name is `ias_relay_server_host.log`. The file is located in the directory specified by the TMP, TEMP, or TMPDIR environment variables. If none of those variables are set, then the Relay Server creates a log file in the `/tmp` directory.

> i 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 *yymmddnn*`.olg`, where *yymmdd* represents the date on which the log file was renamed and *nn* is the sequential version number of the log file for that day.

In this section:

Relay Server State Manager Service (Linux) [page 18]
The SQL AnywhereService utility for Linux (dbsvc) creates, modifies, and deletes services. Use it to set up the Relay server State Manager as a service that starts at boot time.

Relay Server State Manager (rshost) Command-line Syntax (Linux) [page 19]
The rshost command configures the Relay Server State Manager on Linux. This functionality is not available for Windows.

1.5.1 Relay Server State Manager Service (Linux)

The SQL AnywhereService utility for Linux (dbsvc) creates, modifies, and deletes services. Use it to set up the Relay server State Manager as a service that starts at boot time.

For complete usage information, run `dbsvc` without any options.

> i Note
>
> Use absolute paths when you create an rshost service.

Remarks

Use the same user account so that Apache user processes can attach to the State Manager shared memory and be able to read it and write to it.

Example

Set up an auto-started State Manager service named RelayServer	`dbsvc -y -a apache-user -t rshost -w RelayServer -q -qc -f /your-directory/rs.config -os 100K -ot /tmp/rs.log`
Start the service	`dbsvc -u RelayServer`
Stop the service	`dbsvc -x RelayServer`
Uninstall the service	`dbsvc -d RelayServer`

Related Information

Configuring a Relay Server (Command Line) [page 23]

Service Utility (dbsvc) for Windows

1.5.2 Relay Server State Manager (rshost) Command-line Syntax (Linux)

The rshost command configures the Relay Server State Manager on Linux. This functionality is not available for Windows.

```
rshost [ option ]+
```

Parameters

Options

The following options configure the State Manager. They are all optional.

rshost options	Description
-f filename	Indicates the name of the Relay Server configuration file.
-o filename	Indicates the name of the file to use for logging.
-os size	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 <yymmdd><nn>.olg 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.
-oq	Prevents a popup window when there is a startup error.
-q	Runs in a minimized window.
-qc	Closes the window on completion.
-u	Updates the configuration of a running Relay Server.
-ua	Archives the log file to <yymmdd><nn>.log and truncates the file.

1.6 The Relay Server Configuration File

A Relay Server configuration file defines properties for a Relay Server farm and the backend server farms that are made available by the Relay Server farm.

Each section in the Relay Server configuration file starts with a section tag. A section tag is formed by enclosing a keyword that identifies the section name in square brackets. The file is divided into the following sections: [options], [relay_server], [backend_farm], and [backend_server]. Specify these sections in any order.

Several lines follow the section tag, and these lines define properties for that section. Define a property by specifying `property name`= `value`. All section and property names are case insensitive. Use a number sign (#) at the beginning of a line to identify a comment. The configuration file supports only 7-bit ASCII characters.

The File Hiding utility (dbfhide) uses encryption to hide the contents of configuration files and initialization files. The Relay Server and Outbound Enabler automatically detect that a configuration file has been obfuscated with dbfhide.

Configuration File Monitoring

On Microsoft IIS, the Relay Server continuously monitors the configuration file and reports the following status changes in the log file (there is a two second latency in the reporting):

- A fatal error if the configuration file becomes invalid. The Relay Server cannot restart with an invalid configuration file.
- A warning if the configuration file is valid, but does not match the copy being used.
- An informational message when the error or warning is resolved.

Example Relay Server Configuration File

The following Relay Server configuration file defines a Relay Server farm with two Relay Servers, four backend farms, and five backend servers.

```
# Relay Servers:
     rs1.rs.com
     rs2.rs.com
#
# Assume the load balanced address of the Relay Server farm is www.rs.com.
#
# Backend farms and servers and their corresponding rsoe2 command lines:
     Company1.Sales.MLFarm
         MLServer01   rsoe2.exe -cr host=www.rs.com -f Company1.Sales.MLFarm -id 
MLServer01 -t 7b2493b0-d0d4-464f-b0de-24643e1e0feb
         MLServer02   rsoe2.exe -cr host=www.rs.com -f Company1.Sales.MLFarm -id 
MLServer02 -t de1aac83-a653-4e0f-8a6c-0a161a6ee407
     Company1.Manufacturing.MLFarm
         MLServer01   rsoe2.exe -cr host=www.rs.com -f 
Company1.Manufacturing.MLFarm -id MLServer01 -t 621ece03-9246-4da7-99e3-
c07c7599031c
     Company2.AFFarm
         AFServer01   rsoe2.exe -cr host=www.rs.com -f Company2.AFFarm -id 
AFServer01 -t a688728f-1ae7-4438-969e-6f5e30a07882
     Company2.SAPFarm
         SAPG1        rsoe2.exe -cr host=www.rs.com -f Company2.SAPFarm -id 
SAPG1 -t de68b75b-ff33-4c81-a920-2333494dfd8a -cs 
"host=localhost;port=443;https=1;identity=c:
\rsoe.id;identity_password=****;trusted_certificates=c:\testrsaserver.crt"
# Note: The two instances of MobiLinkServer01 are different servers.
#
# The Relay Server configuration file contains authentication information.
# Access to this file must be administered.
#
# URL prefix for client applications:
#     https://www.rs.com/rs17/client/rs.dll/Company1.Sales.MLFarm
#     https://www.rs.com/rs17/client/rs.dll/Company1.Manufacturing.MLFarm
#     https://www.rs.com/rs17/client/rs.dll/Company2.AFFarm
#     https://www.rs.com/rs17/client/rs.dll/Company2.SAPFarm
#
#-------------------------------------
# Relay Server options
#-------------------------------------
[options]
verbosity = 1
#--------------------
# Relay Server peers
#--------------------
[relay_server]
```

```
enable           = yes
host             = rs1.rs.com
http_port        = 80
https_port       = 443
description      = Machine #1 in RS farm
[relay_server]
enable           = yes
host             = rs2.rs.com
http_port        = 80
https_port       = 443
description      = Machine #2 in RS farm
#----------------
# Backend farms
#----------------
[backend_farm]
id               = Company1.Sales.MLFarm
[backend_farm]
enable           = yes
verbosity        = 4
id               = Company1.Manufacturing.MLFarm
description      = Company1's MobiLink farm in their Manufacturing department
[backend_farm]
enable           = yes
id               = Company2.AFFarm
description      = Company2's Afaria farm
[backend_farm]
enable           = yes
id               = Company2.SAPFarm
forward_x509_identity= yes
forwarder_certificate_subject= CN = \*.mysap.com, *
forwarder_certificate_issuer= CN = ca??.mysap.com, *
description      = Company2's SAP Gateway farm
#------------------
# Backend servers
#------------------
[backend_server]
farm     = Company1.Sales.MLFarm
id       = MLServer01
[backend_server]
farm     = Company1.Sales.MLFarm
id       = MLServer02
[backend_server]
enable   = no
farm     = Company1.Manufacturing.MLFarm
verbosity= inherit
mac      = 01-23-45-67-89-ad
id       = MLServer01
token    = 621ece03-9246-4da7-99e3-c07c7599031c
description= ML server number 1
[backend_server]
enable   = yes
farm     = Company2.AFFarm
id       = AFServer01
mac      = 01-23-45-67-89-ae
token    = a688728f-1ae7-4438-969e-6f5e30a07882
[backend_server]
enable   = yes
farm     = Company2.SAPFarm
id       = SAPG1
mac      = 01-23-45-67-89-af
token    = de68b75b-ff33-4c81-a920-2333494dfd8a
```

In this section:

Configuring a Backend Farm (Command Line) [page 26]
  Define properties for a backend server farm in the [backend_farm] section of the Relay Server configuration file.

Configuring a Backend Server (Command Line) [page 30]
  Define properties for a backend server connection that the Outbound Enabler uses when it connects to the Relay Server farm on behalf of a backend server.

Configuring the Relay Server Automatically (Command Line) [page 32]
  Use the auto_config option to set default property values for the backend server and backend farm.

Related Information

File Hiding Utility (dbfhide)

1.6.1 Configuring a Relay Server (Command Line)

Define properties for a Relay Server in the Relay Server configuration file.

Procedure

1. Create a `rs.config` file.
2. Use the following properties in the [relay_server] section of the file to configure the Relay Server:

Property	Description
enable	(Optional) Specifies whether this Relay Server is included in the Relay Server farm. **yes** (Default) 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.
host	Specifies the host name or IP address that the Outbound Enabler uses to make a direct connection to the Relay Server.

Property	Description
http_port	Specifies the HTTP port that the Outbound Enabler uses 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 the Outbound Enabler. **1 to 65535** Enable HTTP at the specified port.
https_port	Specifies the HTTPS port that the Outbound Enabler uses 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	(Optional) Specifies a custom description of up to 2048 characters.

3. Use the following properties in the [options] section of the file to configure the Relay Server. The file contains only one [options] section, and the settings apply to all Relay Servers that are defined in the file.

Property	Description
auto_config	Configures the Relay Server so that Outbound Enablers can connect by using the token of unseen backend farms and backend servers without being configured in advance. New RSOEs can connect if one of the following conditions is true: 1. They are the first to connect with a previously unknown farm name. 2. They connect with a known farm name and the same token as the first RSOE to connect with that farm name. The association between farm names and tokens is lost/reset when you restart the Relay Server.
log_size_limit (Microsoft Windows only)	Sets the maximum size of the Relay Server log file. The supported units are: k, K, m, M, g, and G. The default value is 0 (no limit).
min_thread (Microsoft Windows only)	Specifies the minimum number of threads to allocate to the thread pool. The default setting is *auto*: threads are not returned to the idle pool if the size of the idle pool is equal to or greater than half of *max_thread*.

Property	Description
max_thread (Microsoft Windows only)	Specifies the maximum number of threads to allocate to the thread pool. The default setting is *auto*: the Relay Server does not limit the number of active threads in the pool. If requests arrive when there are no threads left in the idle pool, then the Relay Server allocates new threads up to the thread limit of 65,535. *max_thread* must be greater than *min_thread*. To exceed 5000 concurrent requests for Microsoft IIS 7 or later, specify the following configuration all on one line, where new_limit must be greater than 3x5000: `%systemroot%\system32\inetsrv\appcmd.exe set config "Default Web Site" -section:system.webServer/serverRuntime/appConcurrentRequestLimit:"new-limit"/commit:apphost` For *max_thread* to work as a soft limit, set `new-limit` higher than *max_thread*.
shared_mem (Linux only)	(Optional) Specifies the maximum amount of shared memory that the Relay Server uses for state tracking. The default value is 10 MB. The maximum setting is 4 GB. Consider changing this setting when any 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

Property	Description
verbosity	Specifies the verbosity level: **0** (Default) Log errors only. Use this logging level for deployment. **1** Request logging. All HTTP requests are written to the Relay Server 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.

4. Save the `rs.config` file.

Results

The Relay Server is configured.

1.6.2 Configuring a Backend Farm (Command Line)

Define properties for a backend server farm in the [backend_farm] section of the Relay Server configuration file.

Context

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. Each section is identified by the backend_farm keyword.

Procedure

1. Create a `rs.config` file.
2. Use the following properties in the [backend_farm] section of the file to configure the backend farm:

Property	Description
description	(Optional) Specifies a custom description of up to 2048 characters.
enable	(Optional) Specifies whether to allow connections from this backend server farm. **yes** (Default) Allow connections from this backend server farm. **no** Disallow connections from this backend server farm.
forward_x509_identity	Controls how the SAP NetWeaver Gateway authenticates clients. One way is through X.509 certificate forwarding through trusted intermediaries. When this property is set to yes, the Relay Server can extract forwarded client identity information from a trusted forwarder and forward it to the SAP NetWeaver Gateway or Web Dispatcher by using HTTP headers. The default setting is no.

forwarder_certificate_issue	Controls how client identity headers are treated. When 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 is to replace the existing headers with the identity of the forwarder. To grant permission for a forwarder to forward other client identities, set *forwarder_certificate_issuer*=`match-string` and *forwarder_certificate_subject*=`match-string`, where `match-string` is checked against a serialized form of the corresponding compound name field in the certificate. Use ? to match any character and * to match any string. Use \ as the leading escape character for ?, *, or \ to match them literally. For example:
	`forwarder_certificate_issuer = 'CN =`
	`quicksigner, OU = security`
	`department, O = my org, L = my city,`
	`S = my state, C = my country'`

Property	Description
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 is to replace the existing headers with the identity of the forwarder. To grant permission for a forwarder to forward other client identities, set *forwarder_certificate_issuer*=`match-string` and *forwarder_certificate_subject*=`match-string`, where `match-string` is checked against a serialized form of the corresponding compound name field in the certificate. Use ? to match any character and * to match any string. Use \ as the leading escape character for ?, *, or \ to match them literally. For example: `forwarder_certificate_subject = 'CN = mySapWD??.my.com, OU = SEC, O = SAP, *'`
id	Specifies the name up to 2048 characters for the backend server farm.
inject_affinity_query_header	(Optional) Provides backward compatibility with the Afaria Open Mobile Alliance Device Management (OM ADM) backend server. **yes** Provides backward compatibility. **no** (Default) Does not provide backward compatibility. If no value is specified, then you can connect by using HTTP or HTTPS.
token	(Optional) A security token of up to 2048 characters that the Relay Server uses to authenticate the backend server connection.

Property	Description
verbosity	Sets the verbosity for the backend farm. **0** (Default) Log errors only. Use this logging level for deployment. **1** Request logging. All HTTP requests are written to the Relay Server 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.

3. Save the `rs.config` file.

Results

The backend farm is configured.

Related Information

Affinity [page 10]

1.6.3 Configuring a Backend Server (Command Line)

Define properties for a backend server connection that the Outbound Enabler uses when it connects to the Relay Server farm on behalf of a backend server.

Context

There is a backend server section for each Outbound Enabler that connects to the Relay Server farm. Each backend server section is identified by the backend_server keyword. The backend server section also assigns a backend server to a backend server farm.

The Relay Server supports the following backend servers:

- MobiLink
- SAP Afaria
- SAP Mobile Office
- SAP Mobile Platform
- SAP Mobile Server
- SAP SQL Anywhere

> i Note
>
> Refer to your license agreement or the SAP SQL Anywhere Supported Platforms and Engineering Support Status page for information about which backend servers are supported.

Procedure

1. Create a `rs.config` file.
2. Use the following properties in the [backend_server] section of the file to configure the backend server:

Property	Description
description	(Optional) Specifies a custom description of up to 2048 characters.
enable	(Optional) Specifies whether to allow connections from this backend server. **yes** (Default) Allows connections from this backend server. **no** Disallows connections from this backend server.

Property	Description
farm	Specifies the name of the backend server farm that this backend server belongs to.
id	Specifies the name for that the backend server connection up to 2048 characters.
MAC	(Optional) Specifies the MAC address of the network adapter that the Outbound Enabler uses to communicate with the Relay Server. Use the IEEE 802 MAC-48 format to specify the address. To determine the correct format for the MAC address, look in the Relay Server Outbound Enabler console or log. If the property is not specified, then the MAC address is not checked.
max_junction	Sets the maximum number of active and idle junctions that are available to the Relay Server. This setting overrides the -jl option for rsoe2. The default value for this option is 1000.
max_junction_idle_sec	Controls how long junctions can remain inactive before the Outbound Enabler discards them. The Outbound Enabler automatically replenishes the idle junction pool if its size falls below the spare junction low watermark that the -jsl option controls. The default value is 30.
token	(Optional) A security token of up to 2048 characters that the Relay Server uses to authenticate the backend server connection.
verbosity	Specifies the verbosity level. **0** Log errors only. Use this logging level for deployment. This is the default. **1** Request logging. All HTTP requests are written to the Relay Server 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.

3. Save the `rs.config` file.

Results

The backend server is configured.

Related Information

Supported Platforms

1.6.4 Configuring the Relay Server Automatically (Command Line)

Use the auto_config option to set default property values for the backend server and backend farm.

Context

When the auto_config option is set to on, the Relay Server runs as a trust on first use (TOFU) system. Automatic configuration is useful for testing purposes and for administration-free environments.

Outbound Enablers connect with unseen backend farms and backend servers. Outbound Enablers that belong to the same backend farm connect by using a farm-wide token.

The automatically configured farm settings persist when the Relay Server is restarted. The backend server configuration also persists per Outbound Enabler, with a unique server token for the backend farm.

When you use automatic configuration, you can still change backend farm and backend server settings. To make changes, use rshost locally for Apache or AdminChannel for IIS or Apache.

Procedure

1. Reserve the farm name before the Relay Server starts by specifying it with the token property in the backend_farm section of the Relay Server configuration file.

   All Outbound Enablers that belong to the same automatically configured farm must supply a token that matches the farm-wide token; otherwise, access is denied.
2. Set the auto_config option to on.

Results

When the Relay Server processes the first Outbound Enabler connection with a previously unseen farm name, it creates a new backend farm configuration. The Relay Server updates the original configuration file and stores the supplied token in a new backend farm property. Other backend farm properties are initialized to default values.

1.7 The Outbound Enabler

The Outbound Enabler opens outbound connections from computers running in the corporate LAN to the Relay Server farm running in the DMZ, and forwards client requests received from the Relay Server to the backend server and backend server responses to the client via the Relay Server.

The Outbound Enabler runs on the same computer as the backend server. When the Outbound Enabler starts, it makes an HTTP request to retrieve the list of Relay Servers running in the farm by using a direct or load-balanced URL. The Relay Server receiving the request from the Outbound Enabler returns the connection information for all Relay Servers in the farm by using a URL that maps to the server extension component of the Relay Server. The URL maps to a server extension of the Relay Server directly or indirectly via various proxy, load balancing, or high availability options in a Relay Server farm environment. The URL is stored in the Relay Server configuration file on each Relay Server.

The Outbound Enabler produces junctions, which consist of a logical connection to the Relay Server and a connection to the backend server. A junction represents the one-to-one association of those connections. Junctions transport client requests and the corresponding server responses between the Relay Server and the backend server. The Relay Server and the Outbound Enabler may also use half of a junction to notify each other. The Outbound Enabler maintains a pool of junctions to reduce latency when relay is required. When a request is made, a junction is assigned to the HTTP request for the life of the request. When the request ends, the junction is re-paired with a new backend connection and then returned to the pool for future use. Junctions that are not used within a specified time are destroyed and then recreated. There is a cost for maintaining the junction pool. The rsoe2 utility has options that control the pool size, and the Relay Server configuration file can also limit the maximum number of junctions that are allowed for each backend server.

In this section:

Related Information

Supported Platforms

1.7.1 Relay Server Outbound Enabler Syntax

The rsoe2 command runs the Relay Server Outbound Enabler (RSOE).

```
rsoe2 [ option ]+
```

```
rsoe2 @{ filename | environment-variable } ...
```

Parameters

Options

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, then it must also be specified (-t).

rsoe2 options	Description
`@data`	Reads options from the specified environment variable or configuration file. To protect information in the configuration file, use the File Hiding utility (dbfhide) to encode the contents of the configuration file.

rsoe2 options	Description
`-cr "connection-string"`	Specifies the Relay Server connection string. The format of the Relay Server connection string is a semicolon-separated list of keyword=value pairs. **host** The IP address or hostname of the Relay Server. The default is localhost. **port** Required. The port that the Relay Server is listening on. **http_userid** Optional. The user ID for authentication. Consult your web server (or proxy) documentation to determine how to set up HTTP authentication. **http_password** Optional. The password for authentication. Consult your web server (or proxy) documentation to determine how to set up HTTP authentication. **http_proxy_userid** Optional. The user ID for proxy authentication. Consult your web server (or proxy) documentation to determine how to set up HTTP authentication. **http_proxy_password** Optional. The password for proxy authentication. Consult your web server (or proxy) documentation to determine how to set up HTTP authentication. **proxy_host** Optional. The host name or IP address of the proxy server. i Note If you experience issues with proxy servers that are taking too long to buffer, then use HTTPS, which prevents proxy buffering. **proxy_port** Optional. The port number of the proxy server. **url_suffix** Required. The URL path to the server extension of the Relay Server. **https** 0 - HTTP (default) 1 - HTTPS

rsoe2 options	Description
	For *https=1*, the following options can also be specified. Specify at least one of certificate_name, certificate_company, or certificate_unit to ensure that the Outbound Enabler is connecting to the correct Relay Server. To prevent checking the certificate, specify the certificate_name_check option. **certificate_name** The common name field of the certificate. **certificate_company** The organization name field of the certificate. **certificate_unit** The organization unit field of the certificate. **identity** This option provides the credentials to establish mutually authenticated TLS between the Outbound Enabler and the Relay Server. Mutual authentication is required for the Relay Server. **identity_password** This option provides the credentials to establish mutually authenticated TLS between the Outbound Enabler and the Relay Server. Mutual authentication is required for the Relay Server. **fips** Choose whether to use FIPS-certified encryption implementations for TLS encryption and end-to-end encryption. **skip_certificate_name_check** Controls whether the client library skips the check of the server host name against the database server certificate host names. Set this boolean option to ON or OFF to control whether the host name of the Relay Server matches any of the host names in the root certificate. Enabling this option may prevent the client from fully authenticating the server, leaving it vulnerable to attack. When initiating TLS or HTTPS connections, the client libraries check the host name of the Relay Server against the certificate provided by that server using the procedure described in RFC 2818. This check only happens if none of the certificate_name, certificate_company, or certificate_unit options are specified, or if the skip_certificate_name_check option is not ena-

rsoe2 options	Description
	bled. If any of certificate_name, certificate_company, or certificate_unit are specified, then only those options are verified. The skip_certificate_name_check option disables the host name check when enabled. The host names or IP addresses are derived from the subjectAltName (Subject Alternative Name or SAN) extension and the Common Name (CN) field. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name. **trusted_certificates** This parameter takes the name of a file that contains a list of PEM-encoded X.509 trusted root certificates. To verify the Relay Server, and only the Relay Server, set this property to *backend_server_public_cert_filename*. `trusted_certificates=backend_server_public_cert_filename` On Microsoft Windows, if *trusted_certificates* is not set, then the operating system certificate store is used.

rsoe2 options	Description
`-cs "connection-string"`	Specifies the backend server connection string. The format of the connection string is a semicolon-separated list of name-value pairs. **host** The IP address or hostname of the backend server. The default is localhost. **port** Required. The port the backend server is listening on. The default is 0. **https** 0 - HTTP (default) 1 - HTTPS By default, MobiLink starts the TCP/IP communication protocol. When starting MobiLink for use with the RSOE, start the communication protocol that your RSOE configuration requires. 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. For *https=1*, you can specify the following options. Specify at least one of certificate_name, certificate_company, or certificate_unit to ensure that the Outbound Enabler is connecting to the correct backend server. To prevent checking the certificate, specify the skip_certificate_name_check option. **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. **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. **skip_certificate_name_check**

rsoe2 options	Description
	Controls whether the client library skips the check of the server host name against the database server certificate host names. Set this boolean option to ON or OFF to control whether the host name of the backend server matches any of the host names in the root certificate. Enabling this option may prevent the client from fully authenticating the server, leaving it vulnerable to attack. When initiating TLS or HTTPS connections, the client libraries will check the host name of the backend server against the certificate provided by that server using the procedure described in RFC 2818. This check only happens if none of the certificate_name, certificate_company, or certificate_unit options are specified, or if the skip_certificate_name_check option is not enabled. If any of certificate_name, certificate_company, or certificate_unit are specified, then only those options are verified. The skip_certificate_name_check option disables the host name check when enabled. The host names or IP addresses are derived from the subjectAltName (Subject Alternative Name or SAN) extension and the Common Name (CN) field. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name. **trusted_certificates** This parameter takes the name of a file that contains a list of PEM-encoded X.509 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` On Microsoft Windows, if *trusted_certificates* is not set, then the operating system certificate store is used.
-d `seconds`	Sets the frequency of the backend server liveness ping and backend server status request. The default is 5 seconds.

rsoe2 options	Description
-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.
-jsh `number`	Specifies the maximum number of junctions in the idle junction pool. The total number of junctions is allocated evenly across the number of Relay Servers in the farm. The default value is 200.
-jsl `number`	Specifies the minimum number of junctions in the idle junction pool. The total number of junctions is allocated evenly across the number of Relay Servers in the farm. The default value is 10.
-jl `number`	Specifies the maximum number of junctions (the sum of active and idle junctions). The total number of active and idle junctions is allocated evenly across the number of Relay Servers in the farm. This setting is overridden by the setting for max_junction in the Backend Server section of the Relay Server configuration file. The default value is 1000.
-o `file`	Specifies the file to log output messages to.
-oq	Prevents the appearance of the error window when a startup error occurs.
-os `size`	Sets the maximum size of the message log files. The minimum size limit is 10 KB.
-ot `file`	Truncates the specified log file and logs messages to it.
-q	Runs with a minimized window on startup.
-qc	Shuts down the window on completion.
-s	Stops the Outbound Enabler.
-t `token`	Sets the security token to be passed to the Relay Server.

rsoe2 options	Description
-uc	Starts rsoe2 in shell mode. This is the default. Applies to UNIX and Linux. Only specify one of -uc, -ui, -um, or -ux. When you specify -uc, the RSOE starts in the same manner as previous releases of the software.
-ud	Instructs rsoe2 to run as a daemon. Applies to UNIX and Linux platforms only.
-ui	Starts rsoe2 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 RSOE attempts to find a usable display. If it cannot find one, for example because the X window server isn't running, then rsoe2 starts in shell mode.
-ux	Opens the RSOE messages window where messages are displayed on Linux. On Microsoft Windows, the RSOE messages window appears automatically. 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.

rsoe2 options	Description
-v level	Sets the verbosity level to use for logging. The level can be 0, 1, 2, or higher (higher levels are used primarily for Technical Support): **0** Log errors only. Use this logging level for deployment. **1** Session level logging. This is a higher level view of a synchronization session. **2** Request logging. Provides a more detailed view of HTTP requests. **3 or higher** Detailed logging. Used primarily for Technical Support. Levels 1 and 2 are only written to the message log file and are not displayed. To have all log messages displayed, use the -dl option.

Remarks

Use the dbsvc utility to run the Outbound Enabler as a service. The syntax of dbsvc on Microsoft Windows is different than on UNIX and Linux. On UNIX or Linux, do not specify the full path of the executable as the first parameter after -w option argument.

On UNIX or Linux, specify the Outbound Enabler parameters in a configuration file only. Do not use command-line options in the command to set up the service.

Example

Set up an auto-started RSOE service named oes (Outbound Enabler service) on Microsoft Windows:

```
dbsvc -as -s auto -t rsoe2 -w oes "%SQLANY17%\Bin64\rsoe2.exe"
-cr "host=relayserver.sap.com;port=80" -cs "host=localhost;port=80" -f FarmName -
id ServerName -t token
```

Set up an auto-started RSOE service named oes (Outbound Enabler service) on UNIX/Linux:

```
dbsvc -y -a user-account -t rsoe2 -w oes @/full-dir-path/oe.config
```

1.7.2 Outbound Enabler Deployment Considerations

There are a number of considerations to be aware of when using the Outbound Enabler.

Outbound Enabler as a service

Use the Service utility (dbsvc) to set up and maintain the Outbound Enabler as a service.

Authentication

Use basic or digest authentication.

Related Information

Service Utility (dbsvc) for Windows
MobiLink Client Network Protocol Options
host MobiLink Client Network Protocol Option
port MobiLink Client Network Protocol Option
identity MobiLink Client Network Protocol Option
identity_password MobiLink Client Network Protocol Option
trusted_certificates MobiLink Client Network Protocol Option
-x mlsrv17 Option
File Hiding Utility (dbfhide)

1.8 Relay Server Farm Configuration Updates

The Relay Server farm configuration is defined in the Relay Server configuration file. All of the Relay Servers in a Relay Server farm share the same configuration file.

To update a Relay Server farm configuration, update the Relay Server configuration file at each Relay Server in the farm. Configuration updates include:

- 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.

Perform online updates of the Relay Server configuration by using the AdminChannel in `rstool.jar` or by using the Relay Server State Manager for the Linux Relay Server. Updates can be performed without restarting the Relay Server.

In this section:

Updating a Relay Server Configuration for Apache on Linux (Command Line) [page 45]
Update Relay Server configuration files to add or change Relay Servers or Relay Server farms and change server and farm properties and options.

Related Information

Relay Server State Manager (Linux) [page 18]

1.8.1 Updating a Relay Server Farm Configuration for Microsoft IIS on Microsoft Windows (Command Line)

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.
- The `Admin\rs.dll` URL is published. The setup script does this without hardening. Typical hardening includes adding an authentication requirement.

Procedure

1. Save updates in a new Relay Server configuration file.
2. Run the following command against each Relay Server to apply the configuration update:

```
java -cp %sqlany%\java\rstool.jar com.sap.relayserver.AdminChannel -url
https://rs.my.com/rs17/admin/rs.dll -uid me -pwd myPassword -setRSConfig new-
rs-config-file
```

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

Results

The Relay Server farm configuration is updated.

1.8.2 Updating a Relay Server Configuration for Apache on Linux (Command Line)

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.

Context

The Relay Server State Manager updates the configuration of a running Relay Server farm without interrupting service.

Procedure

1. Update the master copy of the Relay Server configuration file.
2. Copy the updated configuration file to the `/modules` directory under the Apache install directory.
3. From the `/Apache-install/modules` directory, run the following command to apply the configuration update:

   ```
   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.
4. Repeat these steps for each computer in the Relay Server farm that is being updated.

Results

The Relay Server configuration is updated.

1.9 Relay Server Logging and Log Administration

The Relay Server log file displays information, warning, and error 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 Relay Server log file name and directory for Windows is *%SQLANY17%*`\RelayServer\IIS\Bin64\Log\rs.log`.

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

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

In this section:

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.

Outbound Enabler Record [page 52]
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.

Remote Administration Using AdminChannel (Microsoft Windows) [page 53]
The `AdminChannel` class in `rstool.jar` provides remote administration of the Relay Server configuration and log files.

Related Information

Relay Server State Manager (rshost) Command-line Syntax (Linux) [page 19]

1.9.1 Relay Server Logging and SAP Passports

SAP Passports trace requests from the client through to the backend server.

SAP Passports 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. 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.

The Relay Server supports SAP Passport version 2 and version 3.

Passport trace level	RS/OE verbosity level	Description
Low	1	Access level logging
Medium	4	Debug logging with packet logging (plus request header logging on the Relay Server side)
High	5	Debug logging with entire payload

The Relay Server and Outbound Enabler log lines associated with a request involving an SAP Passport have the passport logged in the Request description column of the following table:

SAP Passport version	Request description	Log line example
2	`R{#SAP-PPK#V2#<Transaction-uuid>}`	`I. 2015-05-05 14:38:06.898-0400 J{RSTEST01#F0#S0#1} R{1#SAP-PPK#V2#8fa46833ea42b94a8181b5bc8da3a33c 001e3700e6331ee1b4b2ac6ff58a2de0#001e3700e6331ee1b4b2a7d926ce4de0#001e3700e6331ee1b4b2ac6ff58a2de0#1} M{Relaying header}`
3	`R{#SAP-PPK#V3#<Transaction-uuid>#Root-context-uuid#Connection-uuid#Connection-counter}`	`I. 2015-05-05 14:38:06.898-0400 J{RSTEST01#F0#S0#1} R{2#SAP-PPK#V3#001e3700e6331ee1b4b2ac6ff58a2de0#001e3700e6331ee1b4b2a7d926ce4de0#001e3700e6331ee1b4b2ac6ff58a2de0#1} M{Relaying headers}`

1.9.2 Relay Server Record

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 RSR 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 values that summarize an HTTP request. To assist in interpreting Relay Server Records, the Relay Server log file contains a header describing the RSR values:

Symbol	Data type
b:	The byte count.
c:	Other count.
i:	An ID or numeric code.
m:	The time measured in milliseconds.
x:	A hex number.
`name:string`	A variable-length named string value (including Relay Server error names, Relay Server error parameters, and SAP Passport information).
oe	An element reported by the Outbound Enabler.

Symbol	Data type
up	An element associated with the request (excluding the response) from the Relay Server to the backend.
rtp	An element associated with round-trip transport and processing of the last up and first down packet.
dn	An element associated with the request (excluding the response) from the backend to the Relay Server.
in	The time spent waiting for input to read.
out	The time spent waiting for input to write.
A.B	This notation indicates that B is a child component, subprocess, or an aspect of A.
pkt/packet	The packet created by the Relay Server and/or the Outbound Enabler for communication over the junction.

The symbols are concatenated (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 junction (*up*) writing packets (*.out*) within the relaying period. The dot notation (.) also indicates that *out* is a sub-process of *up*.

Field name	Data type	Values
flag[0]	The affinity decision.	**n=new** indicates a new affinity **c=continue** indicates a subsequent request in an established affinity session **h=homed** indicates a new affinity with the designated backend server **r=renew** indicates that there is a collision, so the affinity information is renewed and a new section is opened up **x=expired** indicates that the Relay Server has signaled the client to expire the affinity cookie
flag[1]	Request persistence.	p=persistent, n=non-persistent
flag[2]	Request transfer encoding.	k=chunked, l=content-length
flag[3]	Response persistence.	p=persistent, n=non-persistent, u=unknown
flag[4]	Response transfer encoding.	k=chunked, l=content-length, u=unknown
b:up	Request size in bytes.	
b:dn	Response size in bytes.	
c:up:pkt	Number of upward request packets sent to the Outbound Enabler.	

Field name	Data type	Values
m:up	Request relaying period involving interleaving reads from the client, writes to the junction, and request packetization.	
m:up.in	Sum of time spent waiting for request payload from the client within the request relaying period.	
m:up.out	Sum of time spent writing packets to the junction within the relaying period.	
m:rtp	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.	
m:oe: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:rtp:kpi	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 (m:rtp - m:oe.rtp).	
c:dn.pkt	Number of downward response packets received from the Outbound Enabler.	
m:dn	Response relaying period involving interleaved reads from the junction and writes to the client.	
m:dn.in	Sum of time spent waiting for response packets from the junction within the response relaying period.	
m:dn.out	Sum of time spent waiting for writing response payload to the client within the response relaying period.	
m:oe.dn	Response receiving period observed by the Outbound Enabler. This period overlaps with time counted in m:dn.in and m:dn.out.	
m:close	Elapsed time between the end of m:dn and exiting the rs_client extension.	
i:dn.stts	HTTP response status.	
i:err	Error ID.	
i:warn	Warning ID.	

Field name	Data type	Values
err	Name of the error.	
warn	Name of the warning.	
oe.err	Name of the Outbound Enabler error.	
oe.err.p0	First error parameter from the Outbound Enabler.	
oe.err.p1	Second error parameter from the Outbound Enabler.	
oe.err.p2	Third error parameter from the Outbound Enabler.	
up.ua	User agent header of the request.	
up.uq	URL query parameters in `name=value` pairs. This can be useful for tagging information within the request when SAP Passports are not available.	
up.AfHdr	Affinity.	
up.cookie	Cookie header in request.	

label	Composite request prefix (located at the end of each line):
	`<label: J{relayserver-host#backend-farm-name#backend-server-name#junction-index} R{request-number}>`
	`<BEFarmName>`
	`<BeServerName>`
	For example:
	`<11436.4592.F0B0S0R0> <RSTEST02.F0> <S0>`	

A Relay Server Record looks similar to the following sample:

```
I. 2015-05-01 18:34:26.059-0400 RSR header 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:oe.dn|m:close|
i:dn.stts  i:err i:warn ...other-variable-length-elements...
I. 2015-05-01 18:34:30.286-0400 RSR data nnlnl 1530 1242 | 2 0 0 0 | 150 149 1 |
2 0 0 0 0 | 1 | 200  0 0 <err:> <warn:> <oe.err:> <oe.err.parameters:>
<up.ua:RSTestClient> <up.uq:> <up.AfHdr:> <up.cookie:> <label:
J{RSTEST01#F0#S0#0} R{1}>
```

Related Information

Affinity [page 10]

1.9.3 Outbound Enabler Record

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 OER 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 OER is comprised of a set of data types (represented by symbols) and values. To assist in interpreting Outbound Enabler Records, the Outbound Enabler log file contains a header describing the OER values. The header line is composed of symbols that are also described in the file. The header line symbols use the following convention:

Symbol	Data type
b:	Byte count.
i:	ID or numeric code.
m:	Time measured in milliseconds.
up	Element associated with the relaying request (excluding the response) to the backend.
rtp	Element associated with round-trip transport and processing of the last up and first down packet.
dn	Element associated with the relaying request (excluding the response) from the backend to the Relay Server.

The symbols are concatenated (as field names) to refer to particular kinds of data. For example, *m:up* corresponds to the duration of the request phase and *b:dn* corresponds to the number of bytes in the response..

Field name	Data type
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	Number of bytes sent to the backend server

Field name	Data type
b:dn	Number of bytes received from the backend server
i:err	Error ID
err	Name of the error
err.parameters	Error parameters

An Outbound Enabler Record looks similar to the following sample:

```
I. 2015-05-01 18:34:26.158-0400 T{000022a4} J{RS-host#farm#server#junction-
index} R{request-number} M{OER HEADER m:up m:rtp m:dn m:close | b:up b:dn |
i:err err err.parameters}
I. 2015-05-01 18:34:30.294-0400 T{00000480} J{RSTEST01#F0#S0#0} R{1} M{OER DATA
0 149 0 0 | 1575 1242 | 0 OK ()}
```

Related Information

Affinity [page 10]

1.9.4 Remote Administration Using AdminChannel (Microsoft Windows)

The AdminChannel class in rstool.jar provides remote administration of the Relay Server configuration and log files.

Command-line usage:

```
java com.sap.relayserver.AdminChannel [{options}]...
```

Option	Description
-url rsAdminUrl	Points to the rs_admin extension.
-uid user -pwd password	Provides credentials for HTTP authentication to access rs_admin.
-ping	Pings the rs_admin extension.
-getRSConfig	Retrieves the Relay Server configuration.
-setRSConfig new-config-file	Updates and backs up the Relay Server configuration.
-hello	Negotiates administration protocol version and ping.
-archiveRSLog	Truncates and archives the current online Relay Server log file.

Option	Description
-xol outFile *none* | beginTime *none* | endTime nRegex "regex"...	Extracts log lines from the archived and/or online local Outbound Enabler logs. The Relay Server does not need to be running and this function provides local extraction only. • *Time is the local timestamp in yyyy-MM-dd HH:mm:ss.SSS format • nRegex is a regular expression
-xrl outFile *none* | beginTime *none* | endTime nRegex "regex"...	Extracts log lines from archived and/or online Relay Server logs remotely. • *Time is the local timestamp in yyyy-MM-dd HH:mm:ss.SSS format • nRegex is a regular expression
? | -? | /? | -h | /h	Outputs this usage description.

Example

```
java.exe -cp rstool.jar com.sap.relayserver.AdminChannel -url https://
rs.my.com/rs17/admin/rs.dll -uid me -pwd passwd -hello
java.exe -cp rstool.jar com.sap.relayserver.AdminChannel -url https://
rs.my.com/rs17/admin/rs.dll -uid me -pwd passwd -xrl rr.xrl none none 1 "RSR
(element|header|data)"
```

1.10 The Relay Server with MobiLink

Use MobiLink to connect clients to Relay Server farms.

Perform the following tasks to use the Relay Server with MobiLink:

1. Set up a Relay Server farm for the MobiLink clients.
2. Run the Outbound Enabler.
3. Use the status page to test the configuration.
4. Run the MobiLink clients.

In this section:

Related Information

MobiLink Server Deployment
Supported Platforms
Relay Server Outbound Enabler Syntax [page 34]
Relay Server Status Page [page 11]

1.10.1 Setting Up a Relay Server Farm for MobiLink Clients (Command Line)

Configure and deploy the configuration file with the appropriate settings so that MobiLink clients can connect to a farm.

Procedure

1. Create the Relay Server configuration file. The file must be named `rs.config`.
2. Deploy the `rs.config` file along with the Relay Server components to the two computers that are running the Relay Server.
3. Start the Outbound Enabler and MobiLink server on each backend server.

Results

Once all servers and Outbound Enablers are running, MobiLink clients can connect to the farm.

Example

Suppose company ABC developed a mobile application and wants to set up the deployment run-time to service the mobile application. Initially, the mobile deployment consists of 10000 devices and grows in the future. The customer wants a fault tolerant and load-balanced environment that handles the load today and can 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, and 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. 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.

This example uses the Microsoft IIS version of the Relay Server.

1. Create the Relay Server configuration file.
   The configuration file is named `rs.config`. For this particular scenario, the following configuration file is used:

   ```
   #
   # 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 will connect 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 `rs.config` file along with the Relay Server components to the two computers that are running the Relay Server.
3. On the computer running the MobiLink server with the ID ml2, run the following commands:

   ```
   mlsrv17 -x http(port=80)
   ```

   ```
   rsoe2 -f abc.mobilink -id ml2 -t mltoken2 -cr
   "host=www.abc.com;port=443;https=1;url_suffix=/rs17/server/rs.dll" -cs
   "host=localhost;port=80"
   ```
4. On the computer running the MobiLink server with ID ml1, run the following commands:

   ```
   rsoe2 -f abc.mobilink -id ml1 -t mltoken1 -cr
   "host=www.abc.com;port=443;https=1;url_suffix=/rs17/server/rs.dll" -cs
   "host=localhost;port=80"
   ```

   ```
   mlsrv17 -x http(port=80)
   ```
5. Once all Outbound Enablers and servers are running, MobiLink clients can connect to the farm by using the following connection information:

HTTPS

protocol

host

www.abc.com

url_suffix

/rs/client/rs.dll/abc.mobilink

Related Information

End-to-end Encryption
The Relay Server Configuration File [page 20]
-x mlsrv17 Option

1.11 Client Connections to a Relay Server Farm

Once a Relay Server farm is configured, clients can connect to it.

The client connects by using the following URL:

```
http://Relay-Server-client-extension-URL/farm-name
```

Options

Option	Description
`Relay-Server-client-extension-URL`	For Microsoft IIS on Windows: `<domain name><relayserver.sap.com>/rs/client/rs.dll` For Apache on Linux: `<domain name>/cli/iarelayserver`
`farm-name`	This value identifies the backend farm that the Relay Server forwards the client request to.

SQL Anywhere MobiLink Client Connection Example

The following command connects a SQL Anywhere MobiLink client to server farm, F1, over HTTP:

```
-e "ctp=http;
     adr='host=relayserver.sap.com;
        url_suffix=/rs17/client/rs.dll/F1'"
```

UltraLite MobiLink Client Connection Example

The ULSyncParms class properties connect an UltraLite MobiLink client to server farm F1:

- Specify the stream type (HTTP or HTTPS)
- Set the stream parameters to the following values:

  ```
  "host=my_rs.my_corp.com;url_suffix=/rs17/client/rs.dll/F1"
  ```

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
    - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
    - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP