Using a reverse proxy with OutSystems

What is a reverse proxy?

A reverse proxy is an application that receives requests from the Internet and forwards them to a set of servers. These servers are usually located on an internal network and are not directly accessible from outside.

Reverse proxy capabilities:

• Load Balancing (TCP Multiplexing)
• SSL Offload/Acceleration (SSL Multiplexing)
• Caching
• Compression
• Content Switching/Redirection
• Application Firewall
• Server Obfuscation
• Authentication
• Single Sign-On

Why use a reverse proxy with OutSystems?

1. **Reduce Load on Application Servers**
   Reverse proxies have the ability to compress and cache content, encrypt data (HTTPS), relieving these tasks from the application server.

2. **Increased security and traceability**
   With all application traffic passing through the reverse proxy, logging, authentication and access control can be configured and managed in a centralized place. It also prevents Internet exposure of application servers, protecting from vulnerabilities the servers might have in other software or
3. **Ensure High Availability**
Reverse proxies can support high-availability methods. This will allow you to eliminate downtime. In an OutSystems farm scenario, there are multiple application servers. The reverse proxy will then enforce a load balancing technique like round-robin to distribute the load among the servers in the cluster.

When a server goes down for maintenance, the system will automatically failover to the next server in the round-robin sequence keeping the applications available.

## Requirements

When serving OutSystems applications through a reverse proxy, the following requirements must be guaranteed:

- **Maintain original host header of the request**
The original host header must be sent to the application server. OutSystems uses this name when generating URLs that go in the HTML. Keeping the original host header prevents internal server names to be used in the HTML generation and ensures URLs are correct.

- **Referenced content availability**
All modules that are used somehow by your application must be available from the outside world. For example, if you have a module `myApp` that uses a screen in `myOtherApp` module for logging in, and images from an `ImageRepository` module, then you need to have the following URL available to the outside world:

  http://www.publicname.com/myApp/
  http://www.publicname.com/myOtherApp/
  http://www.publicname.com/ImageRepository/

- **Only the default ports for protocols HTTP and HTTPS are supported**
OutSystems only serves applications in HTTP port 80 and HTTPS port 443.

- **Disable NEGOTIATE protocol for Windows Integrated Authentication**
In some scenarios, your IIS may have the NEGOTIATE protocol (Kerberos ticket) activated for Windows Integrated Authentication. External users will not be able to use that protocol, and some browsers may block access to your applications if they try to use it. Refer to [this Knowledge Base article](https://success.outsystems.com/Support/Enterprise_Customers/Maintenance_and_Operations/Using_OutSystems_in_Revers…Updated: Wed, 04 Mar 2020 15:00:38 GMT) for more details.

Additionally, when the back-end of one or more OutSystems **Mobile or Reactive Apps** is hosted behind a reverse proxy, the following requirements must also be met:

- **Applications available externally using the domain name configured in Service Center and**
Applications must be externally accessible using their domain name configured in Service Center (which by default is the hostname of the environment) and served via HTTPS.

- **URL paths of applications must be kept unchanged**
  The reverse proxy must not transform URL paths of applications in any way, i.e. applications must be externally available at https://<mobile_app_domain_name_in_Service_Center>/<application_name>.

- **Do not cache Mobile/Reactive App resources in the reverse proxy**
  Mobile and Reactive Apps created in OutSystems load their resources through an optimized process. However, proxy caching can interfere with those optimizations. Make sure that the reverse proxy does not cache Mobile/Reactive App resources to prevent runtime problems in those apps.

- **When using keep-alive/persistent connections, review the connection timeout configuration**
  If your reverse proxy is using persistent connections and thus issuing keep-alive headers, make sure that either there’s no defined keep-alive connection timeout or that the timeout is set to at least 30 seconds. If you are using Apache (or an Apache-based) web server being used as a reverse proxy, review the keep-alive connection timeout configuration being used, since in these servers the default connection timeout value is 5 seconds. This configuration adjustment is described in [F - Adjust keep-alive connection timeout](https://success.outsystems.com/Support/Enterprise_Customers/Maintenance_and_Operations/Using_OutSystems_in_Revers...).