NetIQ Advanced Authentication Framework

Smartphone Authentication Dispatcher Installation Guide

Version 5.1.0

Table of Contents

© *NetIQ*

Introduction

About This Document

Purpose of the Document

This Smartphone Authentication Dispatcher Installation Guide is intended for all system administrators and describes how to use the client part of NetIQ Advanced Authentication Framework solution. In particular, it gives instructions as for how to install Smartphone Authentication Dispatcher.

For more general information on NetIQ Advanced Authentication Framework™ and the authentication software you are about to use, see NetIQ Advanced Authentication Framework – Client User's Guide.

Information on managing other types of authenticators is given in separate guides.

Document Conventions

⚠ **Warning.** This sign indicates requirements or restrictions that should be observed to prevent undesirable effects.

✱ **Important notes.** This sign indicates important information you need to know to use the product successfully.

ℹ **Notes.** This sign indicates supplementary information you may need in some cases.

❓ **Tips.** This sign indicates recommendations.

- Terms are italicized, e.g.: ***Authenticator***.
- Names of GUI elements such as dialogs, menu items, buttons are put in bold type, e.g.: the **Logon** window.

System Requirements

The following system requirements should be fulfilled:

- Microsoft Windows Server 2008 R2 SP1/Microsoft Windows Server 2012

Installing and Removing Smartphone Authentication Dispatcher

NetIQ Advanced Authentication Framework™ package includes Smartphone authentication dispatcher, which is responsible for establishing connection between the NetIQ Smartphone Authenticator and Smartphone authentication provider

After the upgrade of Smartphone Authentication Dispatcher v1.1.32 and earlier to v.1.1.44 and later, the push messaging will not work during up to 6 hours and the app will need to be opened manually. After that time, if during 6 hours after upgrade the app was not opened, it must be opened manually for the first time.

Installing Smartphone Authentication Dispatcher

Smartphone Authentication Dispatcher is designed to provide connection between the NetIQ Smartphone Authenticator and Smartphone authentication provider. Smartphone Authentication Dispatcher receives HTTP requests from a mobile device that is running the mobile part of NetIQ. Also it serves requests from Smartphone authentication provider.

Moreover Smartphone Authentication Dispatcher transfers Push Notifications to mobile devices that are running NetIQ Smartphone Authenticator through the special proxy-server (proxy.authasas.com).

Smartphone Authentication Dispatcher monitors the status of authentication and provides special APIs for BSP (and other exterior applications). It performs a range of tests that verify data authenticity.

Only one Smartphone Authentication Dispatcher can be installed in an environment.

Before the installation of Smartphone Authentication Dispatcher make sure that .NET Framework 4.5 is installed on your computer.

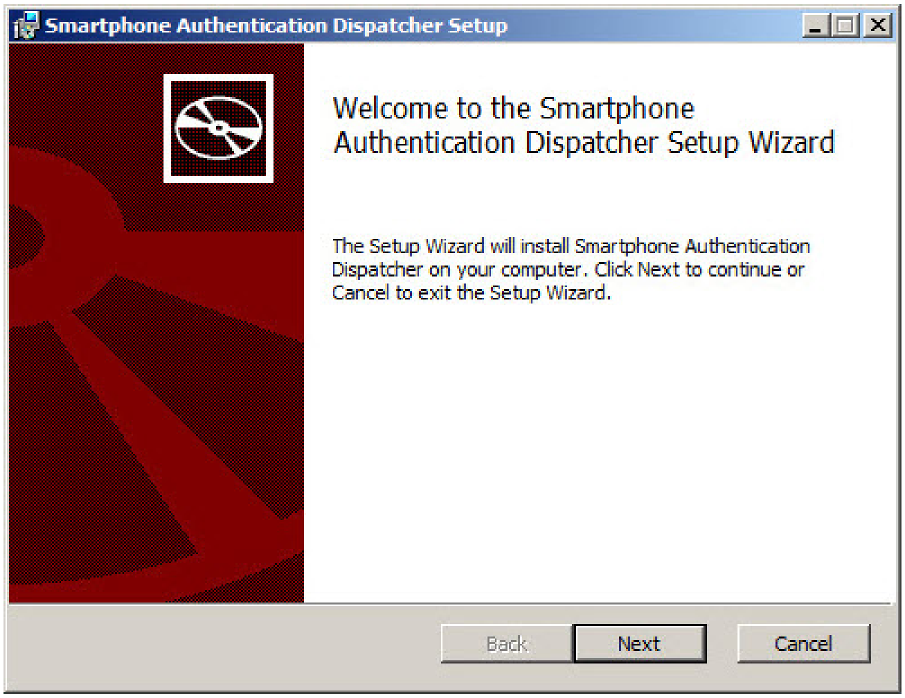
Smartphone Authentication Dispatcher should have the Internet access (it should have an access to proxy.authasas.com via https protocol, port 443).

The start of installation may be frozen for a time up to 1 minute in the case of offline mode. This delay occurs due to check of digital signature of component.

Smartphone Authentication Dispatcher can be installed on the server only.

To install Smartphone Authentication Dispatcher:

1. Run **SaDispatcher.msi**. The **Smartphone Authentication Dispatcher Setup** window will be displayed.

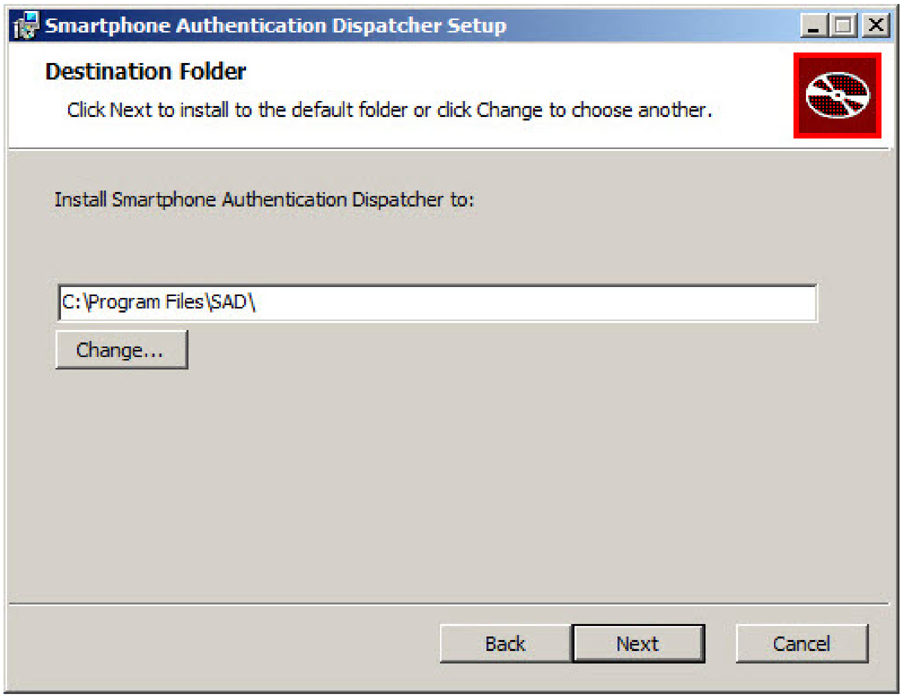


Smartphone Authentication Dispatcher Setup

Welcome to the Smartphone Authentication Dispatcher Setup Wizard

The Setup Wizard will install Smartphone Authentication Dispatcher on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

2. Click **Next** to install to the default folder or click **Change** to choose another.



Smartphone Authentication Dispatcher Setup

Destination Folder

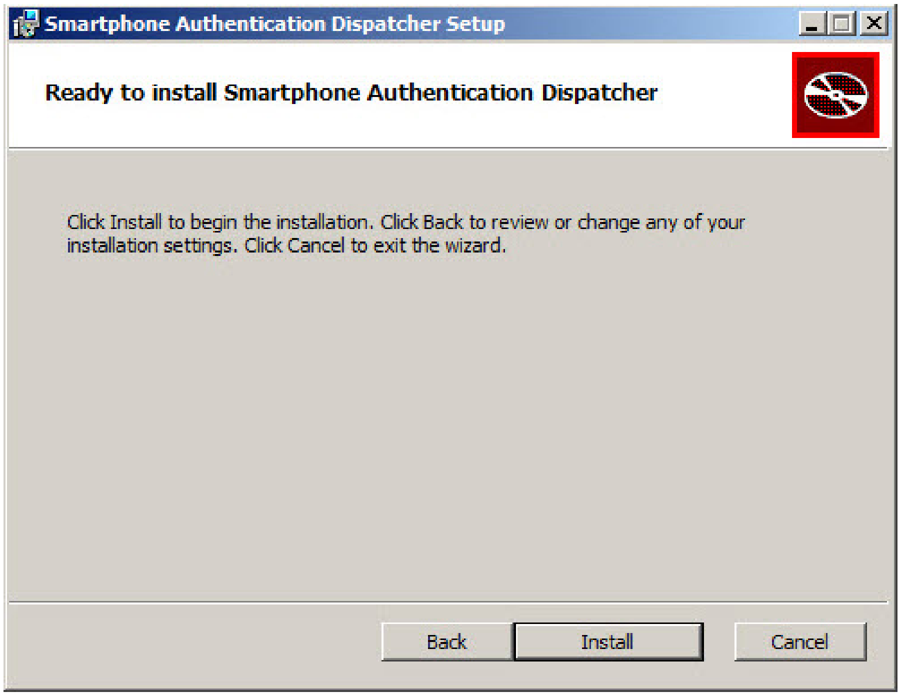
Click Next to install to the default folder or click Change to choose another.

Install Smartphone Authentication Dispatcher to:

C:\Program Files\SAD\

© *NetIQ*

3. Click **Install** to begin the installation. Click **Back** to review or change any of your installation settings. Click **Cancel** to exit the wizard.

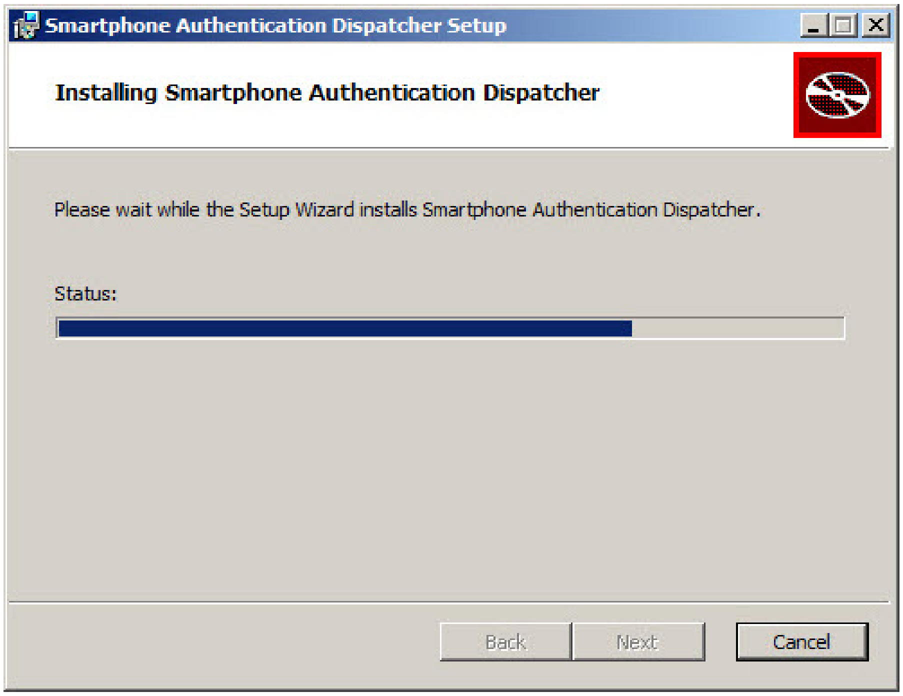


Smartphone Authentication Dispatcher Setup

Ready to install Smartphone Authentication Dispatcher

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

4. Please wait while the Setup Wizard installs Smartphone Authentication Dispatcher.



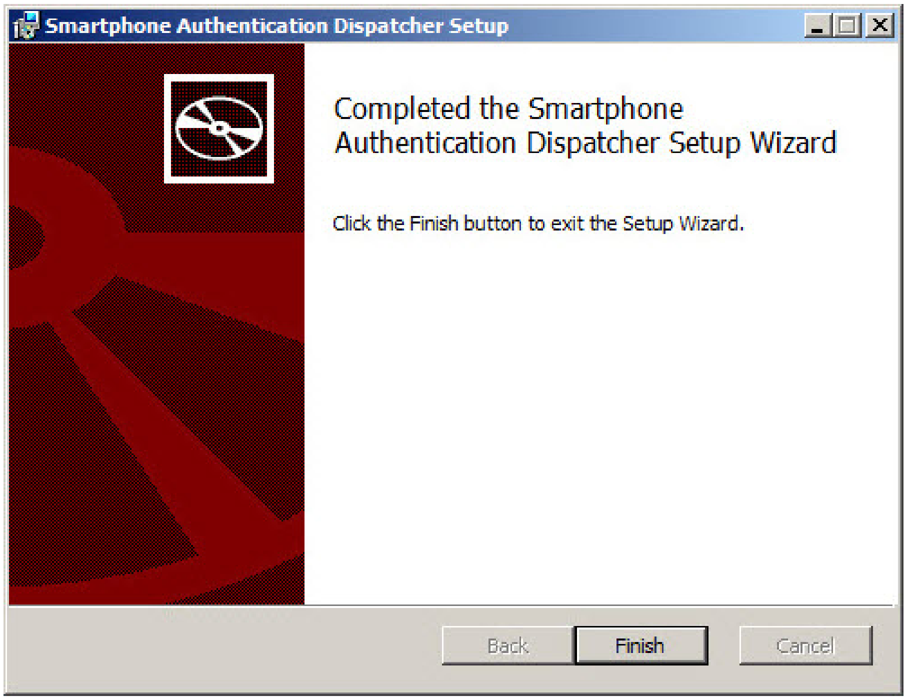
Smartphone Authentication Dispatcher Setup

Installing Smartphone Authentication Dispatcher

Please wait while the Setup Wizard installs Smartphone Authentication Dispatcher.

Status:

5. Click the **Finish** button to exit the Setup Wizard.



Completed the Smartphone Authentication Dispatcher Setup Wizard

Click the Finish button to exit the Setup Wizard.

Configuring Smartphone Authentication Dispatcher via Group Policy

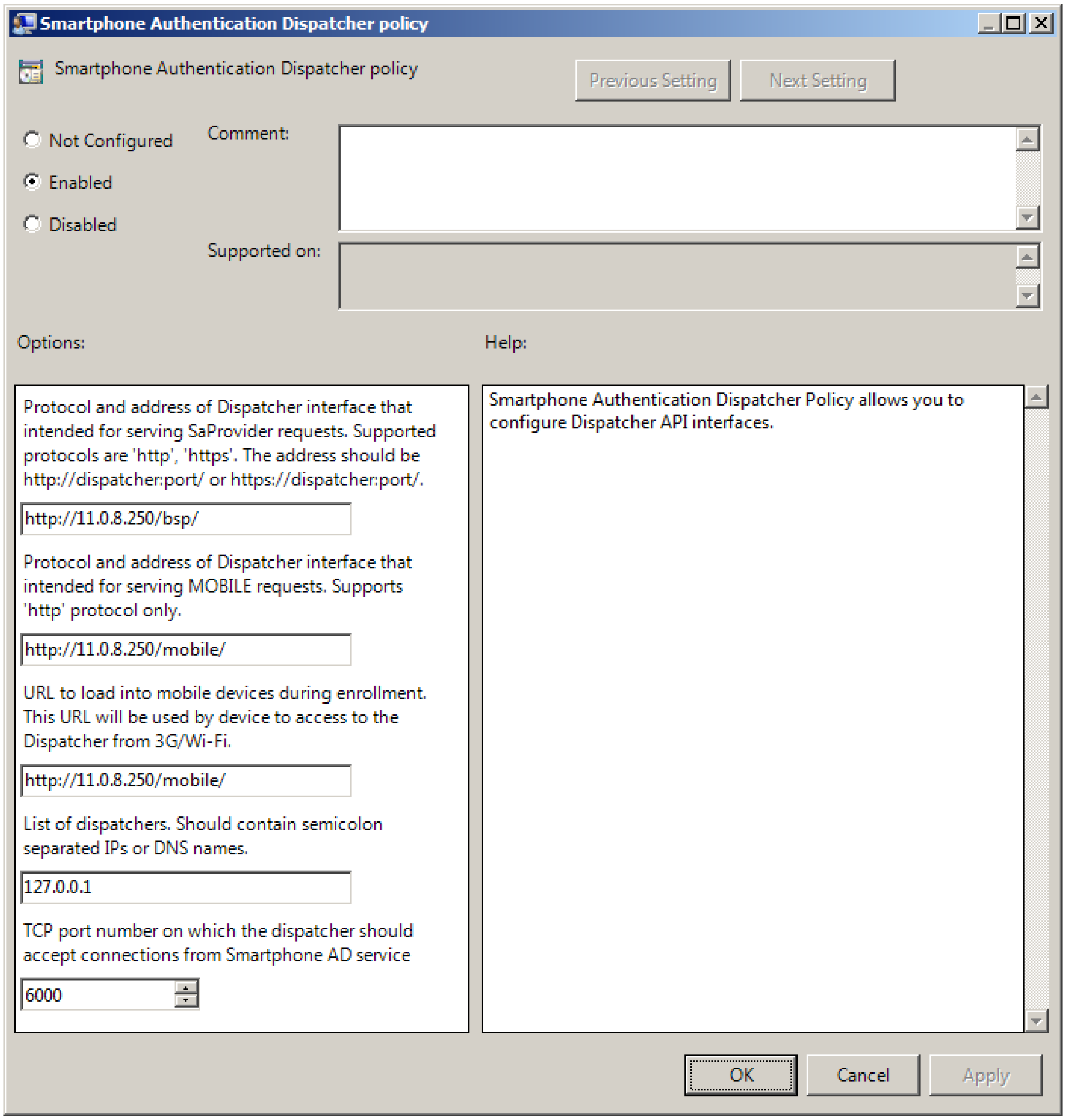
After the installation of both server components (**SaDispatcher.msi** and **SaProvider.msi**), the **Smartphone Authentication Dispatcher** policy will be successfully added to **Group Policy Management Editor**. To activate a policy:

- In **Group Policy Management Editor**, double-click the policy name.
- In the properties dialog, click **Enabled**.

The **Smartphone Authentication Dispatcher** policy allows you to configure Dispatcher API interfaces. It has the following options:

- Protocol and address of Dispatcher interface that intended for serving SaProvider requests. Supported protocols are 'http' and 'https'.
- Protocol and address of Dispatcher interface that intended for serving MOBILE requests. Supports 'http' protocol only. It is the "endpoint" which will accept connections from mobile devices.
- URL to load into mobile devices during enrollment. This URL will be used by device to access to the Dispatcher from 3G/WI-FI. This address will be encoded into QR code, scanned by mobile device and used by it to connect to Smartphone authentication Dispatcher.
- List of dispatchers. Should contain semicolon separated IP's or DNS names (in case of multiple dispatchers).
- TCP port number on which the dispatcher should accept connections from Smartphone AD service.

To save changes, click the **Apply** button.



Smartphone Authentication Dispatcher policy

Smartphone Authentication Dispatcher policy

○ Not Configured

● Enabled

○ Disabled

Comment:

Supported on:

Options:

Protocol and address of Dispatcher interface that intended for serving SaProvider requests. Supported protocols are 'http', 'https'. The address should be http://dispatcher:port/ or https://dispatcher:port/.

http://11.0.8.250/bsp/

Protocol and address of Dispatcher interface that intended for serving MOBILE requests. Supports 'http' protocol only.

http://11.0.8.250/mobile/

URL to load into mobile devices during enrollment. This URL will be used by device to access to the Dispatcher from 3G/Wi-Fi.

http://11.0.8.250/mobile/

List of dispatchers. Should contain semicolon separated IPs or DNS names.

127.0.0.1

TCP port number on which the dispatcher should accept connections from Smartphone AD service

6000

Help:

Smartphone Authentication Dispatcher Policy allows you to configure Dispatcher API interfaces.

- The changes take effect after group policy refresh and restart of Smartphone authentication dispatcher.
- While upgrading Smartphone Authentication Dispatcher to v.4.11.16 and later, it is required to change 'rpc' protocol to 'http' in policy settings.
- Try to open the URL from the **DispExternalMobileInterface** parameter together with the specified port number in browser on the smartphone. There should be displayed the following message: "*Smartphone dispatcher is running*".
- After the installation of Smartphone authentication dispatcher, the cluster mode is disabled by default .
- It is required to specify an applicable value for the **DispIpList** parameter only in case of multiple dispatchers.

To disable an applicable Dispatcher interface, specify the **disabled** value in the registry for the corresponding parameter. As a result the disabled interface will not be started after the launch of Smartphone authentication dispatcher.

HKEY_LOCAL_MACHINE\SOFTWARE\(Wow6432Node\)Policies\BioAPI\BSP\SaDispatcher

DispBspInterface:

- type: REG_SZ
- value: http:<IPAddressForBSPConnections>:<port>
- description: rpc:<IPAddressForBSPConnections>:<port> is protocol and address of Dispatcher interface that are intended for serving SaProvider requests

DispDataPort:

- type: REG_DWORD
- value: 0x00001770 (6000)
- description: 6000 displays the TCP port number that is used to accept connections from Smartphone AD service

DispExternalMobileInterface:

- type: REG_SZ
- value: http://<ExternalIPForMobileConnections>:<port>
- description: http://<ExternalIPForMobileConnections>:<port> is URL to load into mobile devices during enrollment

DispIpList:

- type: REG_SZ
- value: http://<DispatcherIPAddress>
- description: http://<DispatcherIPAddress> is IP address of an applicable dispatcher.

DispMobileInterface:

- type: REG_SZ
- value: http://<IPAddressForMobileConnections>:<port>/
- description: http://<IPAddressForMobileConnections>:<port>/ is protocol and address of Dispatcher interface that are intended for serving mobile requests

Parameters enclosed in angle brackets (<parameter>) should be replaced with applicable values (including angle brackets).

These settings should be applied on Authenticore Server and workstations.

Configuring Smartphone Authentication Dispatcher to Work Through HTTP Proxy

To configure Smartphone Authentication Dispatcher to work through HTTP proxy, follow the steps:

1. Open the **SaDispatcher.exe.config** file.
2. Add **proxyAddress="<IP address of http proxy><port>"** and **useDefaultProxy="false"** attributes to the **<binding>** tag. E.g.:

```
<system.serviceModel>
<bindings>
<basicHttpBinding>
<binding name="BasicHttpsBinding_IPushSenderProxy"
closeTimeout="00:00:05" openTimeout="00:00:05" receiveTimeout="00:00:05" sendTimeoutt="00:00:05"
proxyAddress="http://<IP Address>:<port>" useDefaultWebProxy="false">
<security mode="Transport"/>
</binding>
</basicHttpBinding>
</bindings>
<client>
<endpoint address="https://proxy.authasas.com/OobProxy/Service.svc" binding="basicHttpBinding" bindingConfiguration="BasicHttpsBinding_IPushSenderProxy" contract="IPushSenderProxy" name="BasicHttpsBinding_IPushSenderProxy"/>
</client>
</system.serviceModel>
```

Multiple Dispatchers Support in v4.10 R3

Multiple dispatchers support provides with an opportunity of exchanging updates between dispatchers and keeping data in actual state. In case of installation of several dispatchers, the workflow will be the following:

- The main dispatcher receives data which should be saved in the local database (a new device registration information, push ID update from the registered device, etc.).
- The main dispatcher writes new data in its local database.
- The main dispatcher sends notifications to all specified subsidiary dispatchers.
- All subsidiary dispatchers load updates from the main dispatcher.

In this chapter:

Multiple Dispatchers Support Configuration

To configure multiple dispatchers support:

1. For the main dispatcher:
   - Open the **SaService.exe.config** file on the server.
   - Specify the following values:
     - <add key="DispBspInterface" value="**rpc:<IPAddressForBSPConnections>:<port>**"/>
     - <add key="DispMobileInterface" value="**disabled**"/>
     - <add key="DispExternalMobileInterface" value=""/>
     - <add key="DispSyncInterface" value="**https://<IPAddressForSynchronization>:<port>**"/>
   - Save the file.

The **DispMobileInteface** value should be set to **disabled**. Otherwise the dispatcher will use the value that is specified in the **Smartphone Authentication Dispatcher** policy.

2. For the subsidiary dispatcher:

- Open the **SaService.exe.config** file on the server.
- Specify the following values:
  - <add key="DispBspInterface" value="**disabled**"/>
  - <add key="DispMobileInterface" value="**http://<IPAddressForMobileConnections>:<port>**"/>
  - <add key="DispExternalMobileInterface" value=""/>
  - <add key="DispSyncInterface" value="**https://<IPAddressForSynchronization>:<port>**"/>
- Save the file.

The **DispBspInterface** value should be set to **disabled**. Otherwise the dispatcher will use the value that is specified in the **Smartphone Authentication Dispatcher** policy.

Cluster options are not related to any of the domain policies because cluster nodes work outside the domain infrastructure.

All dispatchers use a special network interface for communication with each other. To configure it, it is required to specify the same **DispSyncInterface** value in the **SaService.exe.config** file for every dispatcher.

The **externaldisps.txt** file is used for storing configuration information about all dispatchers. This file contains URI and password for each of dispatchers. When the dispatcher starts, it reads this file, finds itself and gets a password to use it in decryption of incoming messages. All dispatchers know all passwords and use them during synchronization exchange.

Smartphone Dispatcher Configurer

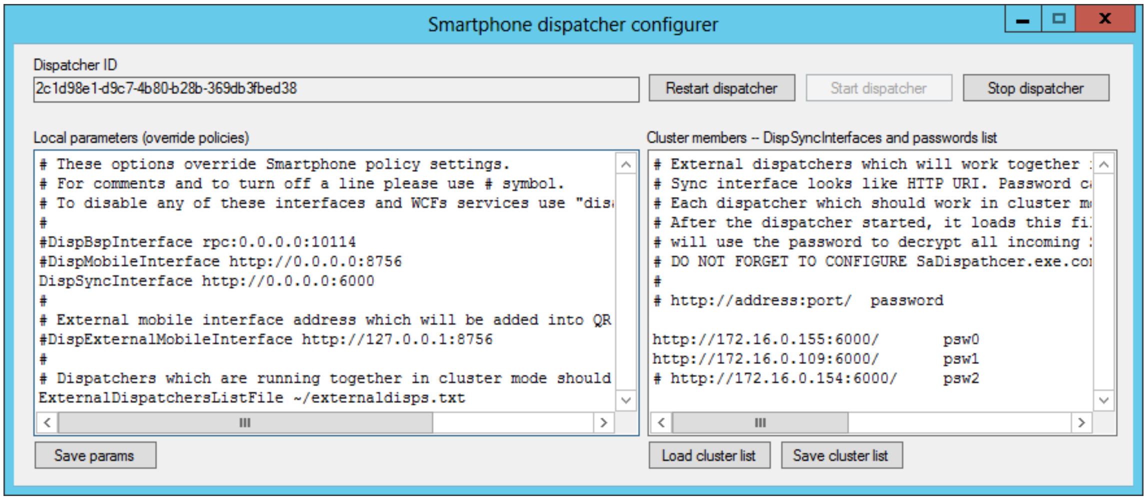
It is highly recommended to use configuration tool instead of direct modification of **SaService.exe.config** file.

The **Smartphone dispatcher configurer** is intended to avoid the modification of **SaService.exe.config** file. The modified **SaService.exe.config** file will be replaced by next update and all configured options will be lost.

The **Smartphone dispatcher configurer** has a single window which is split into two text areas:

- The **Local parameters** text area contains configurable parameters and comments.
- The **Cluster members** text area contains the cluster node list.

The **Smartphone dispatcher configurer** contains the same configurable parameters as the **SaService.exe.config** file.



Dispatcher ID: 2c1d98e1-d9c7-4b80-b28b-369db3fbed38

Local parameters (override policies)

```
# These options override Smartphone policy settings.
# For comments and to turn off a line please use # symbol.
# To disable any of these interfaces and WCFs services use "dis
#
#DispBspInterface rpc:0.0.0.0:10114
#DispMobileInterface http://0.0.0.0:8756
DispSyncInterface http://0.0.0.0:6000
#
# External mobile interface address which will be added into QR
#DispExternalMobileInterface http://127.0.0.1:8756
#
# Dispatchers which are running together in cluster mode should
ExternalDispatchersListFile ~/externaldisps.txt
```

Cluster members -- DispSyncInterfaces and passwords list

```
# External dispatchers which will work together
# Sync interface looks like HTTP URI. Password c
# Each dispatcher which should work in cluster m
# After the dispatcher started, it loads this fi
# will use the password to decrypt all incoming
# DO NOT FORGET TO CONFIGURE SaDispathcer.exe.co
#
# http://address:port/  password

http://172.16.0.155:6000/       psw0
http://172.16.0.109:6000/       psw1
# http://172.16.0.154:6000/     psw2
```

After specifying all required parameters, click the **Save params** button to save the configuration. Then click **Restart dipatcher**.

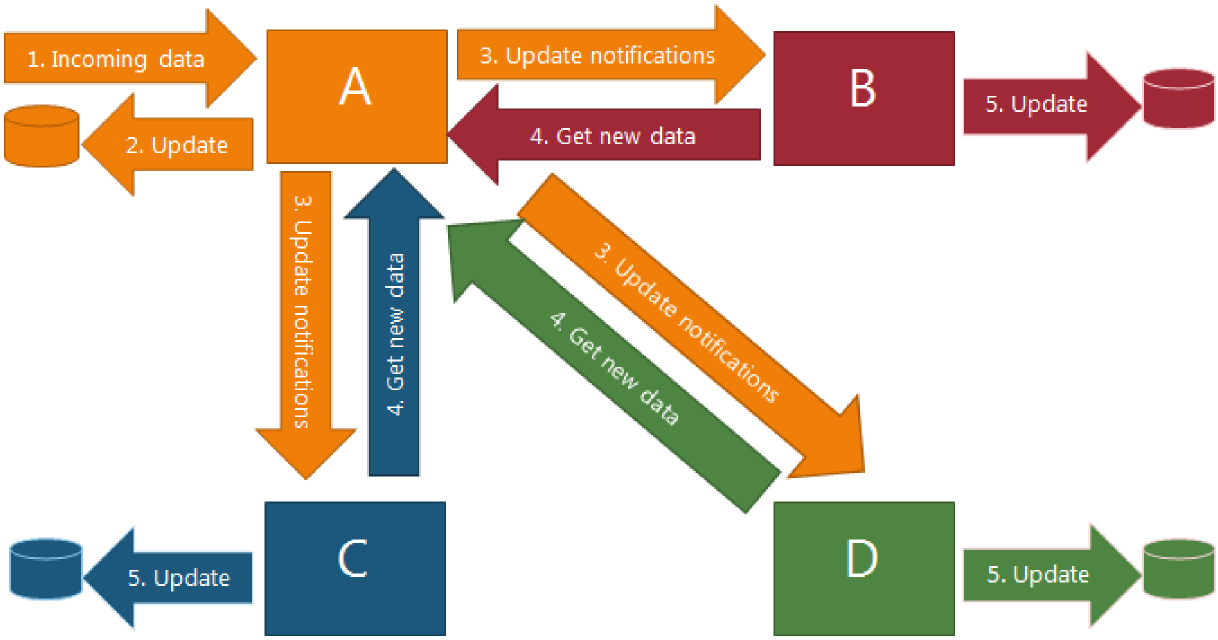
The **Smartphone dispatcher configurer** saves parameters to the predefined **props.config** file. This file will not be overwritten during update. All parameters, that are specified in the **props.config** file, override parameters specified in the **Smartphone Authentication Dispatcher** policy and **SaService.exe.config** file.

Dispatchers Database

Dispatchers have their own local databases. Every row in the database of the main dispatcher contains the following servicing records:

- **GUID** – global unique identifier;
- **DT** – date and time of adding or updating the record;
- **Version** – data version of the added or changed record.

In case of multiple dispatchers installation, the main dispatcher will exchange all updates and all data will be kept in actual state.



1. Dispatcher A receives data that should be saved into the local database (new device registration information or push ID update from the registered device).
2. Dispatcher A writes new data to its local database.
3. Dispatcher A sends notifications to all known dispatchers.
4. All known dispatchers load updates from dispatcher A.
5. All known dispatchers write new data to their local databases.

ℹ Any known dispatcher can act as the main dispatcher (dispatcher A).

The data version number is incremented every time the data is changed in the database. After the data is changed, the main dispatcher runs a set of background threads which send notifications to all subsidiary dispatchers. The notification contains source dispatcher ID and data version. The subsidiary dispatcher, which received a notification, checks the last stored data version of the main dispatcher. If the received data version is newer than the stored version, the dispatcher finds the sender's URI by known sender ID. When the URI has been found, the subsidiary dispatcher asks the main dispatcher for data changes, which were done between last known data version and the version from notification. The main dispatcher sends an answer with a set of required data.

Collision

Collision is a situation when the subsidiary dispatcher gets an update from the main dispatcher and this update contains a record with the same GUID as a record which the main dispatcher contains already in its database. In this situation the subsidiary dispatcher compares DT fields to decide which of these two records is newer. Only newer records will be accepted.

Time Synchronization

Each dispatcher asks the configured NTP server (by default time.windows.com) every 3600 seconds during its work. The time received from NTP server is used to calculate a correction for a system. All data records contain the corrected time.

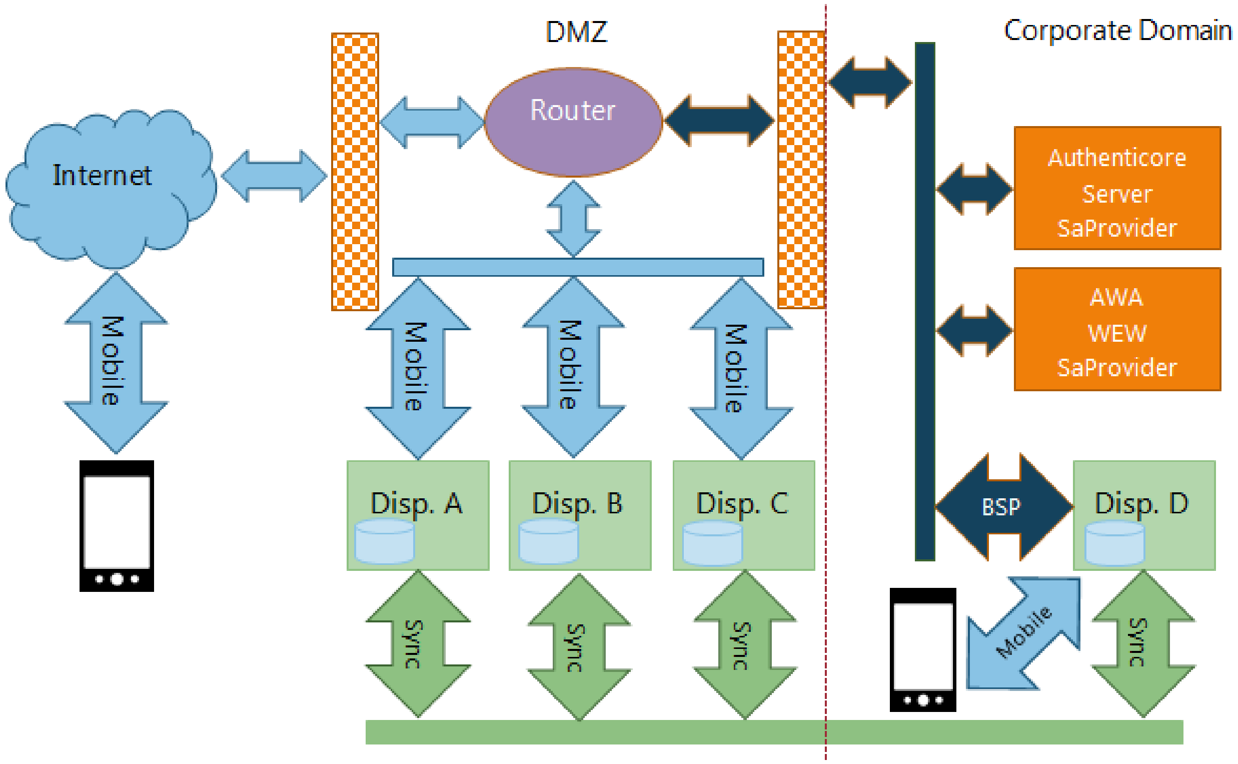
✱ The value of time synchronization can be changed in the configuration file.

© NetIQ

Internal State Synchronization

Internal state of the dispatcher is a set of authentication statuses and their synchronization objects. Unlike the data synchronization, internal state synchronization works with locks. It means that every event which comes to the main dispatcher from outside (except subsidiary dispatchers) causes a sequence of communications between the main dispatcher and subsidiary dispatchers. After the data update on all secondary dispatchers, the main dispatcher changes its internal state in accordance with received event.

As a result the dispatcher can support distributed Smartphone authentication. E.g., NetIQ Smartphone Authenticator can request salt from the main dispatcher, and after that NetIQ Smartphone Authenticator can send authentication answer to one of the subsidiary dispatchers.



Multiple Dispatchers Support in v4.11

Starting from NetIQ Advanced Authentication Framework v4.11 multiple dispatchers support is available only through ARR/IIS.

In this chapter:

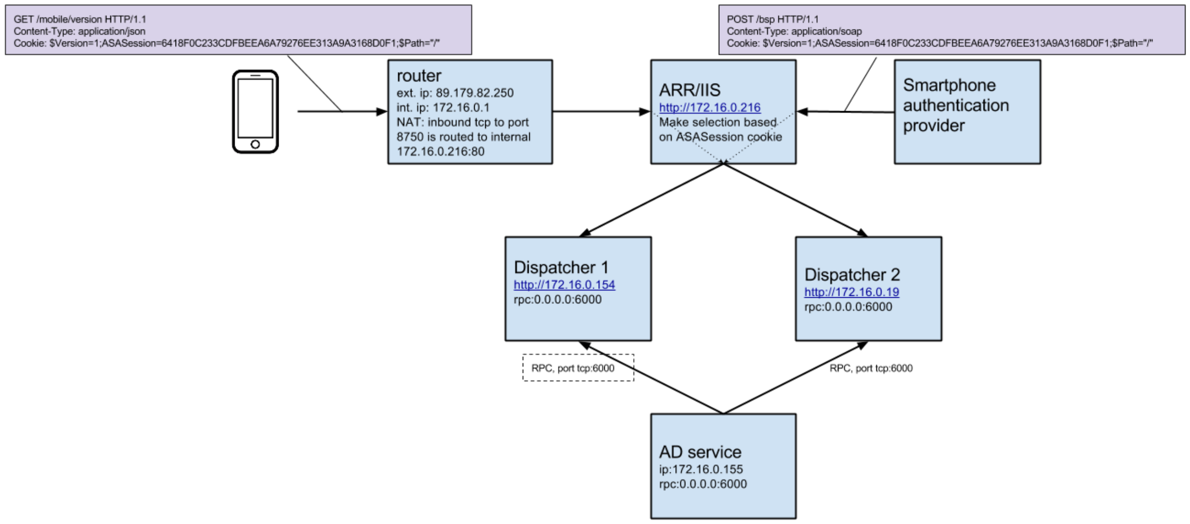
- Multiple Dispatchers Support Overview
- Installing Application Request Routing
- Configuring Application Request Routing

Multiple Dispatchers Support Overview

Both NetIQ Smartphone Authenticator and Smartphone authentication provider should have access to Smartptone authentication dispatcher through ARR/IIS.

Smartphone AD service is intended to serve extended AD/LDS scheme which stores Push ID. Push ID is a variable data. It is used to send popup (push) notifications. Smartphone AD service connects to all dispatchers whose IP addresses are included into the **Smartphone Authentication Dispatcher policy**. Smartphone AD service uses the same TCP port for all dispatchers. The dispatcher and the AD service interact with each other throug RPC protocol.

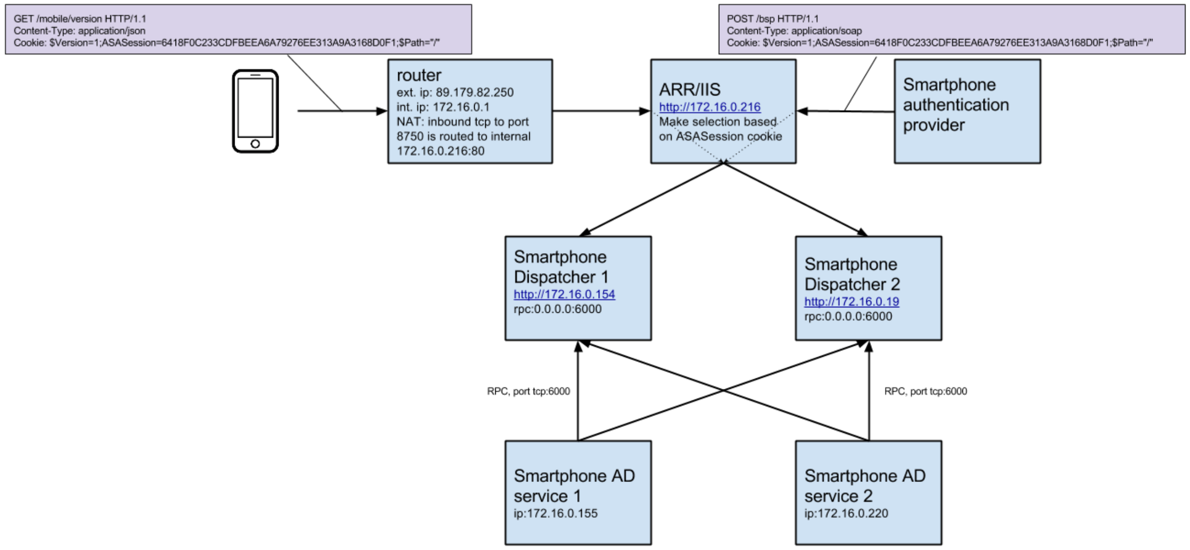
The figure below illustrates the Smartphone authentication dispatcher workflow.



```
GET /mobile/version HTTP/1.1
Content-Type: application/json
Cookie: $Version=1;ASASession=6418F0C233CDFBEEA6A79276EE313A9A3168D0F1;$Path="/"
```

```
POST /bsp HTTP/1.1
Content-Type: application/soap
Cookie: $Version=1;ASASession=6418F0C233CDFBEEA6A79276EE313A9A3168D0F1;$Path="/"
```

The next figure illustrates Smartphone authentication dispatcher workflow when Smartphone AD service is installed on multiple machines.


GET /mobile/version HTTP/1.1
Content-Type: application/json
Cookie: $Version=1;ASASession=6418F0C233CDFBEEA6A79276EE313A9A3168D0F1;$Path="/"
POST /bsp HTTP/1.1
Content-Type: application/soap
Cookie: $Version=1;ASASession=6418F0C233CDFBEEA6A79276EE313A9A3168D0F1;$Path="/"
router
ext. ip: 89.179.82.250
int. ip: 172.16.0.1
NAT: inbound tcp to port 8750 is routed to internal 172.16.0.216:80
ARR/IIS
http://172.16.0.216
Make selection based on ASASession cookie
Smartphone authentication provider
Smartphone Dispatcher 1
http://172.16.0.154
rpc:0.0.0.0:6000
Smartphone Dispatcher 2
http://172.16.0.19
rpc:0.0.0.0:6000
RPC, port tcp:6000
RPC, port tcp:6000
Smartphone AD service 1
ip:172.16.0.155
Smartphone AD service 2
ip:172.16.0.220

Installing Application Request Routing

To install ARR and all its components in the appropriate order, use the Microsoft Web Platform Installer. Follow the steps:

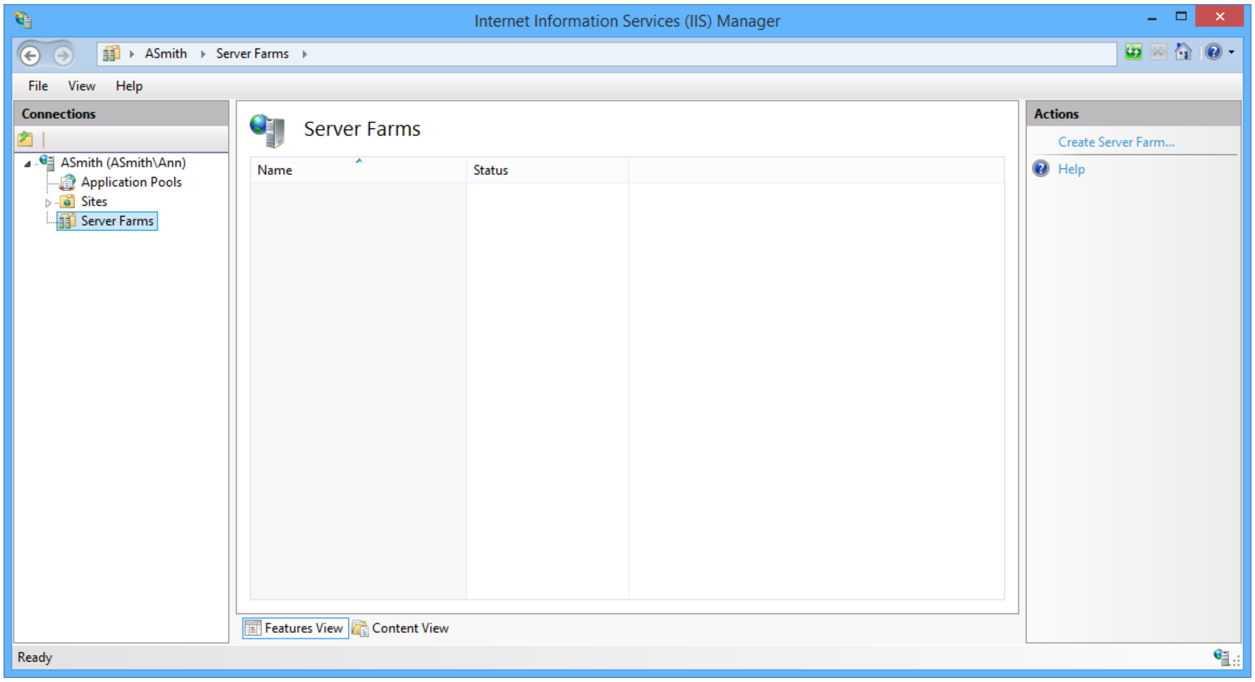
1. Go to the Official Microsoft IIS Site.
2. Open the Application Request Routing page and click **Install this extension**.
3. Click **Install Now**. The installation file will be downloaded on your computer.
4. Run the installation file. The Web Platform Installer will be launched.
5. Click **Install**.
6. Click **I accept** if you agree to the license terms of the third party and Microsoft software. Wait until the components are installed.
7. View the list of products that were installed. Click **Finish**.
8. Click **Exit** to close the Web Platform Installer.

Configuring Application Request Routing

After the installation of Application Request Routing, it is required to create a Server Farm and add dispatchers to it.

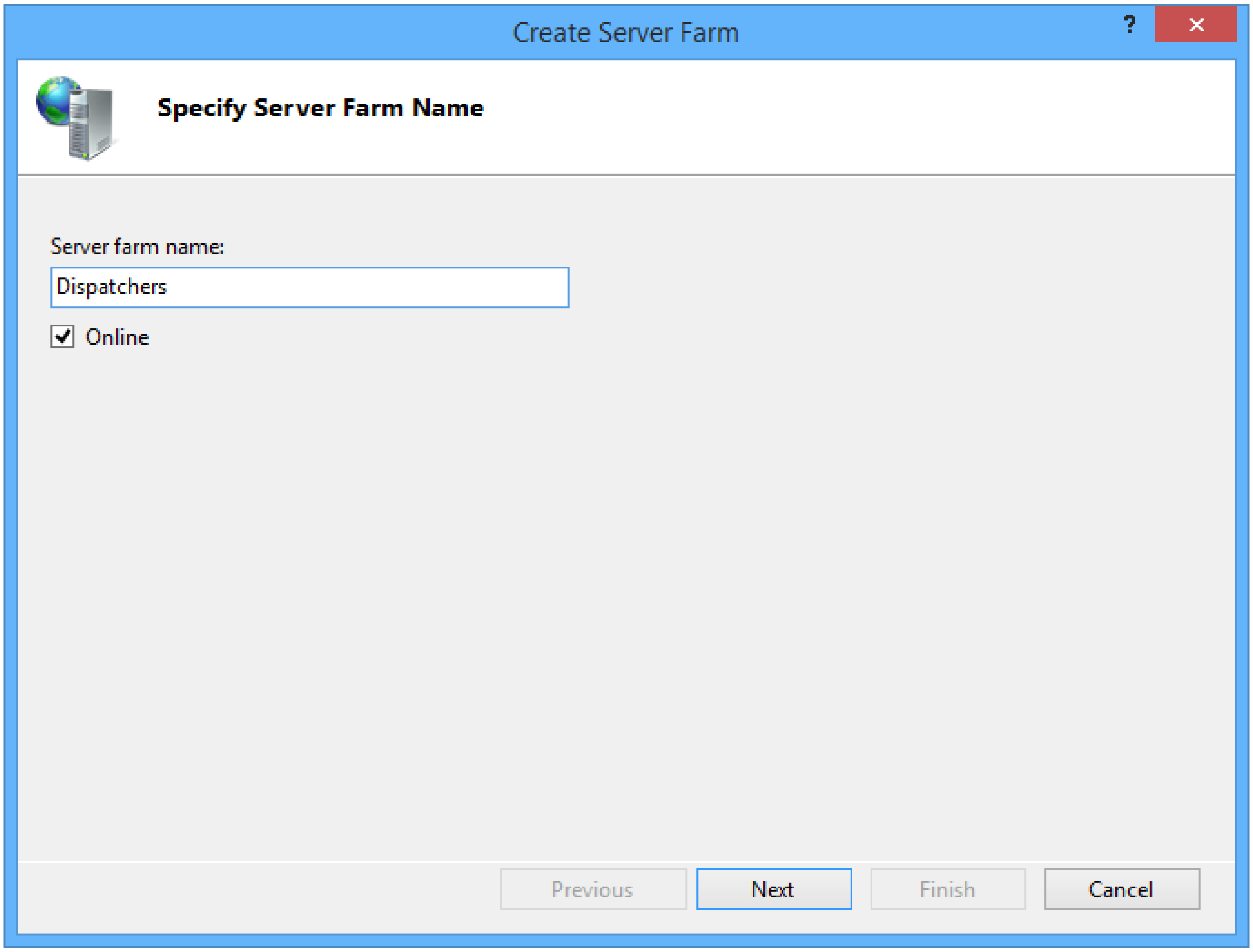
To configure Application Request Routing, follow the steps:

1. Open IIS Manager and expand the nodes in the **Connections** pane.



2. Right-click **Server Farm** and then click **Create Server Farm.** The **Create Server Farm** wizard launches.
3. On the **Specify Server Farm Name** page, enter the name of the server farm. Click **Next** to continue.

© NetIQ

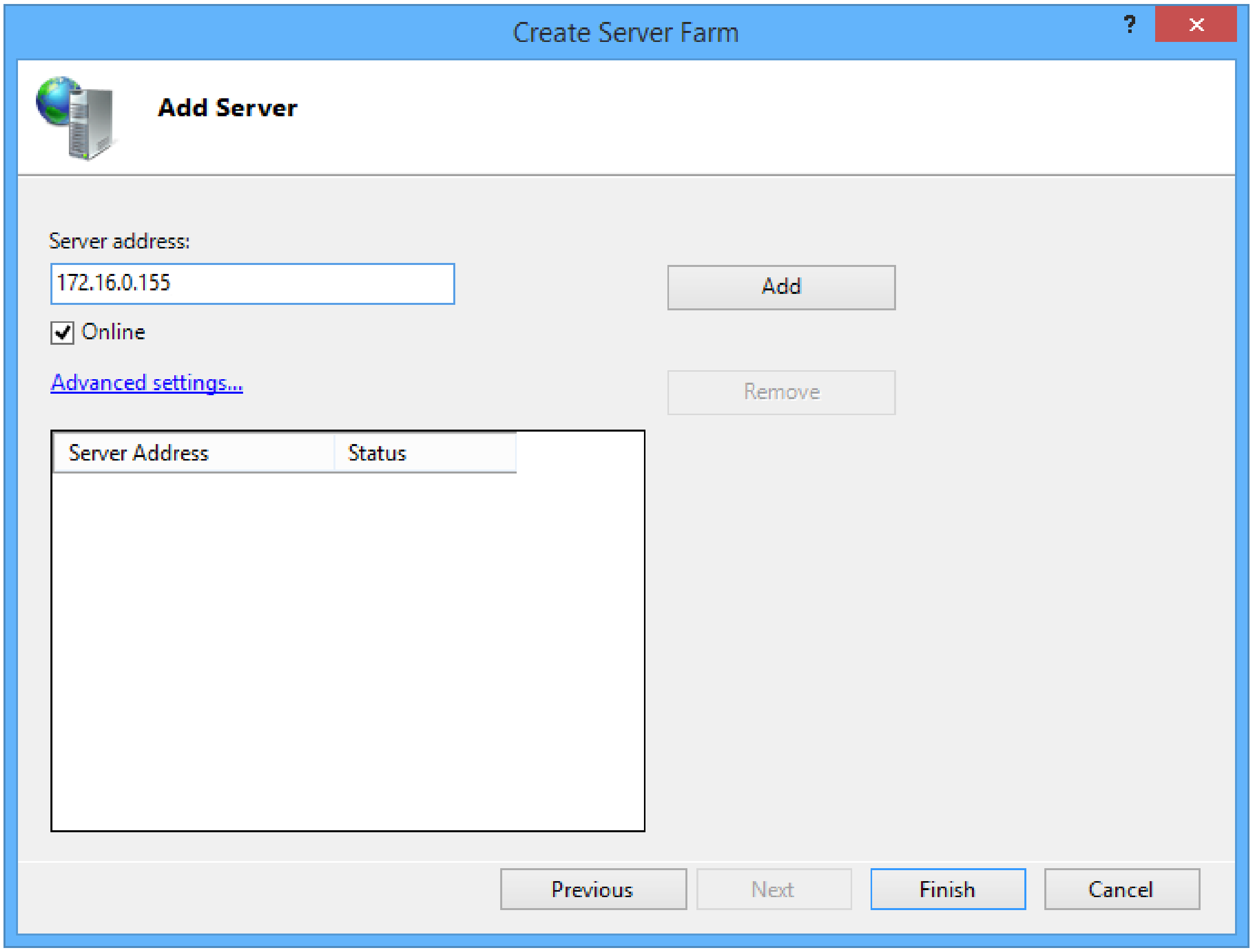


Server farm name:

Dispatchers

☑ Online

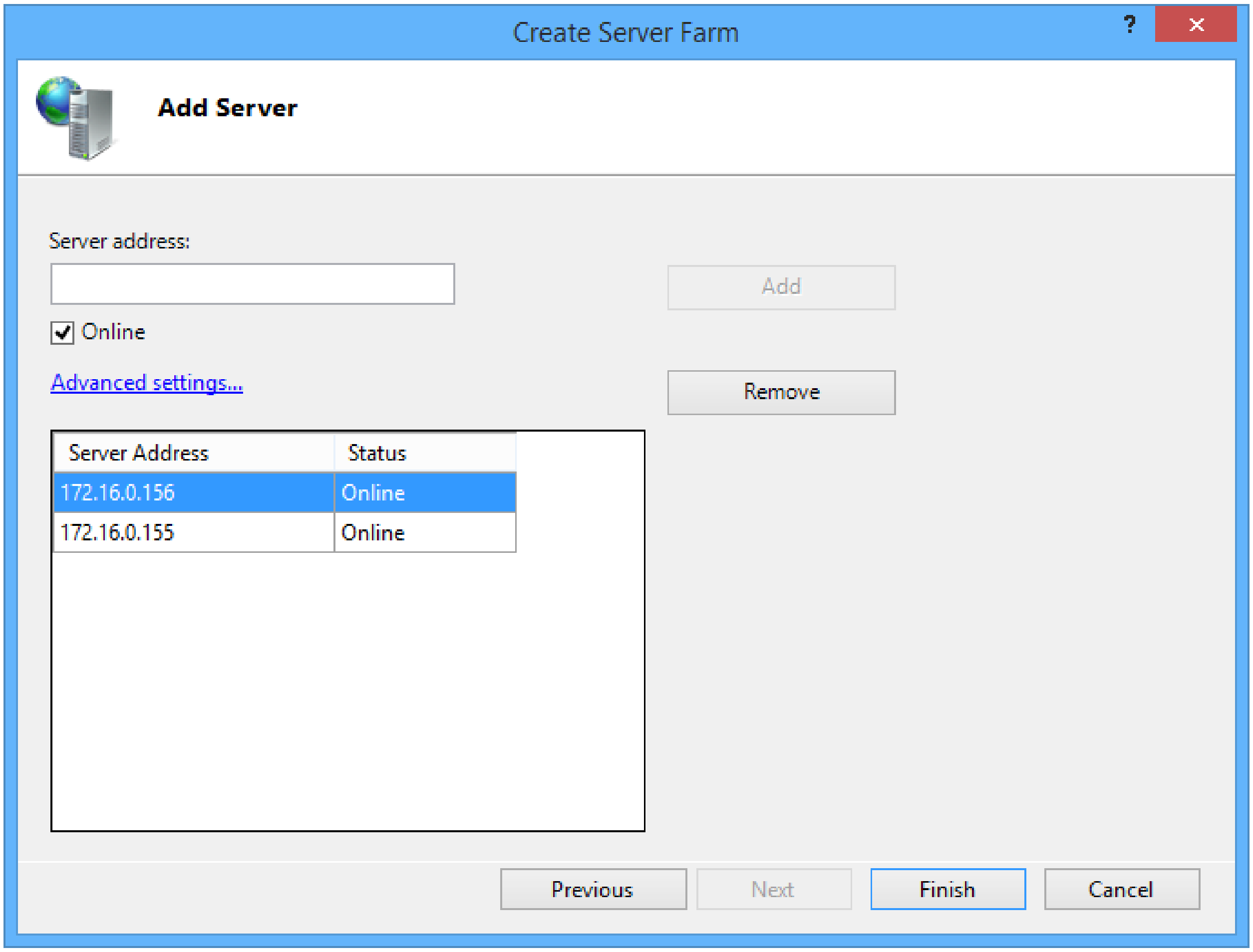
4. On the **Add Server** page, enter the dispatcher IP address. Click **Add**.



Server address: 172.16.0.155

☑ Online

5. Enter IP addresses of other dispatchers. When you are finished adding servers to the farm, click **Finish**.



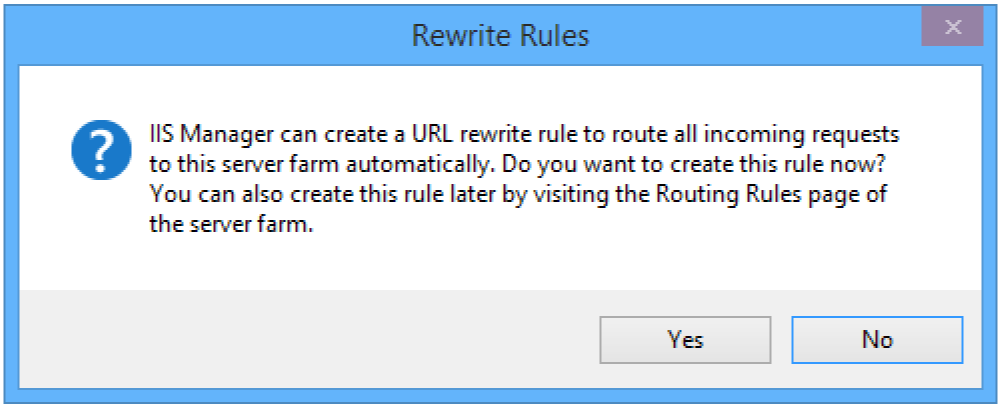
Server address:

☑ Online

Advanced settings...

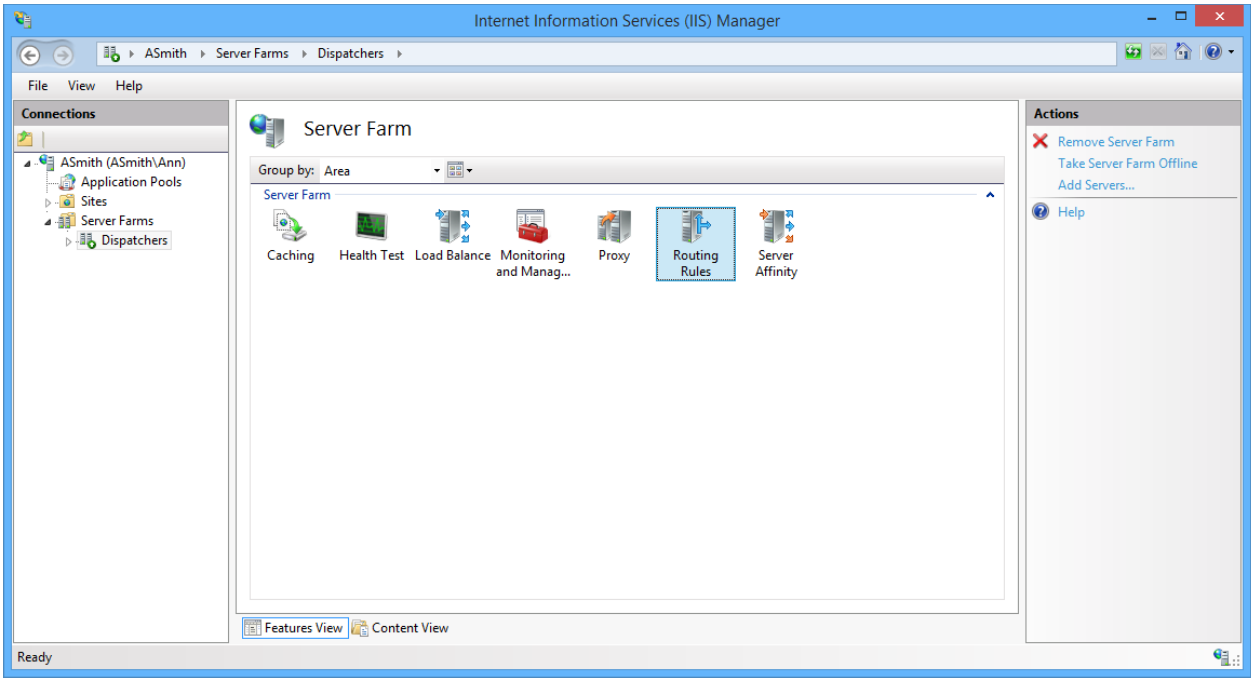
Server Address	Status
172.16.0.156	Online
172.16.0.155	Online

6. The **Rewrite Rules** dialog box is displayed. Click **No** to create the rule later.

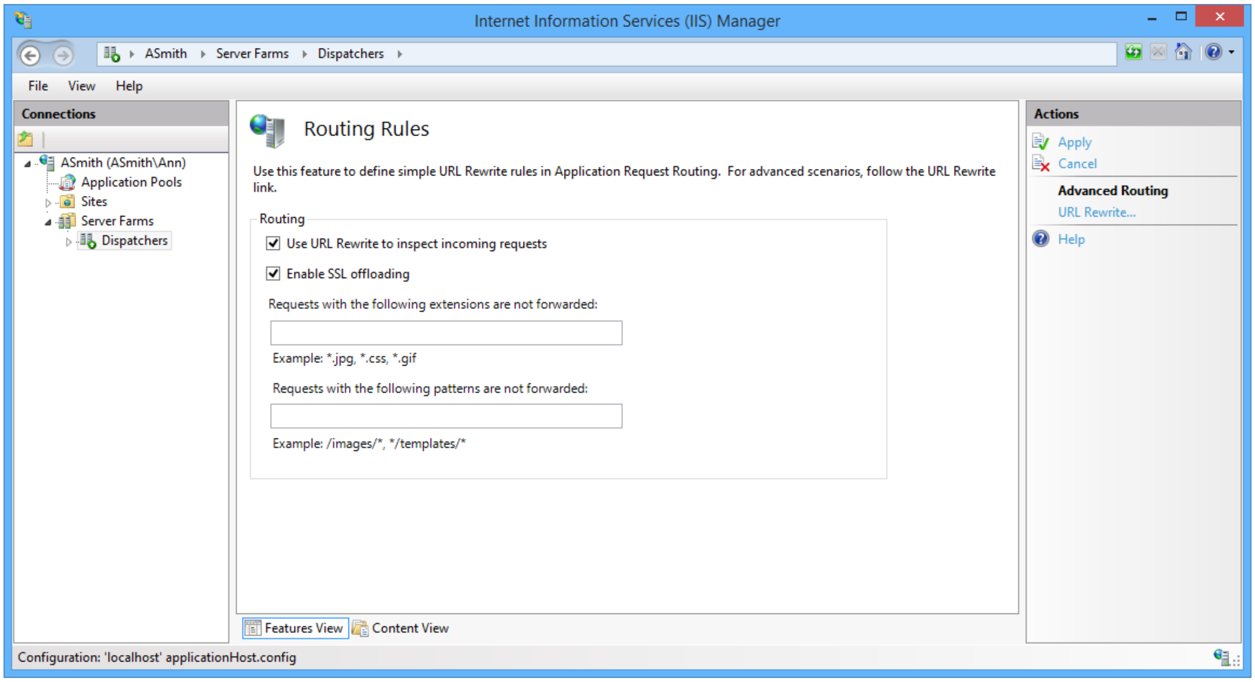


IIS Manager can create a URL rewrite rule to route all incoming requests to this server farm automatically. Do you want to create this rule now? You can also create this rule later by visiting the Routing Rules page of the server farm.

7. Expand the **Server Farms** node and click the name of the created farm. Double-click the **Routing Rules** item on the **Server Farm** page.



8. On the Routing Rules page, select the **User URL Rewrite to inspect incoming requests** and **Enable SSL offloading** checkboxes. Click **Apply** in the **Actions** pane to save changes.



Use this feature to define simple URL Rewrite rules in Application Request Routing. For advanced scenarios, follow the URL Rewrite link.

Routing

☑ Use URL Rewrite to inspect incoming requests

☑ Enable SSL offloading

Requests with the following extensions are not forwarded:

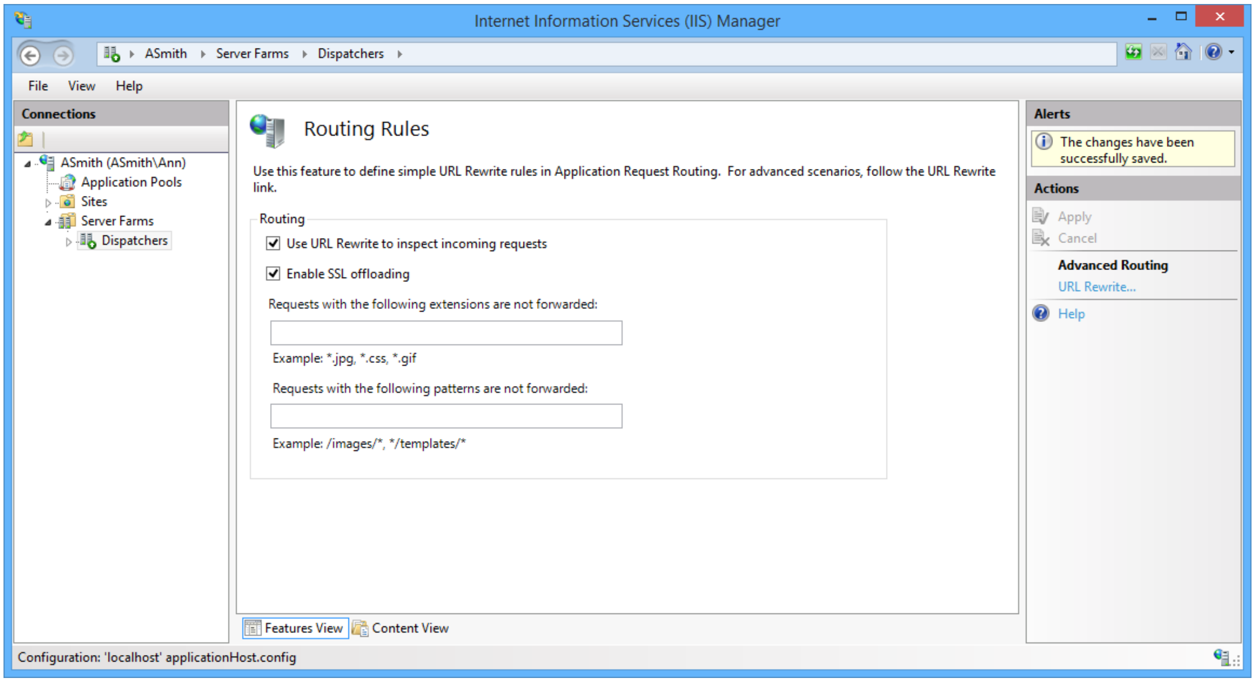
Example: *.jpg, *.css, *.gif

Requests with the following patterns are not forwarded:

Example: /images/*, */templates/*

Configuration: 'localhost' applicationHost.config

9. After the saving the changes, click **URL Rewrite** in the **Advanced Routing** subsection of the **Actions** pane.



Routing Rules

Use this feature to define simple URL Rewrite rules in Application Request Routing. For advanced scenarios, follow the URL Rewrite link.

Routing

☑ Use URL Rewrite to inspect incoming requests

☑ Enable SSL offloading

Requests with the following extensions are not forwarded:

Example: *.jpg, *.css, *.gif

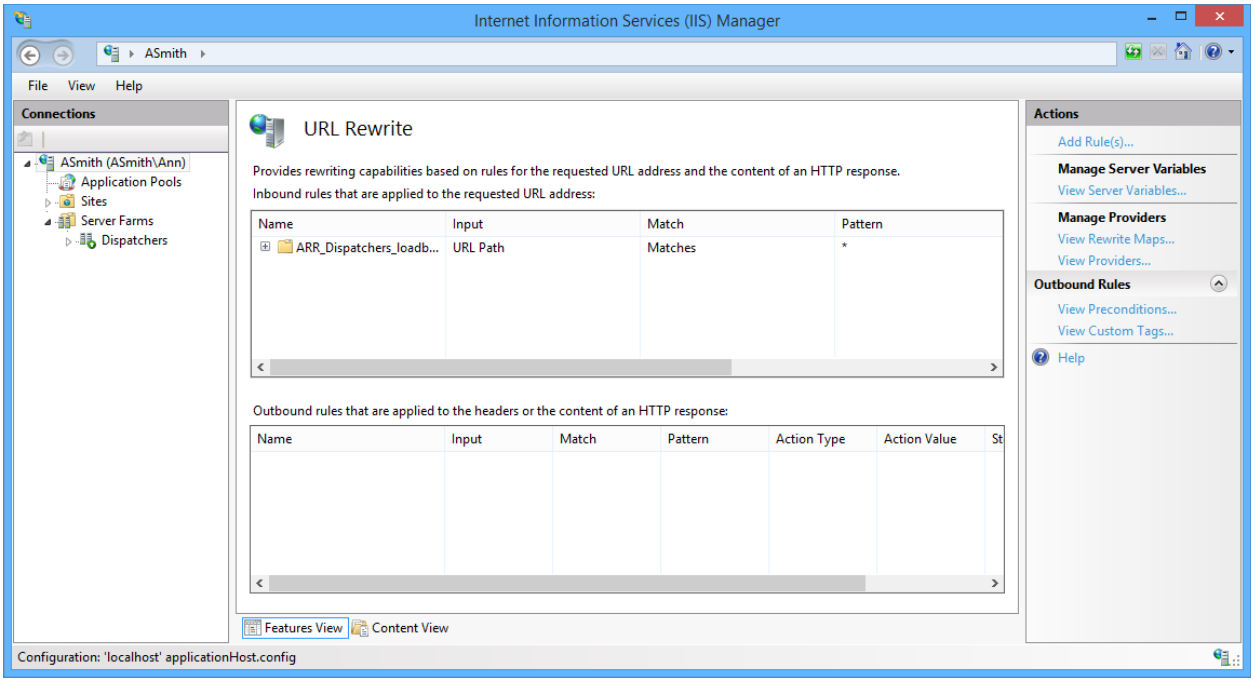
Requests with the following patterns are not forwarded:

Example: /images/*, */templates/*

The changes have been successfully saved.

Configuration: 'localhost' applicationHost.config

10. On the **URL Rewrite** page, click **Add Rule(s)** in the **Actions** pane.



URL Rewrite

Provides rewriting capabilities based on rules for the requested URL address and the content of an HTTP response.

Inbound rules that are applied to the requested URL address:

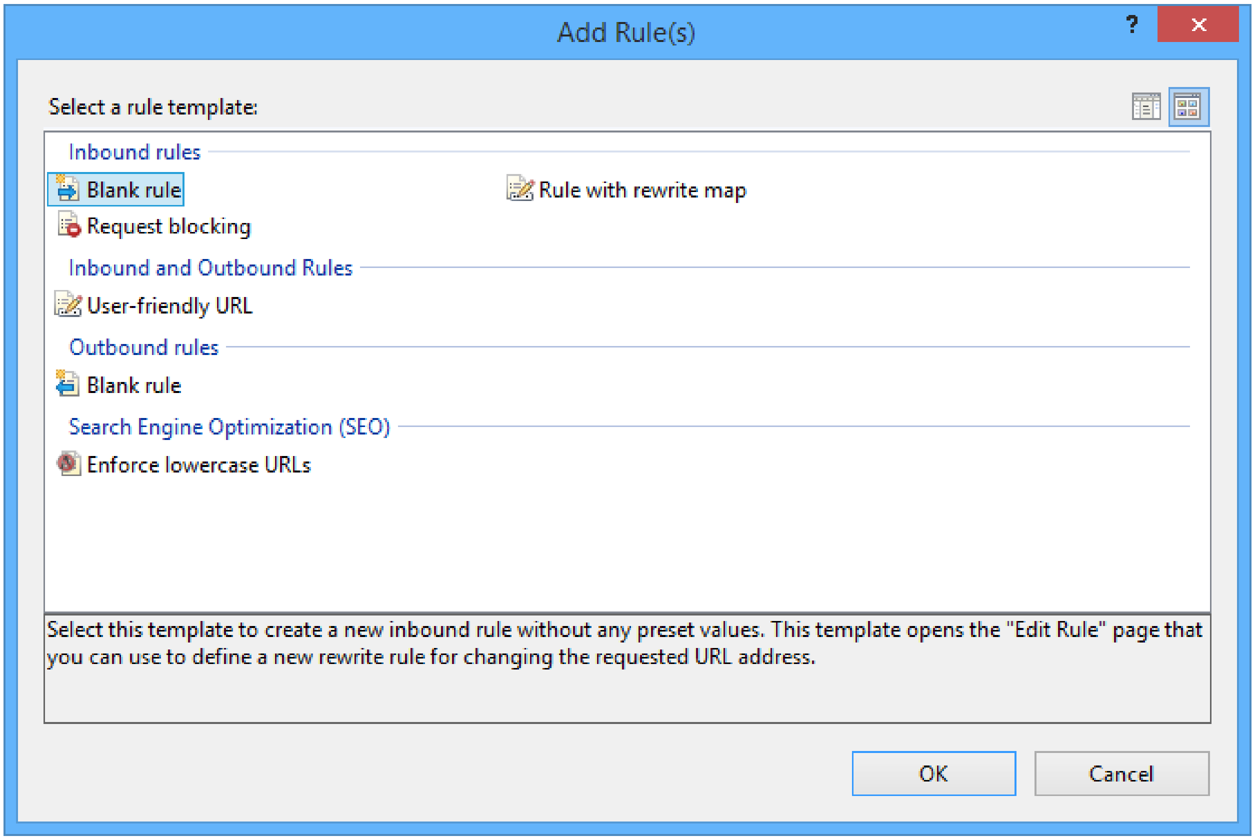
Name	Input	Match	Pattern
ARR_Dispatchers_loadb...	URL Path	Matches	*

Outbound rules that are applied to the headers or the content of an HTTP response:

Name	Input	Match	Pattern	Action Type	Action Value	St

Configuration: 'localhost' applicationHost.config

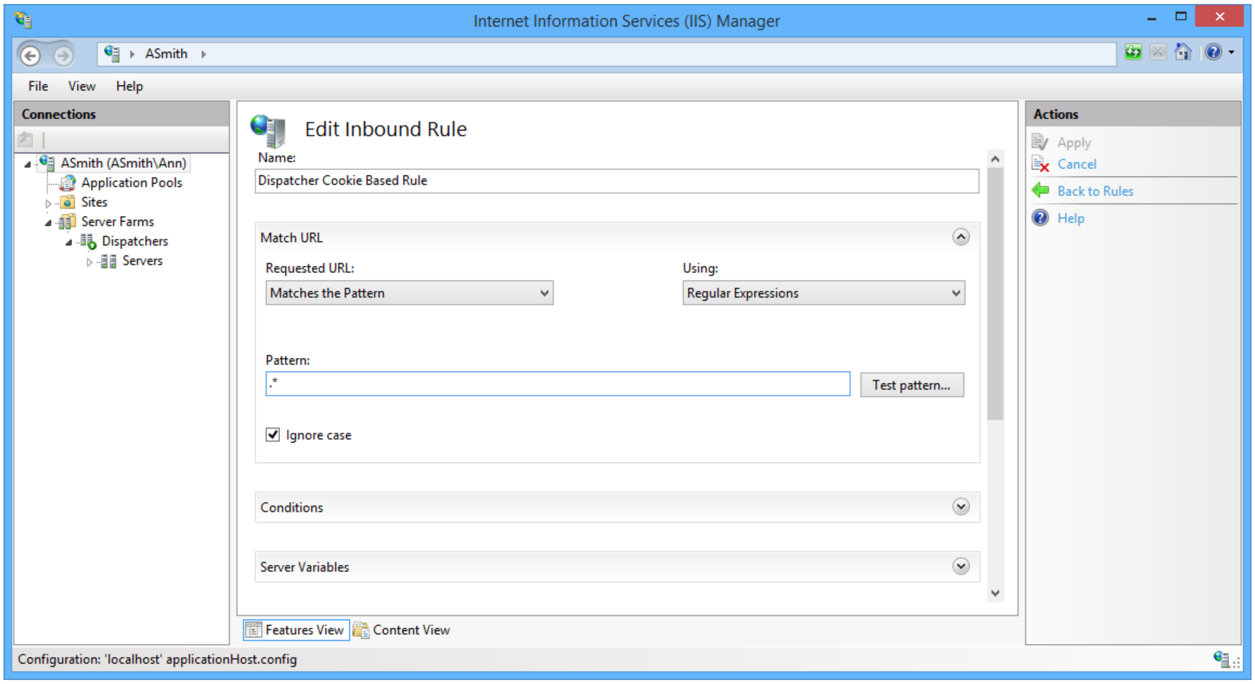
11. In the **Add Rule(s)** window, select **Blank rule** to create a new inbound rule without any preset values. Click **OK** to continue.



Select this template to create a new inbound rule without any preset values. This template opens the "Edit Rule" page that you can use to define a new rewrite rule for changing the requested URL address.

12. On the **Edit Inbound Rule** page, specify the name of the new rule.
13. In the **Match URL** group box, do the following:
    - Select **Matches the Pattern** from the **Requested URL** dropdown.
    - Select **Regular Expressions** from the **Using** dropdown.
    - Specify the **.*** value in the **Pattern** text field.
    - Select the **Ignore case** checkbox.

© *NetIQ*



Name: Dispatcher Cookie Based Rule

Requested URL: Matches the Pattern

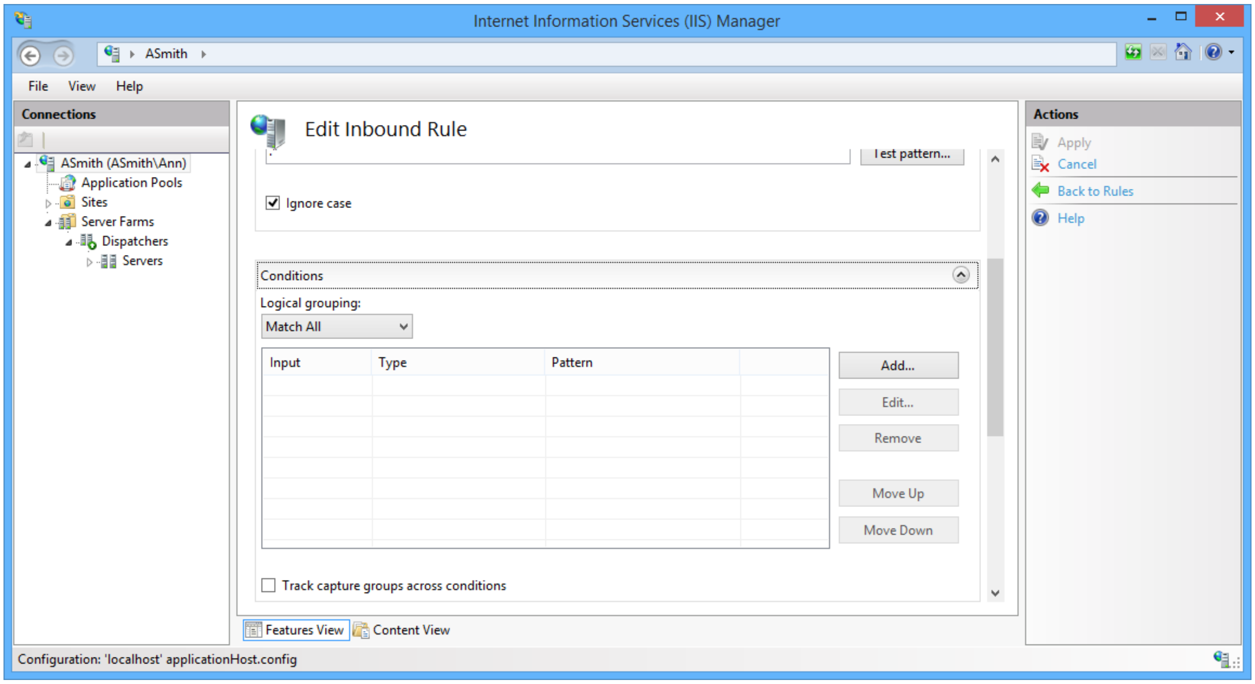
Using: Regular Expressions

Pattern: .*

☑ Ignore case

Configuration: 'localhost' applicationHost.config

14. In the **Conditions** group box, click the **Add** button to add a condition.



☑ Ignore case

Logical grouping: Match All

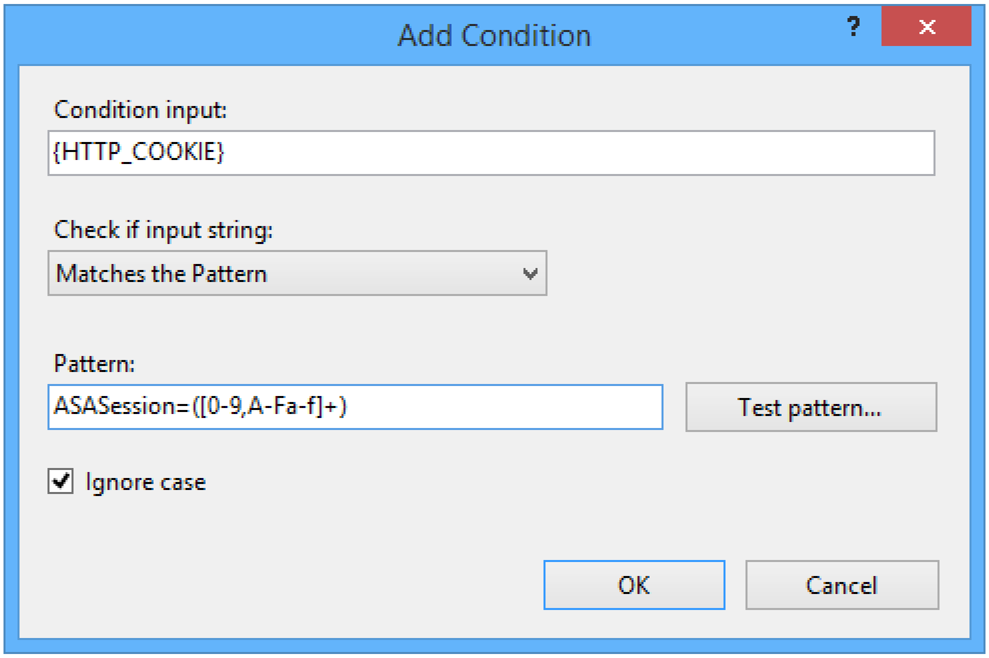
Input	Type	Pattern

☐ Track capture groups across conditions

Configuration: 'localhost' applicationHost.config

15. In the **Add Condition** window, perform the following actions:
    - Specify the **{HTTP_COOKIE}** value in the **Condition input** text field.
    - Select **Matches the Pattern** from the **Check if input string** dropdown.

- Specify the **ASASession=([0-9,A-Fa-f]+)** value in the **Pattern** text field.
- Click **OK** to continue.



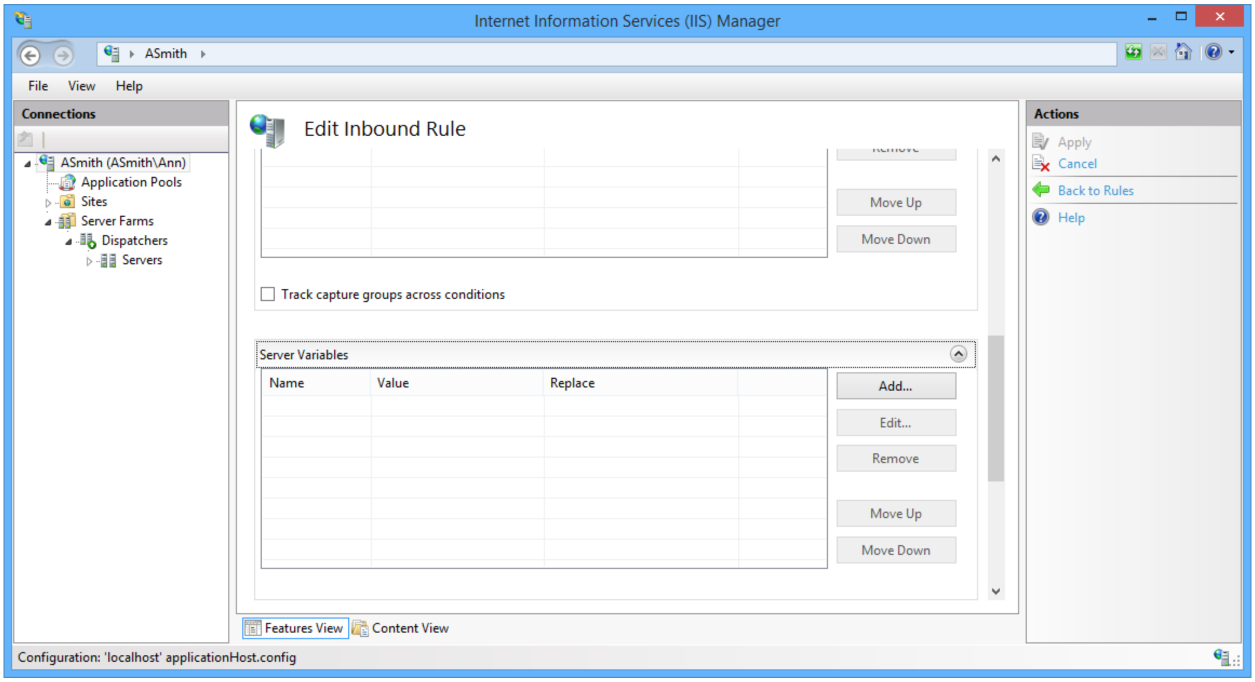
Condition input:
{HTTP_COOKIE}

Check if input string:
Matches the Pattern

Pattern:
ASASession=([0-9,A-Fa-f]+)

☑ Ignore case

16. In the **Server Variables** group box, click the **Add** to set server variable.

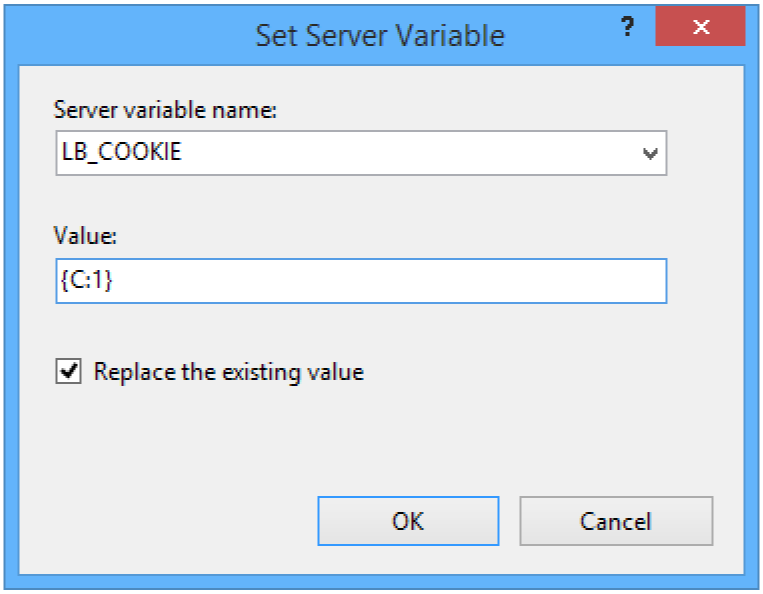


□ Track capture groups across conditions

Name	Value	Replace

Configuration: 'localhost' applicationHost.config

17. In the **Set Server Variable** window, perform the following actions:
    - Specify the **LB_COOKIE** value in the **Server variable name** text field.
    - Specify the **{C:1}** value in the **Value** text field.
    - Click **OK** to continue.

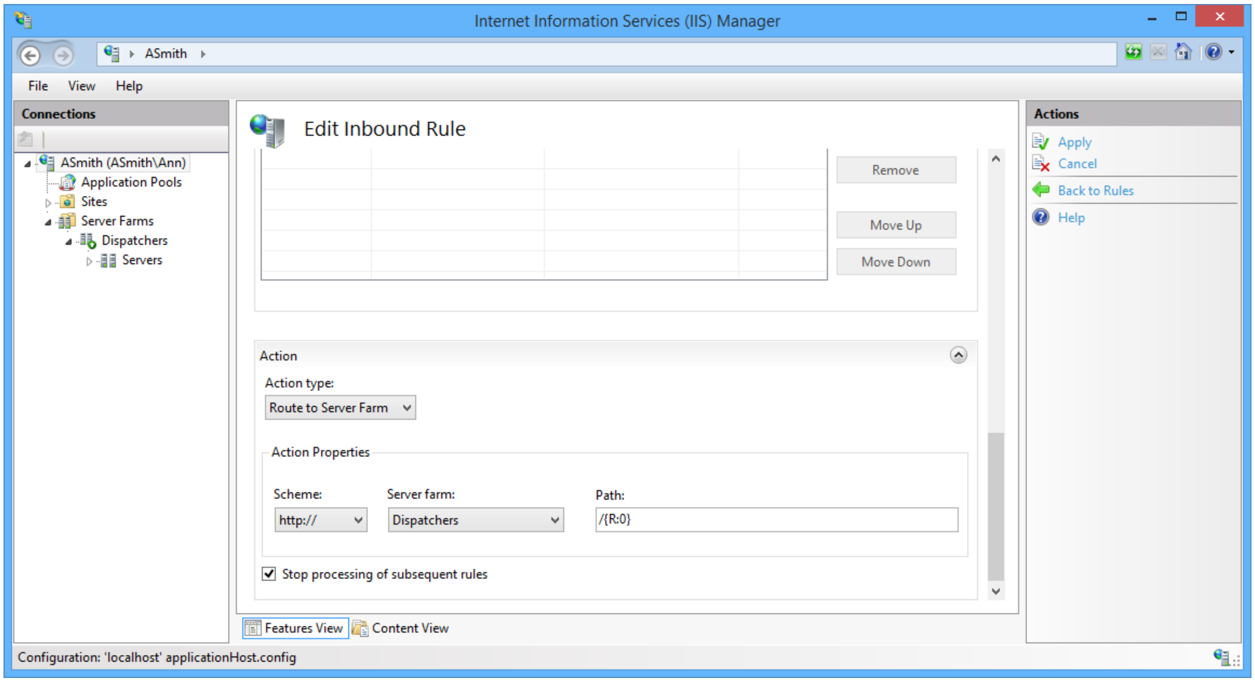


Server variable name: LB_COOKIE

Value: {C:1}

☑ Replace the existing value

18. In the **Action** group box, perform the following actions:
    - Select **Route to Server Farm** from the **Action type** dropdown.
    - Select the **Stop processing of subsequent rules** checkbox.
    - Click **Apply** in the **Actions** pane to save changes.



Action

Action type: Route to Server Farm

Action Properties

Scheme: http://

Server farm: Dispatchers

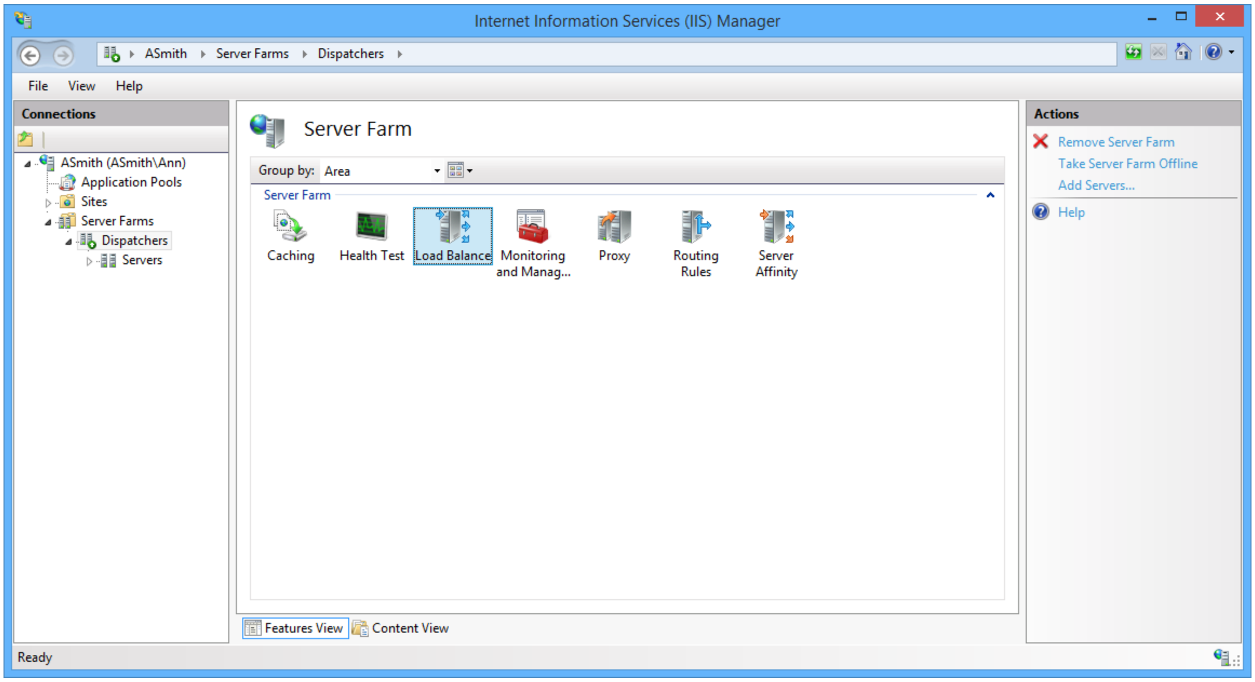
Path: /{R:0}

☑ Stop processing of subsequent rules

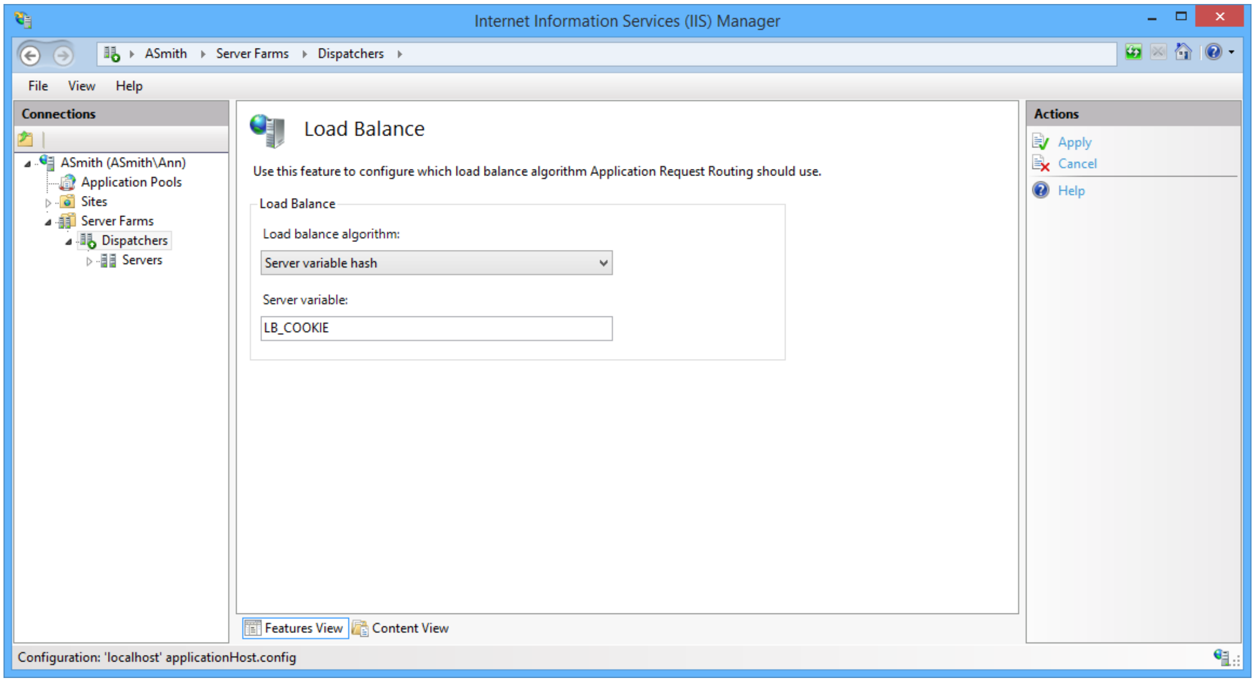
Configuration: 'localhost' applicationHost.config

© NetIQ

19. Click the name of the farm in the **Connections** pane.
20. On the **Server Farm** page, double-click the **Load Balance** item.



21. On the **Load Balance** page, perform the following actions:
    - Select **Server variable hash** from the **Load balance algorithm** dropdown.
    - Specify the **LB_COOKIE** value in the **Server variable** text field.
    - Click **Apply** in the **Actions** pane to save changes.



Load Balance

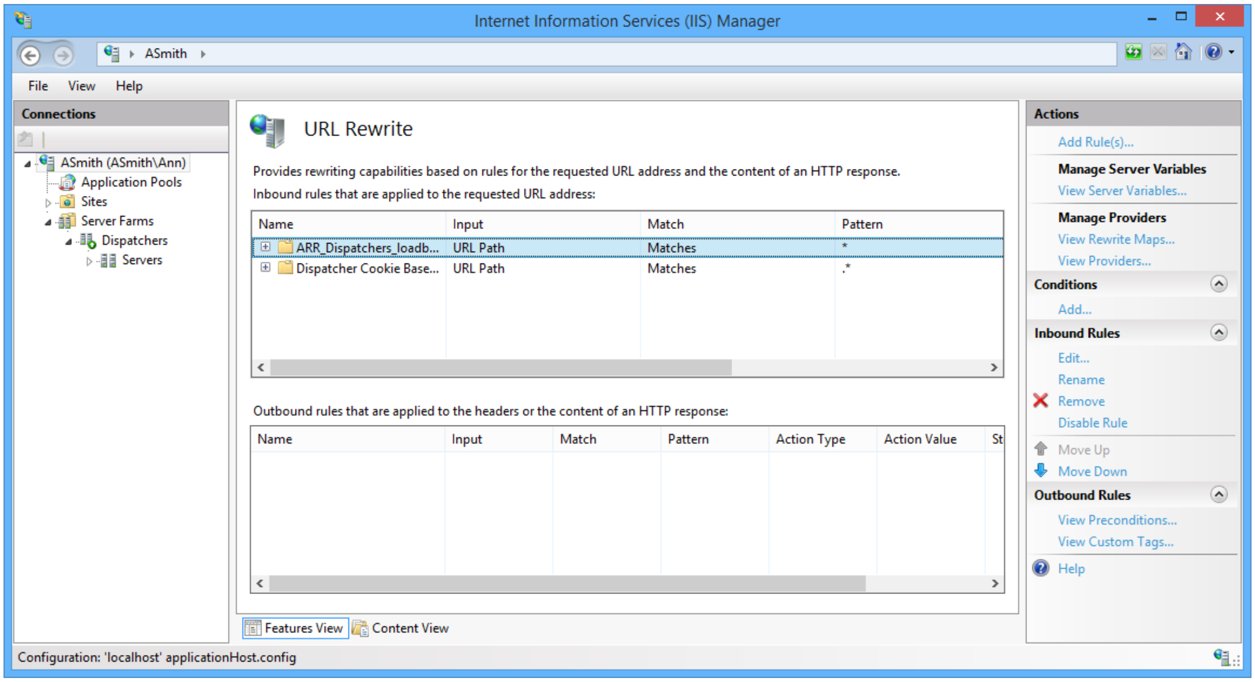
Use this feature to configure which load balance algorithm Application Request Routing should use.

Load Balance

Load balance algorithm: Server variable hash

Server variable: LB_COOKIE

22. Click the name of the server farm and double-click the **Routing Rules** item.
23. On the **Routing Rules** page, click **URL Rewrite** in the **Advanced Routing** subsection of the **Actions** pane.
24. Select the default rule (ARR_Dispatchers_loadbalance). Click **Move Down** in the **Inbound Rules** subsections of the **Actions** pane.



URL Rewrite

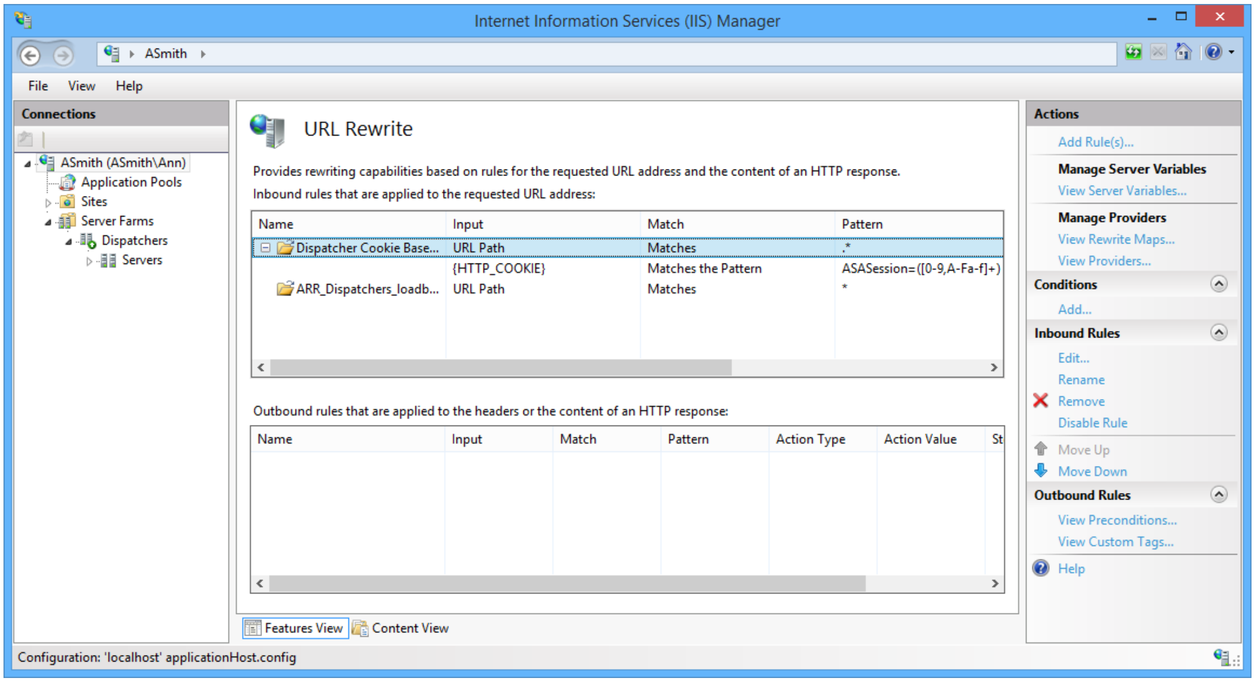
Provides rewriting capabilities based on rules for the requested URL address and the content of an HTTP response.

Inbound rules that are applied to the requested URL address:

Name	Input	Match	Pattern
ARR_Dispatchers_loadb...	URL Path	Matches	*
Dispatcher Cookie Base...	URL Path	Matches	.*

Outbound rules that are applied to the headers or the content of an HTTP response:

Name	Input	Match	Pattern	Action Type	Action Value	St



URL Rewrite

Provides rewriting capabilities based on rules for the requested URL address and the content of an HTTP response.

Inbound rules that are applied to the requested URL address:

Name	Input	Match	Pattern
Dispatcher Cookie Base...	URL Path	Matches	.*
	{HTTP_COOKIE}	Matches the Pattern	ASASession=([0-9,A-Fa-f]+)
ARR_Dispatchers_loadb...	URL Path	Matches	*

Outbound rules that are applied to the headers or the content of an HTTP response:

Name	Input	Match	Pattern	Action Type	Action Value	St

Configuration: 'localhost' applicationHost.config

All ARR/IIS parameters are stored in the **applicationHOST.config** file. The file is located in **C:\Windows\System32\inetsrv\config\**.

Removing Smartphone Authentication Dispatcher

In this chapter:

- Microsoft Windows 7/Microsoft Windows Server 2008 R2
- Microsoft Windows Server 2012

Microsoft Windows 7/Microsoft Windows Server 2008 R2

1. In the **Start** menu, select **Control panel** and then double-click **Programs and Features**.
2. Select **Smartphone Authentication Dispatcher** and click **Uninstall**.
3. Confirm the removal.
4. Wait a few seconds until the removal is completed.

Microsoft Windows Server 2012

1. In the **Search** menu, select **Apps > Control Panel > Programs > Programs and Features**.
2. Select **Smartphone Authentication Dispatcher** and click **Uninstall**.
3. Confirm the removal.
4. Wait a few seconds until the removal is completed.

Migrating Smartphone Authentication Dispatcher

In case of migrating Smartphone Authentication Dispatcher from one server to another, it is required to save its data storage file. The **SaDispatcherDb-***.sdf** data storage file is located in **C:\ProgramData** folder on the server with the installed Smartphone Authentication Dispatcher. Copy this file to the **C:\Program Data** folder on the new server and run Smartphone Authentication Dispatcher.

Index

© *NetIQ*