Web Sockets

Version 2018

Contents

Web Sockets **4**

Introduction 4
WebSockets and IIS 4
The JadeWebSocket Initialization File 4
Exercise 1 – Enabling the Winsock Interface for IIS 6
Exercise 2 – Installing the JadeWebSocket Module 6
Exercise 3 – Enabling the JadeWebSocketModule 9
Exercise 4 – Configuring the JadeWebSockets Initialization File 9
The JadeWebSocketServer Class 10
Connecting to a JADE WebSocket Server 10
Exercise 5 – Creating a WebSocket Server Application 10
The JadeWebSocket Class 13
Exercise 6 – Responding to Messages 14
Exercise 7 – Reimplementing onOpen, onClose, and sendText 15
Exercise 8 – Sending Images over WebSockets 16

Web Sockets

Introduction

The WebSocket communications protocol facilitates real-time data streaming between a client and a server.

Use of WebSockets allows JADE server-side applications to send content to clients (whether other JADE systems or external systems) without needing to be first requested by the client. This allows messages to be passed back and forth while keeping the connection open, enabling two-way ongoing conversations between the client and server.

WebSockets and IIS

The Internet Information Server (IIS) is a Microsoft Web Server that allows a standard device (a desktop PC or a laptop) running Microsoft Windows to act as a web server. By proper configuration of IIS, any WebSocket requests that are sent to the device can be handled by a JADE database, which can respond to those requests.

By default, the WebSocket protocol is disabled in IIS. The handling of WebSocket requests must be enabled in the Windows Control Panel windows features so that IIS can then forward the requests to the JADE database.

When IIS is set up to respond to WebSocket requests, it must also be provided with a Native Module, which in the case of JADE, is a link to the **JadeWebSockets_IIS.dll** file in the **bin** directory of your JADE database.

Finally, this Native Module must be associated with the kind of request it is equipped to handle; that is, WebSocket requests (***.ws**).

The JadeWebSocket Initialization File

The **JadeWebSockets_IIS.ini** file has the following sections.

- [JadeWebSocket Rules], which is used for establishing mappings between URIs and specific hosts and port numbers.
- [JadeWebSocket Logging], which controls logging by the **JadeWebSockets_IIS.dll** library file.

The [JadeWebSocket Rules] section must have at least one mapping rule and can have as many mapping rules as required. Mapping rules have the following format.

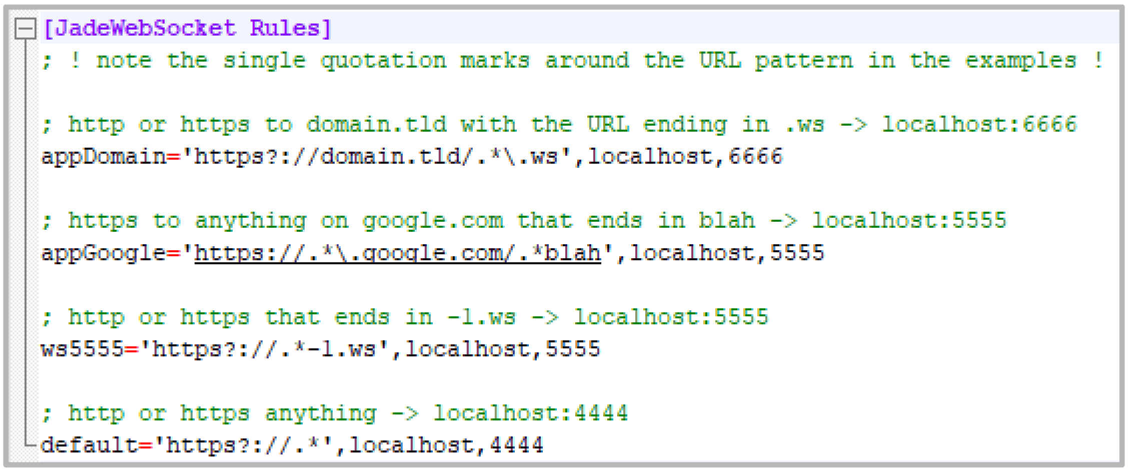
RuleName=URL-pattern,host,port

In this format:

- ***RuleName*** is any unique name that identifies the rule.
- ***URL-pattern*** is a regular expression. All URLs that match the pattern are mapped to the host and port.

- ***host*** is a network-resolvable host name or IP address for the machine that hosts the JADE WebSocket server.
- ***port*** is the port number of the TCP port being handled by the JADE WebSocket server.

When attempting to match a URL to a host and port, the rules are tried in the order in which they appear in the **JadeWebSockets_IIS.ini** file. As such, if two rules can handle the same URL, the first one defined is used. For example, consider the following set of rules. (Lines beginning with **;** are comments.)



```
[JadeWebSocket Rules]
; ! note the single quotation marks around the URL pattern in the examples !

; http or https to domain.tld with the URL ending in .ws -> localhost:6666
appDomain='https?://domain.tld/.*\.ws',localhost,6666

; https to anything on google.com that ends in blah -> localhost:5555
appGoogle='https://.*\.google.com/.*blah',localhost,5555

; http or https that ends in -1.ws -> localhost:5555
ws5555='https?://.*-1.ws',localhost,5555

; http or https anything -> localhost:4444
default='https?://.*',localhost,4444
```

In this example, **https://domain.tld/WS-1.ws** goes to **localhost:6666**, although it also matches the rule for **localhost:5555** and **localhost:4444**. This is because **localhost:6666** appears earlier in the [JadeWebSocket Rules] section of the file.

The [JadeWebSocket Logging] section controls logging by the **JadeWebSockets_IIS.dll** library file and it has the following parameters.

Parameter	Value Type	Purpose
Trace	Boolean	Controls the logging of **JadeWebSockets_IIS.dll** library activity. If **false**, no logging is performed. If **true**, logged messages acknowledge only that messages have been sent or received; not the content of those messages. The default value, if unset, is **false**.
TraceFile	String	Optional parameter that allows for the specification of a file path to override the default **jadeWebSockets_IIS<timestamp>.log** log file value.
TraceFileSize	Integer	Specifies the maximum file size, in bytes, before switching log files. The default value, if unset, is **1000000** (one million bytes).

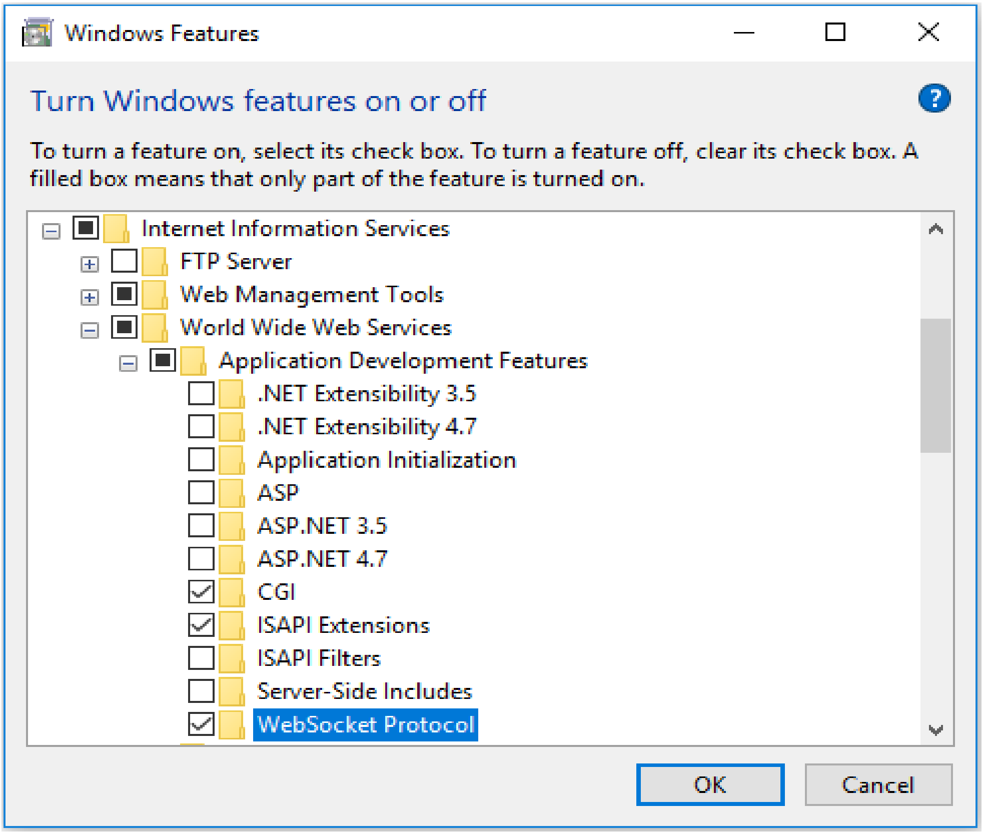
Exercise 1 – Enabling the Winsock Interface for IIS

In this exercise, you will ensure that the WebSocket protocol is enabled for IIS on your machine.

Note These instructions assume you are using Windows 10. Some details can be slightly different for earlier versions of Windows. In addition, to use WebSockets in JADE, IIS 8 or later is required.

1. Open the Windows Features window, by searching for **Turn Windows features on or off** in Windows search or finding **Turn Windows features on or off** in the panel at the left of the **Programs and Features** category of the Windows Control Panel.

2. Check the **WebSocket Protocol** check box in the expanded **Application Development Features** under **Internet Information Services**.



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

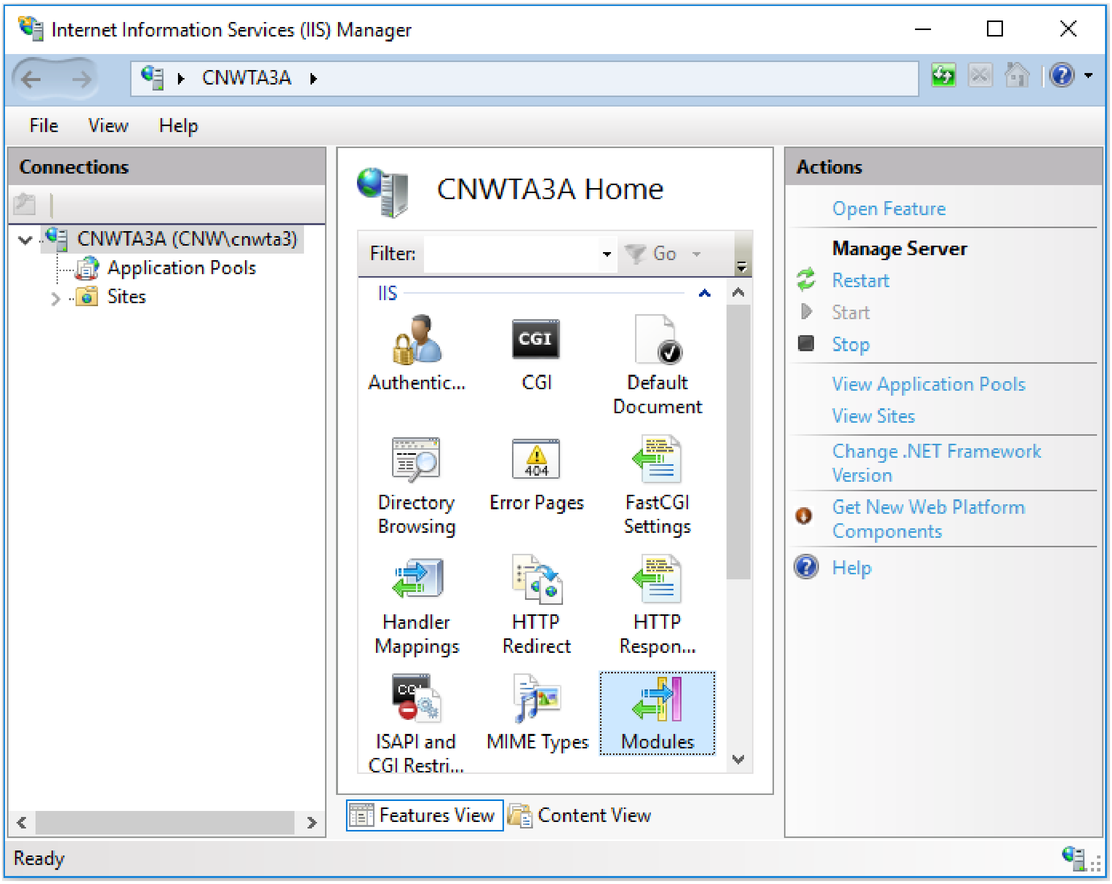
- ■ Internet Information Services
  - ☐ FTP Server
  - ■ Web Management Tools
  - ■ World Wide Web Services
    - ■ Application Development Features
      - ☐ .NET Extensibility 3.5
      - ☐ .NET Extensibility 4.7
      - ☐ Application Initialization
      - ☐ ASP
      - ☐ ASP.NET 3.5
      - ☐ ASP.NET 4.7
      - ☑ CGI
      - ☑ ISAPI Extensions
      - ☐ ISAPI Filters
      - ☐ Server-Side Includes
      - ☑ WebSocket Protocol

Exercise 2 – Installing the JadeWebSocket Module

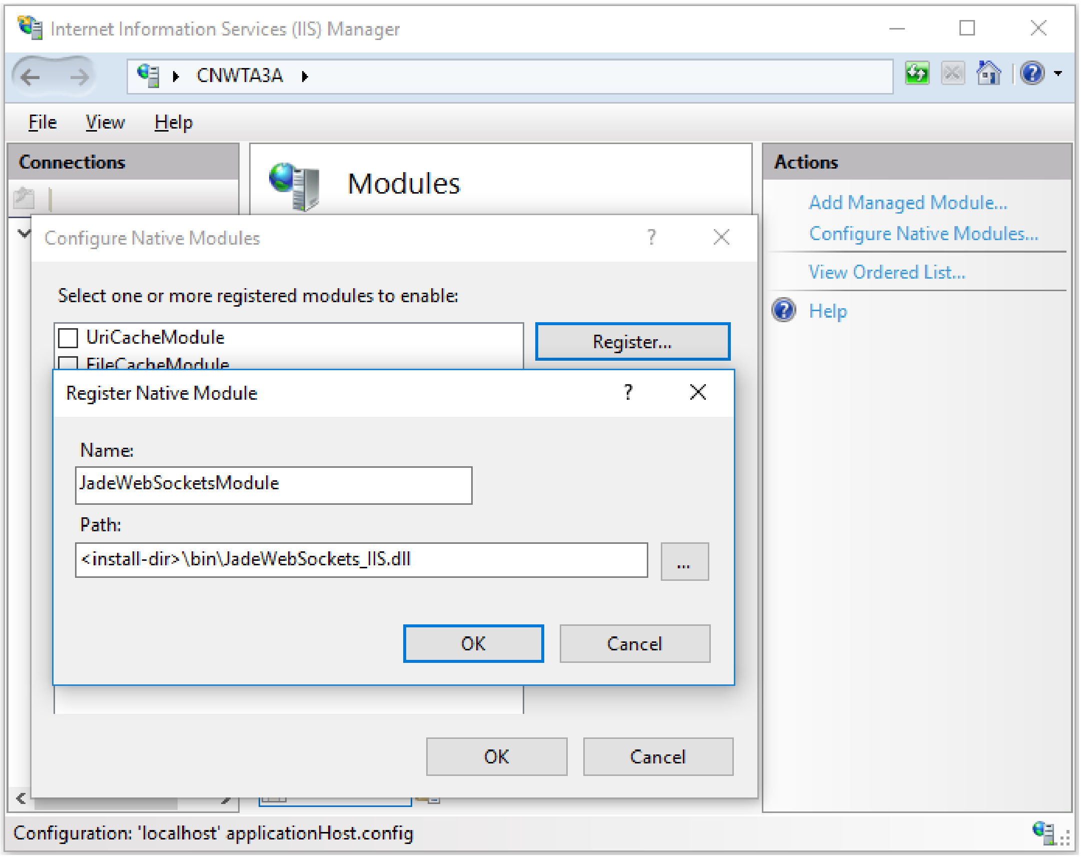
In this exercise, you will install **JadeWebSocket** as a module in IIS.

1. Open the IIS Manager by searching for **IIS** in Windows search or finding **Internet Information Services (IIS) Manager** in the Windows Control Panel **Administrative Tools** category.

2. Select your machine from the **Connections** panel (**CNWTA3A**, in the following example), then double-click **Modules** in the center panel.



3. Select **Configure Native Modules** from the **Actions** panel, click **Register**, and fill out the dialog as follows. (The **<install-dir>** value is the location of your JADE installation.)

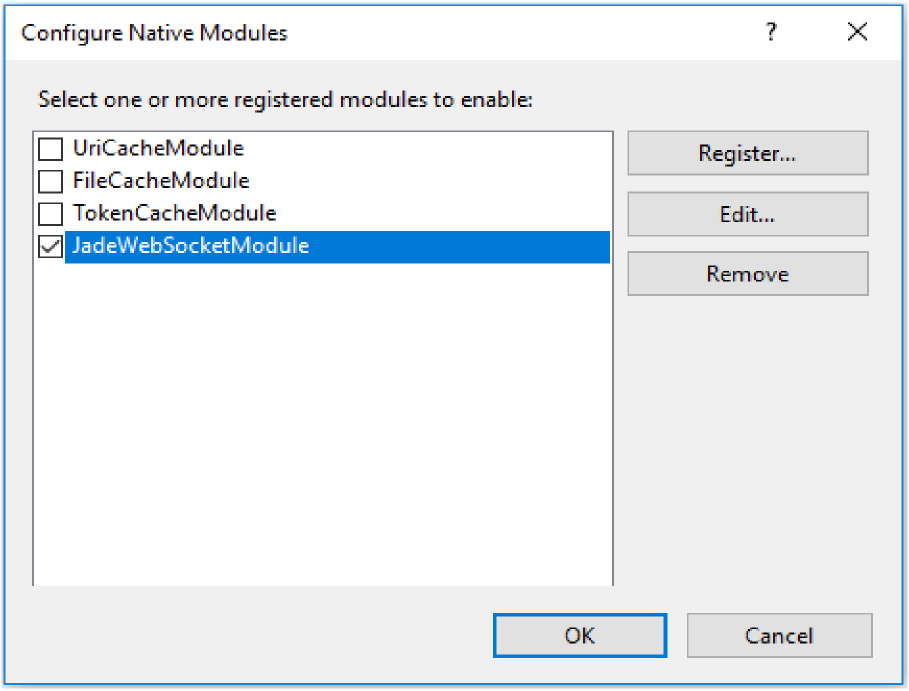


Name: JadeWebSocketsModule

Path: <install-dir>\bin\JadeWebSockets_IIS.dll

Configuration: 'localhost' applicationHost.config

4. Check the **JadeWebSocketModule** check box on the Configure Native Modules dialog and then click **OK**.

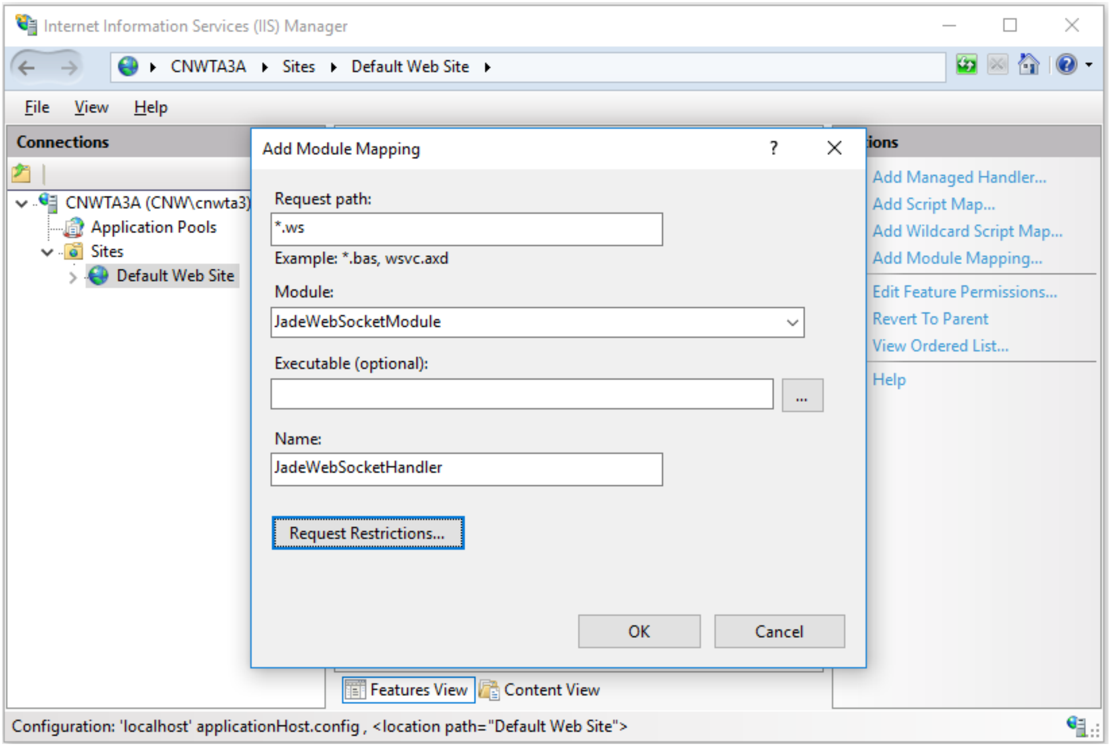


- [ ] UriCacheModule
- [ ] FileCacheModule
- [ ] TokenCacheModule
- [✓] JadeWebSocketModule

Exercise 3 – Enabling the JadeWebSocketModule

In this exercise, you will set all WebSocket requests to use the **JadeWebSocketModule** that was installed in the previous exercise.

1. In the **Connections** panel on the left, select **Default Web Site**.
2. From the middle panel, double-click **Handler Mappings**.
3. Select **Add Module Mapping** from the **Actions** panel on the right.
4. Fill out the Add Module Mapping dialog as follows and then click **Request Restrictions**.



Internet Information Services (IIS) Manager

CNWTA3A ▸ Sites ▸ Default Web Site ▸

Add Module Mapping

Request path: *.ws

Example: *.bas, wsvc.axd

Module: JadeWebSocketModule

Executable (optional):

Name: JadeWebSocketHandler

Configuration: 'localhost' applicationHost.config , <location path="Default Web Site">

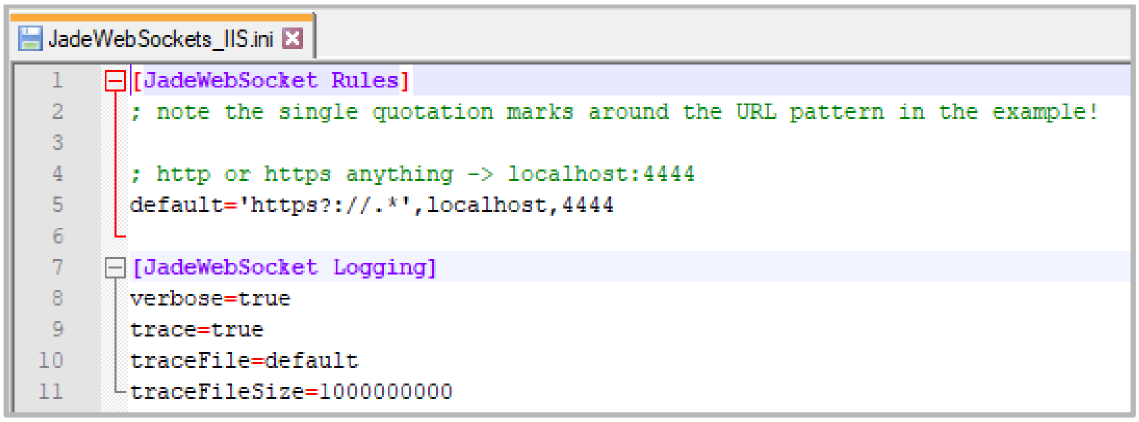
5. On the **Mapping** sheet of the Request Restrictions dialog, uncheck the **Invoke handler only if request is mapped to:** check box and then click **OK**.

Exercise 4 – Configuring the JadeWebSockets Initialization File

In this exercise, you will configure the **JadeWebSockets_IIS.ini** initialization file to map URIs to port numbers.

1. Open the **ws://localhost/WebSocketClientTest/websocket-1.ws** file, found in **<install-dir>\bin_JadeWebSockets_IIS\ini**.

2. Set the following parameter values in the **JadeWebSockets_IIS.ini** file.



```
[JadeWebSocket Rules]
; note the single quotation marks around the URL pattern in the example!

; http or https anything -> localhost:4444
default='https?://.*',localhost,4444

[JadeWebSocket Logging]
verbose=true
trace=true
traceFile=default
traceFileSize=1000000000
```

The JadeWebSocketServer Class

The **JadeWebSocketServer** class of **RootSchema** handles all incoming TCP/IP connections (for example, WebSocket) coming over IIS from a specific interface and TCP port.

The **JadeWebSocketServer** class provides the following methods.

Method	Description
getWebSocket	Takes a WebSocket ID parameter and returns the **JadeWebSocket** with that ID.
run	Takes a **localInterface**, a **tcpPort**, and a **webSocketClass** parameter and begins assigning all connections on the specified **tcpPort** that meet the specified **localInterface** to the specified **webSocketClass**. This method continues executing until the **stop** method is called.
stop	Signals any active **run** methods to stop listening on the local interface and TCP/IP port, and terminates all WebSocket connections.

Connecting to a JADE WebSocket Server

When a JADE WebSocket server is set up and accepting WebSocket connections, external applications can send messages to the JADE database over the WebSocket protocol. For example, a web site could access the database with some JavaScript code.

A full explanation of how to implement WebSocket clients in JavaScript or other technologies is beyond the scope of this module. However, the **WebSocketImage.html** and **WebSocketTest.html** files, found at https://www.jadeworld.com/developer-center/learn-jade/, provide working demonstrations.

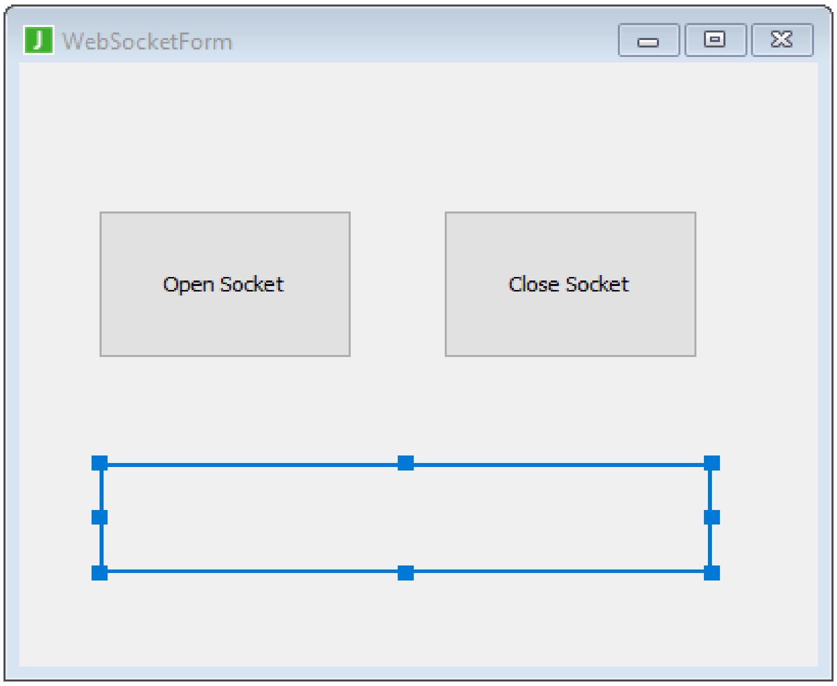
Exercise 5 – Creating a WebSocket Server Application

In this exercise, you will create a simple JADE application that can accept a connection from an external WebSocket client.

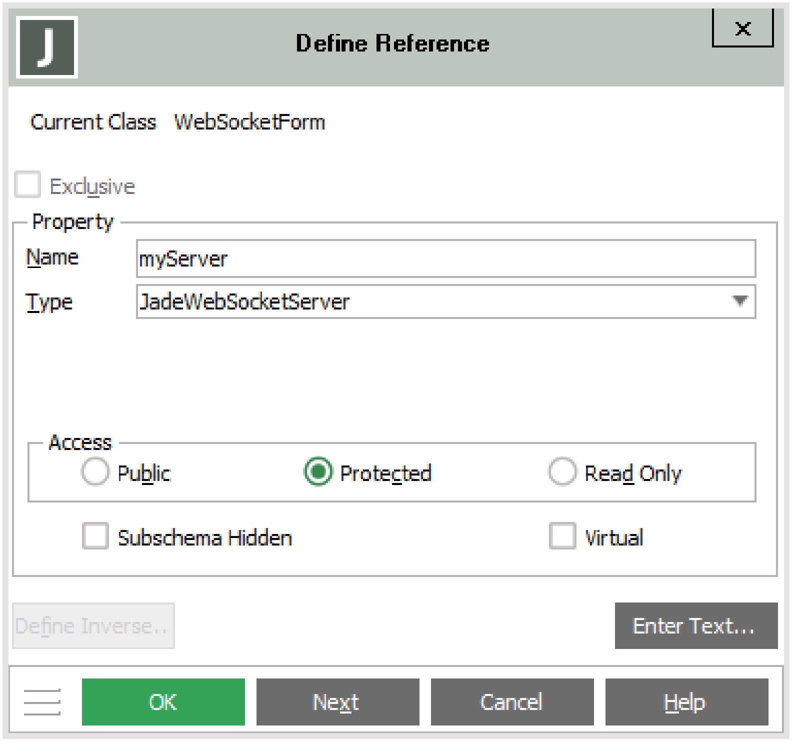
For this exercise, you will verify only that you can open a connection, although in future exercises you will add the ability to send and receive messages over WebSockets.

1. Create a schema called **WebSocketSchema** and from this schema, open the JADE Painter.

2. Create a form called **WebSocketForm**, with two buttons called **btnOpen** and **btnClosed**, and a label called **lblState**, as follows.



3. Create a reference called **myServer** of type **JadeWebSocketServer** in the **WebSocketForm** class, as follows.



Current Class WebSocketForm

[ ] Exclusive

Property

Name: myServer

Type: JadeWebSocketServer

Access: Public [✓ Protected] Read Only

[ ] Subschema Hidden [ ] Virtual

4. Code the **btnOpen_click** method as follows.

```
btnOpen_click(btn: Button input) updating;

begin
    create myServer transient;
    self.lblState.caption := "Socket is OPEN";
    self.myServer.run(null, 4444, JadeWebSocket);

epilog
    delete myServer;
    self.lblState.caption := "Socket is CLOSED";
end;
```

5. Code the **btnClose_click** method as follows.

```
btnClose_click(btn: Button input) updating;

begin
    if not myServer = null then
        self.myServer.stop();
    endif;
end;
```

6. Code the **unload** method (under **Form Events**) as follows.

```
unload() updating;

begin
    if not myServer = null then
        self.myServer.stop();
    endif;
end;
```

   This ensures that the server is left in a closed state if the form is closed without closing the WebSocket connection.

7. Create a JadeScript method called **startForm** and code it as follows.

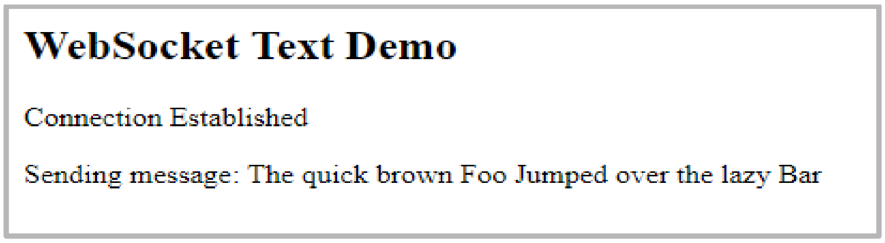
```
startForm();

vars
    wsForm : WebSocketForm;

begin
    create wsForm;
    wsForm.showModal();
end;
```

8. Run the JadeScript **startForm** method and then click **Open Socket**.

9. Run the **WebSocketTest.html** file, found at https://www.jadeworld.com/developer-center/learn-jade/.



WebSocket Text Demo

Connection Established

Sending message: The quick brown Foo Jumped over the lazy Bar

As the WebSocket server is currently using the default implementation, it will not respond to the message. You should, however, see the WebSocket client successfully open a connection and send the message.

The JadeWebSocket Class

The **JadeWebSocket** class is the base class for handling a WebSocket connection. It differs from the **JadeWebSocketServer** class in that the **JadeWebSocketServer** class listens on a port and forwards connections to a **JadeWebSocket** subclass, which then can process or send messages.

The **JadeWebSocket** base class provides the following methods, which can be reimplemented on user-defined subclasses.

Method	Description
onOpen	Automatically called when the WebSocket is opened from the client side of the connection. It receives one parameter, **fullUrl**, which is the full URL of the WebSocket that was used by the WebSocket initiating client.
onMsg	Automatically called when the WebSocket receives a message from the client side of the connection. It receives three parameters: the binary message that was sent, a boolean representing whether that message was UTF8-encoded, and a boolean representing whether the message is the final fragment; that is, if **false**, there is more message to come.
onClose	Automatically called when the client closes the WebSocket connection. This method receives no parameters.
send	Sends a binary message to the client via the WebSocket connection and requires two parameters: the binary message to send to the client, and a boolean representing whether this is the final fragment or whether there is more message to send.
sendText	Like the **send** method, except the message is sent as a UTF8 string rather than as a binary. As with the **send** method, it requires two parameters. However, the **msg** parameter must be of type UTF8 rather than binary.

As seen in the previous exercise, if the **JadeWebSocketServer** class **run** method is given the base class, it accepts connections but does not reply to any messages. This is because the default implementation simply passes without action when any of the methods in that class are called.

To implement behavior for the **JadeWebSocket** class, you must subclass it and reimplement the **onOpen**, **onMsg**, and **onClose** methods.

You can also reimplement the **send** and **sendText** methods, if required. Although it is not necessary to reimplement them, if you do, you should call the **inheritMethod** instruction within the reimplementation so that those methods still send messages to the client when called.

Exercise 6 – Responding to Messages

In this exercise, you will implement a subclass of the **JadeWebSocket** class to send a reply whenever it gets a message.

1. From the Class Browser or **WebSocketSchema**, press F4 to find the **JadeWebSocket** class and then create a subclass called **WSEcho**.
2. Add a method called **onMsg** to the **WSEcho** class.

   A warning is displayed, saying that you are implementing a superclass method. Click **Yes**.
3. Implement the method as follows.

```
onMsg(msg: Binary; utf8encoded: Boolean; finalFragment: Boolean) updating, protected;

vars
    textMsg : StringUtf8;
begin
    textMsg := msg.StringUtf8;
    self.sendText("I receieved your message: " & textMsg, finalFragment);
end;
```

4. Modify the **btnOpen_click** method to replace **JadeWebSocket** with **WSEcho**, as follows.

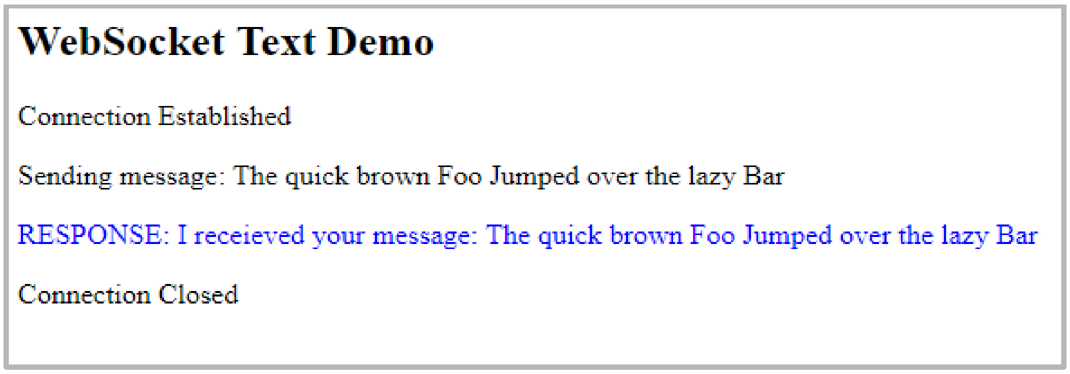
```
btnOpen_click(btn: Button input) updating;

begin
    create myServer transient;
    self.lblState.caption := "Socket is OPEN";
    self.myServer.run(null, 4444, WSEcho);

epilog
    delete myServer;
    self.lblState.caption := "Socket is CLOSED";
end;
```

5. Run the JadeScript **startForm** method and open the socket.

   When it's open, run the **WebSocketTest.html** again. You should see the following.



WebSocket Text Demo

Connection Established

Sending message: The quick brown Foo Jumped over the lazy Bar

RESPONSE: I receieved your message: The quick brown Foo Jumped over the lazy Bar

Connection Closed

Exercise 7 – Reimplementing onOpen, onClose, and sendText

In this exercise, you will reimplement the **onOpen** and **onClose** methods to write to the Jade Interpreter Output Viewer whenever a connection is opened or closed, and the **sendText** method to write the sent message to the Jade Interpreter Output Viewer before sending it.

1. Add the following methods to the **WSEcho** class. Click **Yes** when warned about reimplementing a superclass method.

```
onOpen(fullUrl: String) protected;

begin
    write "Opened connection from URL: " & fullUrl;
end;
```

```
sendText(msg: StringUtf8; finalFragment: Boolean);

begin
    write "Sending message: " & msg;
    inheritMethod(msg, finalFragment);
end;
```

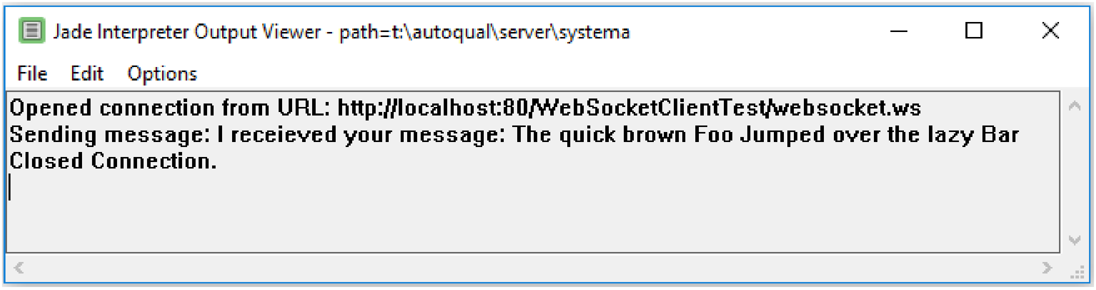
```
onClose() protected;

begin
    write "Closed Connection.";
end;
```

Note The **onOpen** and **onClose** methods have empty default implementations, so it is not useful to call the **inheritMethod** instruction for those methods. However, as the default implementation of the **sendText** method is responsible for sending the message to the client, it is therefore important to call it somewhere in the reimplementation by using the **inheritMethod** instruction.

2. Ensuring that **WebSocketForm** is running and the socket is open, run the **WebSocketTest.html** file.

   In addition to the display in the web browser, the following should be displayed in the Jade Interpreter Output Viewer.


Jade Interpreter Output Viewer - path=t:\autoqual\server\systema
File
Edit
Options

```
Opened connection from URL: http://localhost:80/WebSocketClientTest/websocket.ws
Sending message: I receieved your message: The quick brown Foo Jumped over the lazy Bar
Closed Connection.
```

Exercise 8 – Sending Images over WebSockets

In this exercise, you will use the WebSocket protocol to send an image from JADE to an HTML JavaScript test page.

1. Create a **C:/Images** folder and copy the **JADE.jpeg** file, which can be found at https://www.jadeworld.com/developer-center/learn-jade/, to the newly created folder.
2. Add a **sendImage** method to the **WSEcho** class, coded as follows.

```
sendImage();

vars
    file : File;
begin
    write "Got message to start sending a picture";
    create file;
    file.kind := File.Kind_Binary;
    file.openInput('C:/Images/JADE.jpeg');
    self.send(file.readBinary(file.fileLength), true);
epilog
    delete file;
end;
```

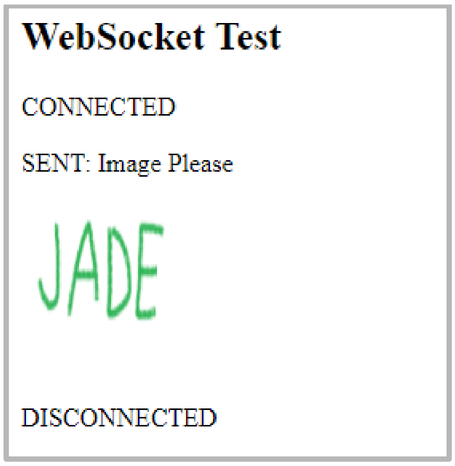
3. Modify the **WSEcho** class **onMsg** method as follows.

```
onMsg(msg: Binary; utf8encoded: Boolean; finalFragment: Boolean) updating, protected;

vars
    textMsg : StringUtf8;

begin
    textMsg := msg.StringUtf8;
    if textMsg = "Image Please" then
        self.sendImage();
    else
        self.sendText("I receieved your message: " & textMsg, finalFragment);
    endif;
end;
```

4. Run the **WebSocketImage.html** file, found at https://www.jadeworld.com/developer-center/learn-jade/. You should see the **JADE.jpeg** image displayed as the returned message.


WebSocket Test

CONNECTED

SENT: Image Please

DISCONNECTED