SGCWEBSOCKETS · TECHNICAL DOCUMENT

STOMP (ActiveMQ)

STOMP 1.2 client tuned for Apache ActiveMQ — destination prefixes, JMS-style headers and broker-specific features.

eSeGeCe

WWW.ESEGECE.COM

Overview

This is the Client Protocol STOMP Component for ActiveMQ Broker.

At a glance

COMPONENT CLASS

`TsgcWSPClient_STOMP_ActiveMQ`

STANDARDS / SPEC

STOMP 1.2 specification

TRANSPORTS

TCP, TLS

PLATFORMS

Windows, macOS, Linux, iOS, Android

FRAMEWORKS

VCL, FireMonkey, Lazarus / FPC

EDITION

Standard / Professional / Enterprise

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Standards & specs	STOMP 1.2 specification · Apache ActiveMQ — STOMP transport
Component class	`TsgcWSPClient_STOMP_ActiveMQ` (unit `sgcWebSocket_Protocol_STOMP_ActiveMQ_Client`)
Frameworks	VCL, FireMonkey, Lazarus / FPC
Platforms	Windows, macOS, Linux, iOS, Android

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`ActiveMQ_Options`	ActiveMQ-specific STOMP extensions added to the CONNECT frame, such as the durable activemq.client-id required to resume durable topic subscriptions.
`Authentication`	Sends login and passcode headers in the STOMP CONNECT frame to authenticate the client against the ActiveMQ broker.
`Client`	References the TsgcWebSocketClient that carries STOMP frames to the ActiveMQ broker when connecting over WebSockets.
`Broker`	References a TsgcWSSTOMPBroker component so the ActiveMQ STOMP protocol runs over raw TCP instead of WebSockets.
`Options`	Generic STOMP-level settings sent in the CONNECT frame, such as the virtual host header. For ActiveMQ-specific JMS extensions use ActiveMQ_Options.
`HeartBeat`	Negotiates the STOMP 1.1/1.2 heart-beat header so the client and the ActiveMQ broker exchange keep-alive newlines and detect silent drops.
`Versions`	Selects which STOMP wire versions (1.0, 1.1, 1.2) are offered to the ActiveMQ broker in the WebSocket subprotocol list and the CONNECT accept-version header.
`Guid`	Unique identifier that binds this ActiveMQ STOMP subprotocol instance to its WebSocket or broker connection.

`Version`	Read-only string with the sgcWebSockets build version of the ActiveMQ STOMP subprotocol component.

Main methods

The principal public methods exposed by the component.

`Disconnect()`	Sends a DISCONNECT frame to gracefully shut down the ActiveMQ STOMP session.
`SubscribeTopic()`	Subscribes to an ActiveMQ topic, prefixing the destination with /topic/.
`UnSubscribeTopic()`	Removes a previous topic subscription.
`PublishTopic()`	Publishes a message to an ActiveMQ topic.
`SubscribeQueue()`	Subscribes to an ActiveMQ queue, prefixing the destination with /queue/.
`UnSubscribeQueue()`	Removes a previous queue subscription.
`PublishQueue()`	Publishes a message to an ActiveMQ queue.
`SubscribeEx()`	Subscribes to an arbitrary STOMP destination using a full ActiveMQ header collection.
`UnSubscribeEx()`	Removes a subscription previously opened with SubscribeEx.
`PublishEx()`	Sends a message to an arbitrary STOMP destination using a full ActiveMQ header collection.

Public events

The component exposes the following published events; consult the online help for full event-handler signatures.

`OnActiveMQConnected`	TsgcWSPClient_STOMP_ActiveMQ › Events › OnActiveMQConnected
`OnActiveMQDisconnected`	TsgcWSPClient_STOMP_ActiveMQ › Events › OnActiveMQDisconnected
`OnActiveMQError`	Fires when an ERROR frame is received from the ActiveMQ broker; delivers the error message and the full ERROR Headers.
`OnActiveMQMessage`	TsgcWSPClient_STOMP_ActiveMQ › Events › OnActiveMQMessage

`OnActiveMQPing`	Fires when a heart-beat (single newline) is exchanged with the ActiveMQ broker; reports direction and timestamps of the last exchange.
`OnActiveMQReceipt`	TsgcWSPClient_STOMP_ActiveMQ › Events › OnActiveMQReceipt

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **Client AMQP Connect — Basic Usage** configuration sourced from the online help.

> **About this scenario.** Connect to AMQP server without authentication. Define the AMQPOptions property values, virtual host and then set in the TsgcWebSocketClient the Host and Port of the server.

Delphi (VCL / FireMonkey)

```
oAMQP := TsgcWSPClient_AMQP.Create(nil);
oAMQP.AMQPOptions.Locale := 'en_US';
oAMQP.AMQPOptions.MaxChannels := 100;
oAMQP.AMQPOptions.MaxFrameSize := 16384;
oAMQP.AMQPOptions.VirtualHost := '/';
oAMQP.HeartBeat.Enabled := true;
oAMQP.HeartBeat.Interval := 60;

oClient := TsgcWebSocketClient.Create(nil);
oAMQP.Client := oClient;
oClient.Specifications.RFC6455 := false;
oClient.Host := 'www.esegece.com';
oClient.Port := 5672;
oClient.Active := True;
```

C++ Builder

```
oAMQP = new TsgcWSPClient_AMQP();
oAMQP->AMQPOptions->Locale = "en_US";
oAMQP->AMQPOptions->MaxChannels = 100;
oAMQP->AMQPOptions->MaxFrameSize = 16384;
oAMQP->AMQPOptions->VirtualHost = "/";
oAMQP->HeartBeat->Enabled = true;
oAMQP->HeartBeat->Interval = 60;

oClient = new TsgcWebSocketClient();
oAMQP->Client = oClient;
oClient->Specifications->RFC6455 = false;
oClient->Host = "www.esegece.com";
oClient->Port = 5672;
oClient->Active = true;
```

.NET (C#)

```
oAMQP = new TsgcWSPClient_AMQP();
oAMQP.AMQPOptions.Locale = "en_US";
oAMQP.AMQPOptions.MaxChannels = 100;
oAMQP.AMQPOptions.MaxFrameSize = 16384;
oAMQP.AMQPOptions.VirtualHost = "/";
oAMQP.HeartBeat.Enabled = true;
oAMQP.HeartBeat.Interval = 60;

oClient = new TsgcWebSocketClient();
oAMQP.Client = oClient;
oClient.Specifications.RFC6455 = false;
oClient.Host = "www.esegece.com";
oClient.Port = 5672;
oClient.Active = true;
```

Common scenarios

The following scenarios are lifted verbatim from the online help. Each shows the configuration and method calls needed to drive the component through a specific real-world flow.

1 · Client AMQP1 Connect — Basic Usage

Connect to an AMQP 1.0.0 server without authentication. Define the AMQPOptions property values, virtual host and then set in the TsgcWebSocketClient the Host and Port of the server.

Delphi (VCL / FireMonkey)

```
// Creating AMQP client
oAMQP := TsgcWSPClient_AMQP1.Create(nil);
// Creating WebSocket client
oClient := TsgcWebSocketClient.Create(nil);
// Setting WebSocket specifications
oClient.Specifications.RFC6455 := False;
// Setting WebSocket client properties
oClient.Host := 'amqp_host_address';
oClient.Port := 5672;
// Assigning WebSocket client to AMQP client
oAMQP.Client := oClient;
// Activating WebSocket client
oClient.Active := True;
```

C++ Builder

```
// Creating AMQP client
oAMQP = new TsgcWSPClient_AMQP1(this);
// Creating WebSocket client
oClient = new TsgcWebSocketClient(this);
// Setting WebSocket specifications
oClient->Specifications->RFC6455 = false;
// Setting WebSocket client properties
oClient->Host = L"amqp_host_address";
oClient->Port = 5672;
// Assigning WebSocket client to AMQP client
oAMQP->Client = oClient;
// Activating WebSocket client
oClient->Active = true;
```

.NET (C#)

```
oAMQP = new TsgcWSPClient_AMQP1(this);
// Creating WebSocket client
oClient = new TsgcWebSocketClient(this);
// Setting WebSocket specifications
oClient.Specifications.RFC6455 = false;
// Setting WebSocket client properties
oClient.Host = "amqp_host_address";
oClient.Port = 5672;
// Assigning WebSocket client to AMQP client
oAMQP.Client = oClient;
// Activating WebSocket client
oClient.Active = true;
```

2 · Client MQTT Connect — Basic Usage

Connect to Mosquitto MQTT server using websocket protocol. Subscribe to topic: "topic1" after connect.

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oClient.Host := 'test.mosquitto.org';
oClient.Port := 8080;
oMQTT := TsgcWSPClient_MQTT.Create(nil);
oMQTT.Client := oClient;
oClient.Active := True;

procedure OnMQTTConnect(Connection: TsgcWSConnection; const Session: Boolean; const ReasonCode:
 const ReasonName: string; const ConnectProperties: TsgcWSMQTTCONNACKProperties);
begin
oMQTT.Subscribe('topic1');
end;
```

C++ Builder

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8080;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const int ReasonCode,
 const string ReasonName, const TsgcWSMQTTCONNACKProperties *ConnectProperties);
{
oMQTT->Subscribe("topic1");
}
```

.NET (C#)

```
oClient = new TsgcWebSocketClient();
oClient.Host = "test.mosquitto.org";
oClient.Port = 8080;
oMQTT = TsgcWSPClient_MQTT.Create(nil);
oMQTT.Client = oClient;
oClient.Active = true;

void OnMQTTConnect(TsgcWSConnection Connection, bool Session, int ReasonCode,
 string ReasonName, TsgcWSMQTTCONNACKProperties ConnectProperties);
{
oMQTT.Subscribe("topic1");
}
```

3 · Subscribe QoS = At Least Once

You can Subscribe to a Topic using method Subscribe from TsgcWSPClient_MQTT. This method has the following parameters:

Delphi (VCL / FireMonkey)

```
MQTT.Subscribe('topic1', mtqsAtLeastOnce);
```

C++ Builder

```
MQTT->Subscribe("topic1", mtqsAtLeastOnce);
```

.NET (C#)

```
MQTT.Subscribe("topic1", TmqttQoS.mtqsAtLeastOnce);
```

4 · Subscribe Topic

Subscribe to Topic "topic1" after a successful connection.

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oClient.Host := 'test.mosquitto.org';
oClient.Port := 8080;
oMQTT := TsgcWSPClient_MQTT.Create(nil);
oMQTT.Client := oClient;
oClient.Active := True;

procedure OnMQTTConnect(Connection: TsgcWSConnection; const Session: Boolean; const ReasonCode:
 const ReasonName: string; const ConnectProperties: TsgcWSMQTTCONNACKProperties);
begin
oMQTT.Subscribe('topic1');
end;
```

C++ Builder

```
oClient = new TsgcWebSocketClient();
oClient->Host = "test.mosquitto.org";
oClient->Port = 8080;
oMQTT = new TsgcWSPClient_MQTT();
oMQTT->Client = oClient;
oClient->Active = true;

void OnMQTTConnect(TsgcWSConnection *Connection, const bool Session, const int ReasonCode,
 const string ReasonName, const TsgcWSMQTTCONNACKProperties *ConnectProperties);
{
oMQTT->Subscribe("topic1");
}
```

.NET (C#)

```
oClient = new TsgcWebSocketClient();
oClient.Host = "test.mosquitto.org";
oClient.Port = 8080;
oMQTT = TsgcWSPClient_MQTT.Create(nil);
oMQTT.Client = oClient;
oClient.Active = true;

void OnMQTTConnect(TsgcWSConnection Connection, bool Session, int ReasonCode, string ReasonName,
 TsgcWSMQTTCONNACKProperties ConnectProperties);
{
oMQTT→Subscribe("topic1");
}
```

5 · Publish Messages

The method PublishMessages is used to send a message to the AMQP server.

Delphi (VCL / FireMonkey)

```
AMQP.PublishMessage('channel_name', 'exchange_name', 'routing_key', 'Hello from sgcWebSockets!!!

procedure OnAMQPBasicReturn(Sender: TObject; const aChannel: string;
 const aReturn: TsgcAMQPFramePayload_Method_BasicReturn;
 const aContent: TsgcAMQPMessageContent);
begin
DoLog('#AMQP_basic_return: ' + aChannel + ' ' + IntToStr(aReturn.ReplyCode) + ' ' + aReturn.Repl
end;
```

C++ Builder

```
AMQP→PublishMessage("channel_name", "exchange_name", "routing_key", "Hello from sgcWebSockets!!

private void OnAMQPBasicReturn(TObject *Sender, const string aChannel,
 const TsgcAMQPFramePayload_Method_BasicReturn *aReturn,
 const TsgcAMQPMessageContent *aContent)
{
DoLog("#AMQP_basic_return: " + aChannel + " " + IntToStr(aReturn→ReplyCode) + " " + aReturn→Re
}
```

.NET (C#)

```
AMQP.PublishMessage("channel_name", "exchange_name", "routing_key", "Hello from sgcWebSockets!!!

private void OnAMQPBasicReturn(TObject Sender, const string aChannel,
  const TsgcAMQPFramePayload_Method_BasicReturn aReturn,
  const TsgcAMQPMessageContent aContent)
{
DoLog("#AMQP_basic_return: " + aChannel + " " + aReturn.ReplyCode.ToString() + " " + aReturn.Rep
}
```

6 · Publish a simple message

You can publish messages to all subscribers of a Topic using Publish method, which has the following parameters:

Delphi (VCL / FireMonkey)

```
MQTT.Publish('topic1', 'Hello Subscribers topic1');
```

C++ Builder

```
MQTT->Publish("topic1", "Hello Subscribers topic1");
```

.NET (C#)

```
MQTT.Publish("topic1", "Hello Subscribers topic1");
```

Sources used to build this document

Every external claim links back to a primary source. The online-help references decode the canonical deep-link the company maintains for this component.

Primary standard / spec — STOMP 1.2 specification	stomp.github.io/stomp-specification-1.2.html
Primary standard / spec — Apache ActiveMQ — STOMP transport	activemq.apache.org/components/classic/documentation/stomp
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/Protocols/Subprotocols/STOMP/Brokers/TsgcWSPClient_STOMP_ActiveMQ.htm
Component page	www.esegece.com/products/websockets/protocols/stomp-activemq/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the STOMP (ActiveMQ) component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.