SGCWEBSOCKETS · TECHNICAL DOCUMENT

AMQP 1.0

AMQP 1.0 client for Delphi, C++ Builder and .NET — the OASIS-standardised messaging protocol used by Azure Service Bus, ActiveMQ Artemis and Apache Qpid.

eSeGeCe

WWW.ESEGECE.COM

Overview

AMQP (Advanced Message Queuing Protocol) 1.0.0 is a messaging protocol designed for reliable, asynchronous communication between distributed systems. It facilitates the exchange of messages between applications or components in a decoupled manner, allowing them to communicate without direct dependencies. Here's a technical breakdown of some key aspects of AMQP 1.0.0:

At a glance

COMPONENT CLASS

`TsgcWSPClient_AMQP1`

STANDARDS / SPEC

AMQP 1.0 overview — OASIS standard

TRANSPORTS

TCP, TLS

PLATFORMS

Windows, macOS, Linux, iOS, Android

FRAMEWORKS

VCL, FireMonkey, Lazarus / FPC, .NET

EDITION

Standard / Professional / Enterprise

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Standards & specs	AMQP 1.0 overview — OASIS standard · AMQP 1.0 transport — OASIS standard
Component class	`TsgcWSPClient_AMQP1` (unit `sgcWebSocket_Protocol_AMQP1_Client`)
Frameworks	VCL, FireMonkey, Lazarus / FPC, .NET
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.

`Client`	References the TsgcWebSocketClient that carries AMQP 1.0 frames when tunnelling the protocol over WebSockets.
`Broker`	Optional TsgcWSPClient_Broker that lets several subprotocols share a single WebSocket connection instead of each subprotocol owning its own Client.
`AMQPOptions`	Container-level parameters negotiated with the peer during the AMQP 1.0 Open frame (container-id, channel-max, max-frame-size, idle-timeout and SASL/link windowing).
`Guid`	Unique identifier that binds this subprotocol instance to its WebSocket or broker connection.
`Version`	Read-only string with the sgcWebSockets build version of the AMQP 1.0 subprotocol component.

Main methods

The principal public methods exposed by the component.

`CreateSenderLink()`	Attaches an AMQP 1.0 sender link to an existing session and returns it, ready to publish messages to the supplied target address.

`CreateReceiverLink()`	Attaches an AMQP 1.0 receiver link to an existing session and returns it, ready to consume messages from the supplied source address.
`PutCBSToken()`	Sends a Claims-Based Security (CBS) put-token request to the $cbs management link, authorising the given audience with the supplied token.
`Close()`	Sends the AMQP 1.0 Close frame to the peer, ending the connection at the container level.
`Ping()`	Sends an empty AMQP 1.0 frame as a keep-alive so the peer does not trip its idle-timeout.
`CloseSession()`	Ends an AMQP 1.0 session by sending the End frame on its channel and releasing all links attached to it.
`CloseLink()`	Detaches an AMQP 1.0 link (sender or receiver) from its session by sending the Detach frame.
`WriteData()`	Low-level method that writes raw bytes straight onto the underlying WebSocket / broker transport, bypassing the AMQP 1.0 codec. Overloaded for string or stream input.
`CreateSession()`	Begins an AMQP 1.0 session on the container by sending a Begin frame and returns the resulting session object.
`CreateCBSLink()`	Attaches the pair of links to the reserved $cbs management node used by Azure Service Bus / Event Hubs for Claims-Based Security.

Public events

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

`OnAMQPBeforeReadFrame`	Fires for every AMQP 1.0 frame received from the peer before the component dispatches it, allowing inspection or suppression.
`OnAMQPBeforeWriteFrame`	Fires for every AMQP 1.0 frame the client is about to emit, allowing inspection, last-moment edits or suppression.
`OnAMQPClose`	Fires when an AMQP 1.0 Close frame is received from the peer, carrying the optional error that terminated the container.
`OnAMQPConnect`	Fires after the peer answers with its Open frame, i.e. when the AMQP 1.0 container-level handshake has completed.

`OnAMQPDisconnect`	Fires after the underlying transport (WebSocket or TCP) has closed, reporting the close code.
`OnAMQPException`	Fires when an unhandled exception is raised while processing AMQP 1.0 frames.
`OnAMQPLinkClose`	Fires when the peer answers with a Detach frame, releasing a sender or receiver link.
`OnAMQPLinkOpen`	Fires when the peer answers with an Attach frame, confirming that a sender or receiver link is fully established.
`OnAMQPMessage`	Fires when a complete message arrives on a receiver link; the handler chooses the terminal delivery state that is sent back to the peer.
`OnAMQPMessageSent`	Fires after the last Transfer frame of a message has been written to the transport on a sender link.
`OnAMQPMessageSentAck`	Fires when the peer returns the Disposition frame that settles a previously sent message, reporting its terminal delivery state.
`OnAMQPSASLAuthentication`	Fires with the SASL outcome returned by the peer during the AMQP 1.0 SASL layer, letting the handler decide whether to swallow a failure.
`OnAMQPSessionClose`	Fires when the peer answers with an End frame, terminating an AMQP 1.0 session.
`OnAMQPSessionOpen`	Fires when the peer answers a session Begin frame, confirming the session has been opened on its channel.

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **TsgcWSPClient_AMQP1 — Connection** configuration sourced from the online help.

> **About this scenario.** The connection starts with the client (usually a messaging application or service) initiating a TCP connection to the server (the message broker). The client connects to the server's port, typically 5672 for non-TLS connections and 5671 for TLS-secured connections. Once the TCP connection is established, the client and server negotiate the AMQP protocol version they will use. AMQP 1.0.0 supports various versions, and during negotiation, both parties agree on using version 1.0.0.

Delphi (VCL / FireMonkey)

```
// Creating AMQP client
    oAMQP := TsgcWSPClient_AMQP1.Create(nil);
    // Setting AMQP authentication options
    oAMQP.AMQPOptions.Authentication.AuthType := amqp1authSASLPlain;
    oAMQP.AMQPOptions.Authentication.Username := 'sgc';
    oAMQP.AMQPOptions.Authentication.Password := 'sgc';
    // Creating WebSocket client
    oClient := TsgcWebSocketClient.Create(nil);
    // Setting WebSocket specifications
    oClient.Specifications.RFC6455 := False;
    // Setting WebSocket client properties
    oClient.Host := 'www.esegece.com';
    oClient.Port := 5671;
    oClient.TLS := True;
    // 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);
    // Setting AMQP authentication options
    oAMQP->AMQPOptions->Authentication->AuthType = amqp1authSASLPlain;
    oAMQP->AMQPOptions->Authentication->Username = L"sgc";
    oAMQP->AMQPOptions->Authentication->Password = L"sgc";
    // Creating WebSocket client
    oClient = new TsgcWebSocketClient(this);
    // Setting WebSocket specifications
    oClient->Specifications->RFC6455 = false;
    // Setting WebSocket client properties
    oClient->Host = L"www.esegece.com";
    oClient->Port = 5671;
    oClient->TLS = true;
    // Assigning WebSocket client to AMQP client
    oAMQP->Client = oClient;
    // Activating WebSocket client
    oClient->Active = true;
```

.NET (C#)

```
oAMQP = new TsgcWSPClient_AMQP1(this);
    // Setting AMQP authentication options
    oAMQP.AMQPOptions.Authentication.AuthType = amqp1authSASLPlain;
    oAMQP.AMQPOptions.Authentication.Username = "sgc";
    oAMQP.AMQPOptions.Authentication.Password = "sgc";
    // Creating WebSocket client
    oClient = new TsgcWebSocketClient(this);
    // Setting WebSocket specifications
    oClient.Specifications.RFC6455 = false;
    // Setting WebSocket client properties
    oClient.Host = "www.esegece.com";
    oClient.Port = 5671;
    oClient.TLS = true;
    // Assigning WebSocket client to AMQP client
    oAMQP.Client = oClient;
    // Activating WebSocket client
    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 · Send Message

Use the method SendMessage passing the Session and SenderLink name to send a text message to the AMQP1 Server. The method has the following parameters:

```
Delphi (VCL / FireMonkey)
```

```
oAMQP1.SendMessage('MySession', 'MySenderLink', 'My first AMQP Message');
```

```
C++ Builder
```

```
oAMQP1->SendMessage("MySession", "MySenderLink", "My first AMQP Message");
```

```
.NET (C#)
```

```
oAMQP1.SendMessage("MySession", "MySenderLink", "My first AMQP Message");
```

3 · SASL Authentication

The most common authentication is using amqp1authSASLPlain type. This authentication type, can be enabled in the AMQP1 component, accessing to the property AMQPOptions.Authentication.

```
Delphi (VCL / FireMonkey)
```

```
procedure OnAMQP1SASLAuthentication(Sender: TObject;
  aCode: TsgcAMQP1SaslCode; const aDescription: string; var Handled: Boolean);
begin
  ShowMessage('#sasl-authentication: ' + aDescription);
end;
```

C++ Builder

```
void __fastcall OnAMQP1SASLAuthentication(System::TObject* Sender, TsgcAMQP1SaslCode aCode,
    const System::UnicodeString aDescription, bool &Handled)
{
  ShowMessage("#sasl-authentication: " + aDescription);
}
```

.NET (C#)

```
public void OnAMQP1SASLAuthentication(object Sender, TsgcAMQP1SaslCode aCode,
        string aDescription, ref bool Handled)
{
  MessageBox.Show("#sasl-authentication: " + aDescription);
}
```

4 · Sending a Close Reason

The AMQP client can inform the server that the connection will be closed and provide information about the reason why is closing the connection. Use the method Close to request a connection close to the server.

Delphi (VCL / FireMonkey)

```
oAMQP.Close('invalid-frame', 'The received frame has an invalid format.');
```

C++ Builder

```
oAMQP.Close('invalid-frame', "The received frame has an invalid format.");
```

.NET (C#)

```
oAMQP.Close('invalid-frame', "The received frame has an invalid format.");
```

5 · Create Receiver Link

To Create a new Receiver Link, call the method CreateReceiverLink which contains the following parameters:

Delphi (VCL / FireMonkey)

```
oAMQP1.CreateReceiverLink('MySession', 'MyReceiverLink');
procedure procedure TfrmClientAMQP1.AMQP1AMQPLinkOpen(Sender: TObject; const aSession: TsgcAMQP1
begin
  ShowMessage('#link-open: ' + aLink.Name);
end;
```

C++ Builder

```
oAMQP1->CreateReceiverLink("MySession", "MyReceiverLink");
oAMQP1->OnAMQPLinkOpen = AMQP1AMQPLinkOpen;
void __fastcall TMyForm::AMQP1AMQPLinkOpen(TObject *Sender, TsgcAMQP1Session *const aSession, Ts
{
    ShowMessage("#link-open: " + aLink->Name);
}
```

.NET (C#)

```
oAMQP1.OnAMQPLinkOpen += AMQP1AMQPLinkOpen;
oAMQP1.CreateReceiverLink("MySession", "MyReceiverLink");
private void AMQP1AMQPLinkOpen(object sender, TsgcAMQP1Session aSession, TsgcAMQP1Link aLink, Ts
{
    Console.WriteLine("#link-open: " + aLink.Name);
}
```

6 · Create Sender Link

To Create a new Sender Link, call the method CreateSenderLink which contains the following parameters:

Delphi (VCL / FireMonkey)

```
oAMQP1.CreateSenderLink('MySession', 'MySenderLink');
procedure procedure TfrmClientAMQP1.AMQP1AMQPLinkOpen(Sender: TObject; const aSession: TsgcAMQP1
begin
  ShowMessage('#link-open: ' + aLink.Name);
end;
```

C++ Builder

```
oAMQP1->CreateSenderLink("MySession", "MySenderLink");
oAMQP1->OnAMQPLinkOpen = AMQP1AMQPLinkOpen;
void __fastcall TMyForm::AMQP1AMQPLinkOpen(TObject *Sender, TsgcAMQP1Session *const aSession, Ts
{
    ShowMessage("#link-open: " + aLink->Name);
}
```

.NET (C#)

```
oAMQP1.OnAMQPLinkOpen += AMQP1AMQPLinkOpen;
oAMQP1.CreateSenderLink("MySession", "MySenderLink");
private void AMQP1AMQPLinkOpen(object sender, TsgcAMQP1Session aSession, TsgcAMQP1Link aLink, Ts
{
    Console.WriteLine("#link-open: " + aLink.Name);
}
```

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 — AMQP 1.0 overview — OASIS standard	docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html
Primary standard / spec — AMQP 1.0 transport — OASIS standard	docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-transport-v1.0-os.html
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/Protocols/Subprotocols/AMQP1/Protocol_AMQP1.htm
Online help — class reference (TsgcWSPClient_AMQP1)	www.esegece.com/help/sgcWebSockets/Components/Protocols/Subprotocols/AMQP1/TsgcWSPClient_AMQP1.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\02.WebSocket_Protocols\11.AMQP1_Client`
.NET demo project (in the sgcWebSockets package)	`.net\demos\02.WebSocket_Protocols\11.AMQP1_Client`
Component page	www.esegece.com/products/websockets/protocols/amqp-100/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the AMQP 1.0 component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.