Using MQTT Adapters in Productions

Version 2024.1
2024-07-02

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Using MQTT Adapters in Productions
InterSystems IRIS Data Platform Version 2024.1 2024-07-02
Copyright © 2024 InterSystems Corporation
All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble® InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

1

Introduction to Message Queuing Telemetry Transport (MQTT)

The Message Queuing Telemetry Transport (MQTT) is a lightweight protocol designed to allow many devices to publish data on the network. MQTT is a lightweight message protocol that is designed to allow a high message throughput even over networks with limited bandwidth. Its publish and subscribe mechanism and use of a broker server make it possible to reliably communicate messages over low-bandwidth networks even if the message publisher and subscriber have an unreliable connection to the network. MQTT is ideally suited to the Internet of Things, where there are many small devices publishing information to the network. Clients can subscribe to messages from any devices with a single connection to a broker. For more information on MQTT, see MQTT.org.

InterSystems IRIS® data platform supports MQTT 3.1. The MQTT specification is defined as an OASIS standard, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

The MQTT adapter EnsLib.MQTT.Adapter.Inbound sends MQTT connect, subscribe, and receive messages. The EnsLib.MQTT.Adapter.Outbound sends MQTT publish messages.

You can develop custom business services and operations using these adapters, or you can use the built-in business service and operation, EnsLib.MQTT.Service.Passthrough and EnsLib.MQTT.Operation.Passthrough, respectively. If you want to use the MQTT protocol outside of an interoperability production, you can use the lower-level %Net.MQTT classes.

The MQTT protocol has uses an MQTT broker server and MQTT clients. Clients publish messages and subscribe to messages. Clients publish messages to the broker and clients subscribe to the broker to receive messages. This architecture allows a client to subscribe to messages published by many different publishers with only one connection to the broker.

In many cases, you can access the MQTT broker using the built-in business service and operation, but if you need more complex behavior you may need to create custom code using the adapters.

The MQTT classes use the Eclipse Paho MQTT C Client Library.

2

Using the MQTT Adapters

This topic describes the behavior of the MQTT inbound and outbound adapters, EnsLib.MQTT.Adapter.Inbound and EnsLib.MQTT.Adapter.Outbound and how to use them in custom business services and operations.

2.1 MQTT Inbound Adapter

The MQTT inbound adapter is responsible for subscribing to messages on the broker and then receiving messages from the subscription. The inbound adapter does the following:

- On initialization, it creates the %Net.MQTT.Client client.
- On task, it will:
  1. Check if it is connected to the broker. If it is being called the first time or the connection has broken, it uses the %Net.MQTT.Client client to connect to the broker and subscribe to the specified topic, which can optionally contain wildcards.
  2. Use the %Net.MQTT.Client client to receive a topic and a message.
  3. If the receive call does not time out, it calls the business service **ProcessInput()** method and passes it the message.

2.2 Creating a Business Service Using the MQTT Inbound Adapter

To use this adapter in your production, create a new business service class as described here. Later, compile it, add it to a production, and configure it in a similar manner as the passthrough service and operation as described in Using the MQTT Passthrough Business Service and Operation. If you need new custom message classes, create them as described in Defining Messages.

The following list describes the basic requirements of the business service class:

- Your business service class should extend Ens.BusinessService.
- In your class, the *ADAPTER* parameter should equal EnsLib.MQTT.Adapter.Inbound.
- Your class should implement the OnProcessInput() method, as described in Implementing the OnProcessInput() Method

- For other options and general information, see Defining a Business Service Class.

For a description of the settings associated with the inbound adapter, see Configuring and Using the MQTT Passthrough Business Service and Operation and Settings for the MQTT Adapter.

The following example shows the overall structure of your business service class.

Class Definition

```
Class EMQTT.NewService1 Extends Ens.BusinessService
{
Parameter ADAPTER = "EnsLib.MQTT.Adapter.Inbound";

Method OnProcessInput(pInput As EnsLib.MQTT.Message, pOutput As %RegisteredObject) As %Status
{
   set tsc=$$$OK
   //your code here
   Quit tsc
}
}
```

2.3 Implementing the OnProcessInput() Method

Within your custom business service class, the signature of your **OnProcessInput()** method should be similar to

```
Method OnProcessInput(pInput As EnsLib.MQTT.Message, pOutput As %RegisteredObject) As %Status
```

2.4 MQTT Outbound Adapter

The outbound adapter is responsible for publishing messages to the broker. When the outbound adapter is called with a message, it does the following:

- It creates a connection to the broker if it does not already have one.
- If its input parameter includes a value for the topic, it uses it. If the topic is not supplied, it uses the topic specified in the setting.
- It publishes the message to the broker with the topic from its input parameter or the setting.

3

Configuring and Using the MQTT Passthrough Business Service and Operation

This topic describes the behavior of the built-in MQTT passthrough business service and the passthrough business operation.

3.1 Adding and Configuring the Passthrough Business Service

To add your business service to a production, use the Management Portal to do the following in an interoperability-enabled namespace

1. If you are using a username/password authentication with the MQTT broker, you need to first define credentials with that information (**Interoperability** > **Configure** > **Credentials** in the Management Portal). If you are using TLS authentication, see InterSystems TLS Guide.
2. Add an instance the EnsLib.MQTT.Service.Passthrough business service class to the production.
3. Configure the business service so that it can receive input.
   - Set the **TargetConfigNames** property in **Basic Settings**. This property is common to many business services and specifies where the business services sends its messages. The target can be one or more business processes and business operations.
   - Optionally, set the **CleanSession** property. If **CleanSession** is checked, the broker will not preserve any information about the subscription from one connection to the next. If you want to receive any messages that were published while your connection was temporarily down, do not check **CleanSession**.
   - Set the **ClientID** property. This identifies your client to the broker. It must be ASCII-encoded and between 1 and 23 characters. If you disconnect and then reconnect to the broker, it will preserve your identity if the **ClientID** is the same and return any messages that were sent while you were disconnected (unless you check **CleanSession**). If you leave the **ClientID** field empty, the client assigns a unique **ClientID**. If you have multiple productions running the same business service, ensure that each has a different **ClientID** or the broker will treat them as a single client and only return a message to one of them.

- Set the **ConnectTimeout** property. This sets the number of seconds that the business service will wait for an acknowledgement from the broker.
- If you are using login credentials, set the **CredentialsName** property to the credentials name you defined in Step 1.
- Set the **KeepAlive** property for the maximum number of seconds that should be between messages from the client to the broker.
- Optionally, set the **LWTTopic** and **LWTMessage** properties. These set the Last Will and Testament message for the connection. If the broker detects that the connection has unexpectedly been lost, it will publish a message with the specified topic and message.
- Set the **QOS** property, which sets the Quality of Service in the connection with the broker. A value of 0 tells the business service to not wait for an acknowledgement from the broker and a value of 1 says that the business service will wait for an acknowledgement that the broker has received the message.
- Do not check the **Retained** property. It is only used for publishing messages not for subscribing to messages. Setting this property when publishing messages tells the broker to retain this message and return it to any new subscribers.
- If you are using TLS security, set **SSLConfigName** to the name of the TLS configuration.
- If you have set **QOS** to 1, set the **Timeout** property to the number of seconds that you want the business service to wait for an acknowledgment.
- Set the **Topic** property to the topic that you want to subscribe to. The topic is typically a hierarchal string with levels of subtopics separated by a / (forward slash). On subscriptions, the topic can contain wildcards:
  - + (plus sign)—Matches any string at that level. For example "/temperatures/+/highest" would match the topics "temperatures/Paris/highest" and "temperatures/Bangkok/highest" but would not match "temperatures/Paris/highest/today". The plus sign cannot be combined with other characters within the same level.
  - # (number sign)—Can only be the last level and matches any string at that level and any subtopics. For example "temperatures/Paris/#" would match all of the following topics: "temperatures/Paris/highest", "temperatures/Paris/highest/today", and "temperatures/Paris/lowest/record/thiscentury".
- If you want to trace the MQTT actions taken by the business service, check the **Trace** property and set the **TraceLevel** property. The MQTT trace messages are written to the production log for the business service. TraceLevel can have the following values:
  - 0— MQTTCLIENTTRACEMAXIMUM, maximum level tracing.
  - 1—MQTTCLIENTTRACEMEDIUM, medium level tracing.
  - 2—MQTTCLIENTTRACEMINIMUM, minimum level tracing.
  - 3—MQTTCLIENTTRACEPROTOCOL, trace protocol level.
  - 4—MQTTCLIENTTRACEERROR, trace only errors.
  - 5—MQTTCLIENTTRACESEVERE, trace only severe errors.
  - 6—MQTTCLIENTTRACEFATAL, trace only fatal errors.
- Set the **Url** property with the URL and port number of the broker. The scheme is either "tcp" or "ssl" followed by the domain name and port delimited by a ":", for example, "tcp://MQTTBroker.example.com:1883". Typically TLS-enabled end points are configured with a port of 8883, but this is not mandatory.

4. Enable the business service in the **Basic Settings**.
5. Run the production.

Note: When using TLS, you must use an absolute path and not a relative path to the certificate file. For more information, see Creating or Editing a TLS Configuration.

3.2 Adding and Configuring the Passthrough Business Operation

To add your business operation to a production, use the Management Portal to do the following in an interoperability-enabled namespace

1. If you are using a username/password authentication with the MQTT broker, you need to first define Credentials with that information (**Interoperability** > **Configure** > **Credentials** in the Management Portal). If you are using TLS authentication, see Using TLS with InterSystems IRIS.
2. Add an instance the EnsLib.MQTT.Operation.Passthrough business operation class to the production.
3. Configure the business operation so that it can publish messages.
   - Optionally, set the **CleanSession** property. If the Clean Session property is checked, the broker will not preserve any information about the session from one connection to the next.
   - Set the **ClientID** property. This identifies your client to the broker. It must be ASCII-encoded and between 1 and 23 characters. If you disconnect and then reconnect to the broker, it will preserve your identity if the ClientID is the same. If you leave the ClientID field empty, the client assigns a unique client ID.
   - Set the **ConnectTimeout** property. This sets the number of seconds that the business service will wait for an acknowledgement from the broker.
   - If you are using login credentials, set the **CredentialsName** property to the credentials name you defined in Step 1.
   - Set the **KeepAlive** property for the maximum number of seconds that should be between messages from the client to the broker.
   - Optionally, set the **LWTTopic** and **LWTMessage** properties. These set the Last Will and Testament message for the connection. If the broker detects that the connection has unexpectedly been lost, it will publish a message with the specified topic and message.
   - Set the **QOS** property, which sets the Quality of Service in the connection with the broker. A value of 0 tells the business operation to not wait for an acknowledgement from the broker and a value of 1 says that the business operation will wait for an acknowledgement that the broker has received the message.
   - Check the **Retained** property if you are publishing the message to retain. Each subscription has only one retained message. It is the message that is sent to a first-time subscriber. Other messages are only sent to the subscribers if they are published after the subscription has started.
   - If you are using TLS security, set **SSLConfigName** to the name of the TLS configuration.
   - If you have set **QOS**to 1, set the **Timeout** property to the number of seconds that you want the business service to wait for an acknowledgment.
   - Set the **Topic** property, which specifies the default topic for publishing messages. The topic is typically a hierarchal string with levels of subtopics separated by a / (forward slash). The operation will use the topic specified in its input parameter. If the input parameter does not have a topic, it uses the default topic. When publishing messages, the topic should not contain any wildcards.

- If you want to trace the MQTT actions taken by the business service and business operation select the **Trace** property and set the **TraceLevel** property. The MQTT trace messages are written to the production log. TraceLevel can have the following values:
  - 0— MQTTCLIENTTRACEMAXIMUM, maximum level tracing.
  - 1—MQTTCLIENTTRACEMEDIUM, medium level tracing.
  - 2—MQTTCLIENTTRACEMINIMUM, minimum level tracing.
  - 3—MQTTCLIENTTRACEPROTOCOL, trace protocol level.
  - 4—MQTTCLIENTTRACEERROR, trace only errors.
  - 5—MQTTCLIENTTRACESEVERE, trace only severe errors.
  - 6—MQTTCLIENTTRACEFATAL, trace only fatal errors.
- Set the **Url** property with the URL and port number of the broker. The scheme is either "tcp" or "ssl" followed by the domain name and port delimited by a ":", for example, "tcp://MQTTBroker.example.com:1883". Typically TLS-enabled end points are configured with a port of 8883, but this is not mandatory.

4. Enable the business operation in **Basic Settings**.
5. Run the production.

Note: When using TLS, you must use an absolute path and not a relative path to the certificate file. For more information, see Creating or Editing a TLS Configuration.

3.3 MQTT Message Format

The passthrough business service sends an EnsLib.MQTT.Message message and the passthrough business operation receives an EnsLib.MQTT.Message message. The EnsLib.MQTT.Message class is an extension of the Ens.StringContainer class with an additional property, Topic. The MQTT topic is stored in the Topic property and the MQTT message is stored in the string container value.

4

Settings for the Inbound and Outbound MQTT Adapter

This topic provides reference information on the MQTT inbound and outbound adapters settings.

Also see Settings in All Productions.

4.1 Summary

The MQTT adapters have the following settings:

Group	Settings
MQTT	Clean Session, Credentials Name, Client ID, Connect Timeout, Keep Alive, LWT Topic, LWT Message, QOS, Retained, SSL Config Name, Timeout, Topic, Trace, Trace Level, Url

The remaining settings are common to all business services. For information, see Settings for All Business Services.

4.2 Clean Session

Determines if the session to the broker will use a clean session.

4.3 Client ID

This is the string which identifies this client to the broker. It must be ASCII-encoded and between 1 and 23 characters.

If not specified the client will create a unique id.

4.4 Connect Timeout

This is the connect timeout. Connecting to a busy server may take some time and this timeout can be used to avoid a premature connection failure. Specifies the number of seconds to wait before a connection attempt fails.

4.5 Credentials Name

This is the ID name of the set of credentials values used to access the MQTT broker. The username and password defined in your Credentials item must be ASCII-encoded. Not required if the broker does not require login credentials.

4.6 Keep Alive

Specifies the maximum number of seconds that should be between messages from the client to the broker.

4.7 LWT Topic

This the LWT (Last Will and Testament) Topic. If specified must be ASCII-encoded. If the client disconnects due to an network error, the LWTMessage will be delivered to subscribers to the LWTTopic.

4.8 LWT Message

This is the LWT (Last Will and Testament) message. If specified must be ASCII-encoded.

4.9 QOS

This determines the quality of service required. It can have either of these two values:

- 0— QOSFireAndForget: Do not wait for a response from the broker.
- 1—QOSWaitForDelivery: Wait for a response from the broker and issue an error if the broker does not respond..

4.10 Retained

This is the flag that indicates to the broker whether the message should be retained by the broker.

4.11 SSL Config Name

This is the ID name of the TLS configuration that you wish to be used to communicate with the broker. Only used if TLS communication is required.

Note: When using TLS, you must use an absolute path and not a relative path to the certificate file. For more information, see Creating or Editing a TLS Configuration.

4.12 Timeout

This is the timeout expressed in seconds to wait to send (with ack) or receive a message.

4.13 Topic

This is the name of the topic to which you wish to publish or subscribe. The topic must be ASCII-encoded. The topic is typically a hierarchal string with levels of subtopics separated by a / (forward slash). In a subscription, a topic can have wildcards as a topic level:

- + (plus sign)—Matches any string at that level. For example, “/temperatures/+/highest” would match the topics “temperatures/Paris/highest” and “temperatures/Bangkok/highest” but would not match “temperatures/Paris/highest/today”. The plus sign cannot be combined with other characters within the same level.
- # (number sign)—Can only be the last level and matches any string at that level and any subtopics. For example “temperatures/Paris/#” would match all of the following topics: “temperatures/Paris/highest”, “temperatures/Paris/highest/today”, and “temperatures/Paris/lowest/record/thiscentury”.

4.14 Trace

This enables or disables the MQTT tracing facility. The MQTT trace messages are written to the production log for the business service or operation.

4.15 Trace Level

This is the trace level for the MQTT library. Set this to log the required detail of trace information. Trace Level can have the following values, which are defined in the `%Net.MQTT` include file:

- 0— MQTTCLIENTTRACEMAXIMUM, maximum level tracing.
- 1—MQTTCLIENTTRACEMEDIUM, medium level tracing.
- 2—MQTTCLIENTTRACEMINIMUM, minimum level tracing.
- 3—MQTTCLIENTTRACEPROTOCOL, trace protocol level.

- 4—MQTTCLIENTTRACEERROR, trace only errors.
- 5—MQTTCLIENTTRACESEVERE, trace only severe errors.
- 6—MQTTCLIENTTRACEFATAL, trace only fatal errors.

4.16 Url

This is the URL of the broker to which you wish to communicate. The scheme is either “tcp” or “ssl” followed by the domain name and port delimited by a “:”, for example, “tcp://BIGBADAPPLE.local:1883”. Typically TLS-enabled end points are configured with a port of 8883, but this is not mandatory. The **Url** must be ASCII-encoded.