

## Reference Manual

7.3

### Message Hub Adapter



---

Last updated 2018-10-30

---

The documentation is designed to support Unit4 Business World.

This document is intended for Unit4 Business World Consultants and customer Super Users, and thus assumes in-depth knowledge of existing Unit4 Business World functionality.

Every effort has been made to supply complete and accurate information; however, the information in this document is subject to change without notice. UNIT4 N.V. and its group companies assume no responsibility or liability for any errors that may occur in the documentation.

Unit4 welcomes your comments as part of the process of continuous development and improvement of the documentation. Please contact [Unit4 Customer Support](#) if you have any questions.

Copyright of the attached documentation is the property of UNIT4 N.V. and/or its group companies. Reproduction of this documentation for any purposes is prohibited without the prior express written authorisation of UNIT4 N.V. or its group companies. Any unauthorised use, copying or sale of the above may constitute an infringement of copyright and may result in criminal or other legal proceedings.

Copyright © 2018 UNIT4 N.V. and/or its group companies. All rights reserved. Any other brand names and/or trademarks referenced herein are either registered or unregistered trademarks of their respective proprietors.

## Table of contents

---

<b>Read this first</b> .....	<b>4</b>
<b>What is Unit4 Business World Adapter?</b> .....	<b>5</b>
<b>How does Unit4 Business World Adapter work?</b> .....	<b>6</b>
<b>Configuring Unit4 Business World Adapter</b> .....	<b>8</b>
Configuring U4BW Adapter service .....	9
Configuring U4BW Adapter through App.config configuration file .....	10
Cache configuration and other runtime parameters .....	12
Configuring U4BW Adapter parameters through U4BW Management Console - AMC (premise only) .....	13
Configuring tenants using U4BW Adapter .....	15
Configuring tenants in the cloud .....	16
Configuring tenants on premise .....	18
Configuring U4BW Publish/Subscribe API .....	19
<b>Running Unit4 Business World Adapter on premise</b> .....	<b>21</b>
Sending the messages to Unit4 Message Hub .....	22
Programming a receiver assembly for U4BW Adapter .....	23
<b>Example of use</b> .....	<b>24</b>

## Read this first

### Purpose

This reference manual gives a detailed description of Message Hub Adapter, and is intended for people responsible for its implementation. We assume that the reader - you - belong to this group of people.

### Read on-line

The manual is published as a PDF document with hyperlinks, intended to be read on-line. Hyperlinks are displayed like this: [Read this first](#).

### Content

We have divided the manual into these sections:

- [What is Unit4 Business WorldAdapter?](#) Introduces the definition of Message Hub Adapter. This chapter explains the possible scenarios where Message Hub Adapter works.
- [How does Unit4 Business WorldAdapter work?](#) Describes the general functionality of Message Hub Adapter in its two main roles: a receiver and a sender.
- [Configuring Unit4 Business World Adapter](#). Contains information about the common and specific configuration of Message Hub Adapter, both as a sender and a receiver.
- [Running Unit4 Business World on premise](#). Provides details about running the service relevant to Message Hub Adapter.
- Finally, an [example of use](#), which outlines the process to have Message Hub Adapter working.

## What is Unit4 Business World Adapter?

Unit4 Business World Adapter is a common multi-tenant service that allows the communication with other Unit4 products using Unit4 Message Hub (U4MH or the Message Hub). It can work in both premise and cloud scenarios, as shown in Figure 1 and Figure 2.

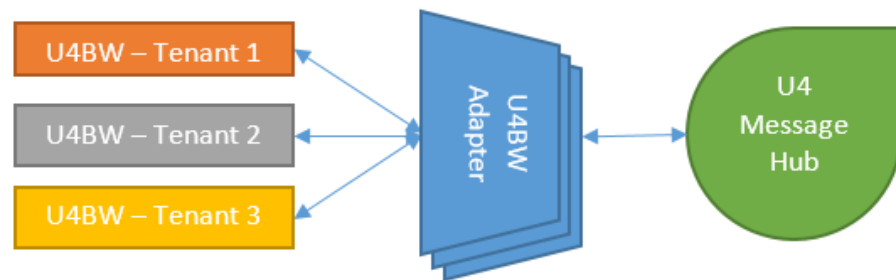


Figure 1: U4BW Adapter situation in relation to U4BW and U4MH in a cloud scenario

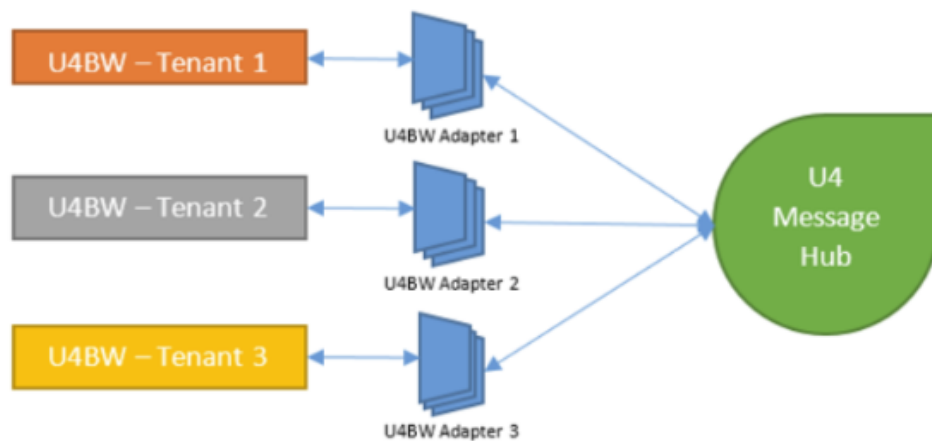


Figure 2: U4BW Adapter situation in relation to U4BW and U4MH in a premise scenario

## How does Unit4 Business World Adapter work?

Unit4 Business World Adapter (U4BW Adapter) picks up messages from Unit4 Business World Internal Bus (U4BW Internal Bus or the Internal Bus), transforms them to a valid Unit4 Message Hub message and sends them to U4MH in a transparent way for the user. This is performed following the steps illustrated in Figure 3:

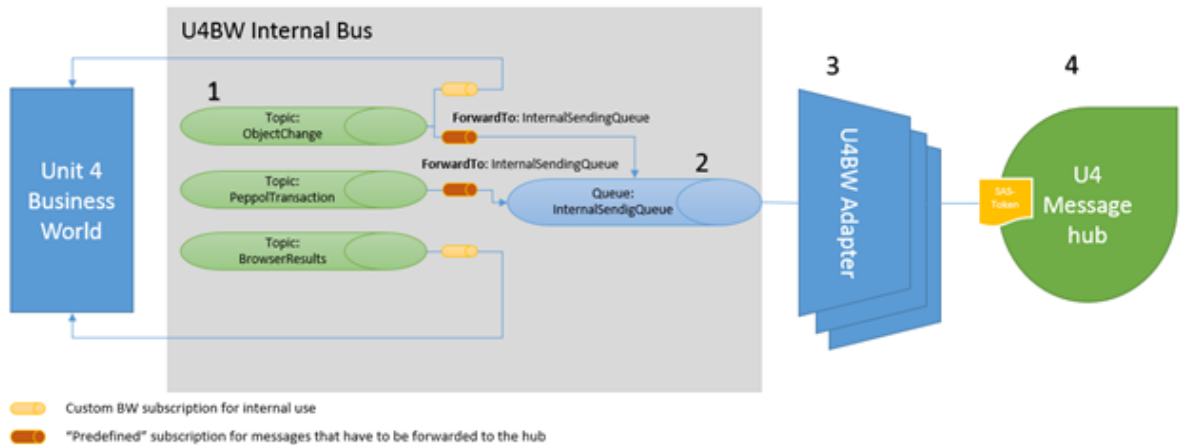


Figure 3: Steps followed by a message since it is published by U4BW until it is sent to U4MH

1. By either modifying an object in Unit4 Business World (U4BW) or using the Publish/Subscribe API, a message is sent to one of the U4BW Internal Bus topics. For example, changes in any U4BW objects will be sent to a topic named **ObjectChange** and the Peppol transaction can be sent to a topic called **PeppolTransaction**.
2. If U4BW Adapter uses a topic, it will have a subscription that automatically forwards messages to a common internal queue, named **InternalSendingQueue**. A topic can also have custom subscriptions for internal use.
3. U4BW Adapter picks up messages from an internal queue when they arrive. This is the only queue that U4BW Adapter listens to. Once a message is received, it is validated. Then its properties are copied to U4MH and the object is retrieved as JSON (only if the original message is object related). Moreover, U4BW Adapter checks if U4MH contains an **EventType** message property (creating it if necessary) and if it has any subscribers listening.
4. Finally, U4BW Adapter will send the message to U4MH, which will deliver it to the correct queue at the other side of Message Hub.

U4BW Adapter allows U4BW to receive messages from U4MH too, performing any operation on message reception. The process to receive a message from the Message Hub has the following steps (represented in Figure 4):

1. U4BW Adapter uses dynamically loaded assemblies previously programmed for each user, which include the **SourceSystem** and the **EventType** in U4MH to use, a callback function that will be performed on message reception and an exception handler.
2. U4BW Adapter creates corresponding subscriptions to the Message Hub entities.
3. Received messages are distributed per tenant, using the correct function per assembly.

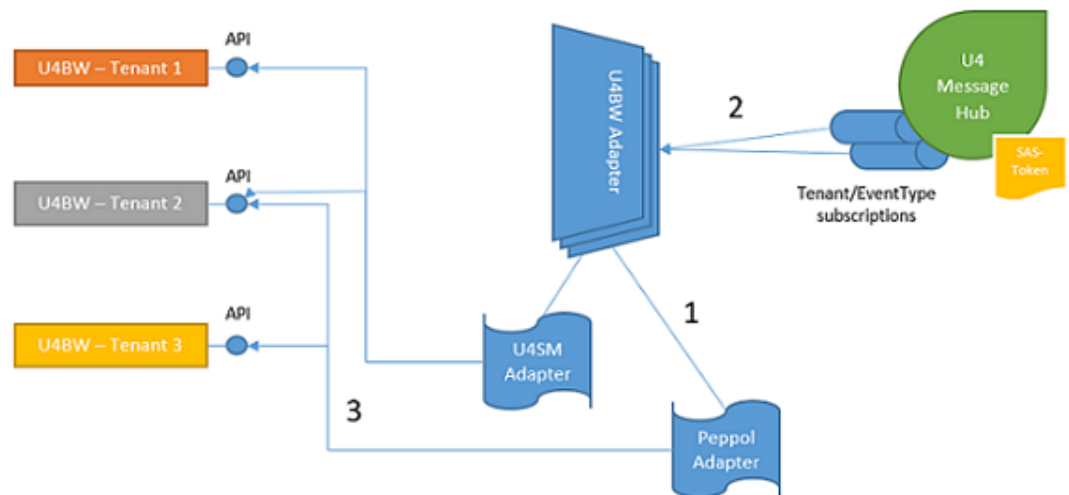


Figure 4: Steps followed by U4BW Adapter to receive messages from U4MH

## Configuring Unit4 Business World Adapter

Because the U4BW Adapter's sending and reception capabilities are completely independent, its configuration is divided into three main parts:

- [The common configuration](#)
- [The cloud configuration](#)
- [The premise configuration](#)



## Configuring U4BW Adapter service

The run parameters of the U4BW Adapter service are stored in the **App.config** file, located in the binary folder. This file contains program runtime parameters, such as the source system name, the cache configuration, the log configuration, retry policy or keep alive time. In the cloud installation, all the configuration regarding U4MH will also be stored in this file.

For premise installations, the configuration is stored in the U4BW sensitive data repository (table *aagwebconfig* in the database). To access these parameters, it is necessary to use Unit4 Business World Management Console (AMC). They can be found under the Messaging node, located under `<DataSource>/Features/Business Server- /Configuration` route (see Figure 5).

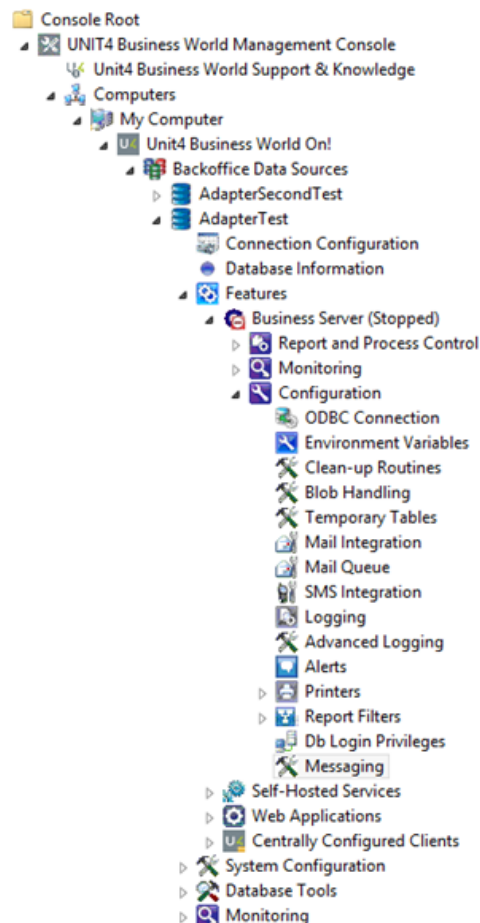


Figure 5: The U4BW running parameters configuration node

## Configuring U4BW Adapter through App.config configuration file

The **App.config** file is an XML file, located in the binary folder, which contains **service runtime parameters**. These parameters are categorized in the following groups:

### a. Source system name

Since the main purpose of the U4BW Adapter is to send messages to U4MH, it is necessary to configure it with the source system name. This is done with the *SourceSystemName* parameter. The source system name for U4BW will be **u4bw**.

### b. Unit4 Message Hub (U4MH) service parameters (cloud only)

To connect to U4MH it is also necessary to provide a URL to the *Access Provider* and the *Manager* services. The location of these resources are indicated through the following parameters:

- *MessageHubAccessProviderUri*: the URL for accessing the U4MH Access Provider service
- *MessageHubManagerUri*: the URL for accessing the U4MH Management service

For more information about U4MH, see <https://docs-external.u4pp.com/message-hub/>

### c. Unit4 Identity Services (U4IDS) parameters (cloud only)

U4IDS is a single identity solution and architecture that allows users of the Unit4 eco-system to have one single identity across multiple platforms. Since U4MH is a main part in this eco-system, it is necessary that U4BW Adapter authenticates with it.

In U4MH, U4BW Adapter needs an IDS client to be able to perform both management (creation of event types, etc.) and full relay operations (sending and receiving). This client is configured using the following parameters:

<i>MessageHubIdsAuthority</i>	A URL for accessing the IDS authority that will validate the identity
<i>MessageHubIdsClient</i>	The IDS client name for the Message Hub client
<i>MessageHubIdsClientSecret</i>	The IDS client secret key for the Message Hub client

See [Unit4Identity Services library](#) for more information about U4IDS.

#### d. Unit4 Tenant Management System (U4TMS) parameters (cloud only)

U4TMS is a central cloud-hosted system for storing information about tenants and their configuration. When running in the cloud, U4BW Adapter uses U4TMS for recovering the existing tenant information, allowing them to send their messages to U4MH.

The specific U4TMS configuration for U4BW Adapter is located in the **U4.TMS.SDK.dll.config** file located in the binary folder.

You need to provide the following parameters for the **App.config** file:

- *TmsUser*: the U4TMS user name
- *TmsSecret*: the U4TMS user secret key

For more information on how to configure each tenant in U4TMS to work with U4BW Adapter, see [Configuring tenants in the cloud](#)

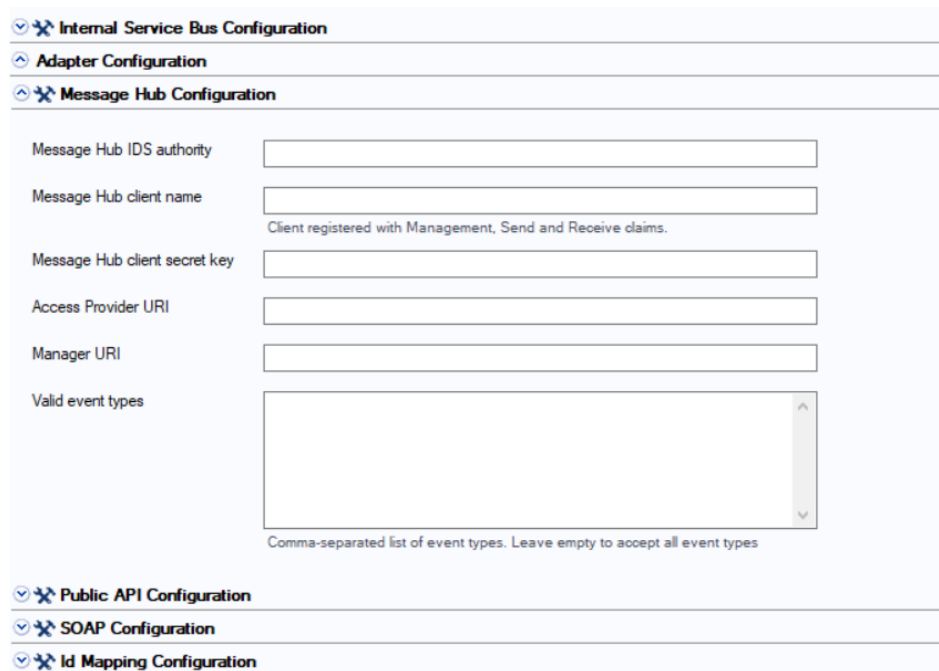
## Cache configuration and other runtime parameters

Additional configurable runtime parameters in **App.config** to set the U4BW Adapter behaviour are:

Name	Description
<i>CacheMemoryLimit</i>  Default value: 10	The amount of memory in bytes that can be used by the cache.
<i>CachePhysicalMemoryLimit</i>  Default value: 0 (zero)	The percentage of physical memory that the cache can use.
<i>CachePollingInterval</i>  Default value: 00:05:00	A time interval after the cache implementation that compares the current memory load against the memory limits, which are set for the cache instance. The settings for this configuration parameter are specified in the format HH:MM:SS.
<i>CacheAbsoluteExpirationInSeconds</i>  Default value: 300	Specifies whether a cache entry should be evicted after a specified duration in seconds.
<i>KeepAliveTimeInSeconds</i>  Default value: 30	A time interval after which U4BW Adapter refreshes the tenant configuration, updating or reconnecting the existing ones when necessary and closing the deleted ones.
<i>DeadLetterMessagesTimeToLiveInMinutes</i>  Default value: 1440 (24 hours)	A time interval in minutes after which U4BW Adapter completes all the dead-lettered messages associated with the internal sending queue.
<i>MaxInitializationRetries</i>  Default value: 5	The number of times that U4BW Adapter will retry the initialization of any tenant connection in case of an error.
<i>Logger.MinimumLevel</i>  Default value: Information	Sets the log verbosity.

## Configuring U4BW Adapter parameters through U4BW Management Console - AMC (premise only)

The U4MH configuration indicated as **cloud only** in "Configuring U4BW Adapter through App.config configuration file" on page 10 can be also set for premise installations in the AMC Messaging configuration window (Figure 6).



The screenshot displays the 'Message Hub Configuration' section of the AMC Messaging configuration window. It includes the following fields and options:

- Message Hub IDS authority:** Text input field.
- Message Hub client name:** Text input field with a note: "Client registered with Management, Send and Receive claims."
- Message Hub client secret key:** Text input field.
- Access Provider URI:** Text input field.
- Manager URI:** Text input field.
- Valid event types:** Text area for a comma-separated list of event types. A note below reads: "Comma-separated list of event types. Leave empty to accept all event types".

Below the Message Hub Configuration section, there are three expandable sections:

- Public API Configuration
- SOAP Configuration
- Id Mapping Configuration

Figure 6: The AMC Messaging configuration window with the Message Hub Configuration parameters

The Message Hub Configuration parameters include:

- *Access Provider URI:* a URL for accessing the U4MH Access Provider service
- *Manager URI:* a URL for accessing the U4MH Management service

Additionally, the IDS configuration for U4MH can be set in the Message Hub Configuration section:

Name	Description
<i>Message Hub IDS authority</i>	A URL for accessing the IDS authority that will validate the identity for the Message Hub
<i>Management Hub client name</i>	The IDS client name for the Message Hub client
<i>Management Hub client secret key</i>	The IDS client secret key for the Message Hub client

## Configuring tenants using U4BW Adapter


The tenant configuration for U4BW Adapter can be loaded in two different ways depending on the execution mode selected: it will be read from U4TMS in the cloud mode and from *DataSource* on premise.

For both of them, it is necessary to have previously set the correct configuration for using the U4BW Publish/Subscribe API in order to send messages to U4MH.

Furthermore, to make U4BW work properly with U4BW Adapter, the value of the common parameter `TENANT_ID`, which uniquely identifies the product installation, should be set, since all the messages will include it as a property and as *SessionId*. Without this parameter the message will not be received by U4BW Adapter from the Internal Service Bus.

Apart from the common configuration, the additional configuration concerning the services used by U4BW Adapter (SOAP and ID Mapping) and the valid *EventTypes* can be set.

For information on how to program a new assembly that can be read from U4BW Adapter in order to perform any operation on message reception, see [Programming a receiver assembly for U4BW Adapter](#).

 Any change in the configuration of these parameters makes it necessary to restart the U4BW—stop and restart the *IIS Express* service for the web client.

## Configuring tenants in the cloud

The cloud installation will use the tenants configured in U4TMS. In order to send and receive the messages using U4BW Adapter, the tenant configuration loaded in the U4TMS must include the following configuration parameters:

- *ServiceBusConnectionString*: set as a secure parameter, contains the connection string for U4BW Internal Bus. This connection string must be the same as used by the Publish/Subscribe API.
- *InternalTopics*: must contain the names of internal topics that send the U4BW messages to the internal sending queue and the time in seconds used for each topic by the *Duplicate Detection Time Window* parameter if U4BW Adapter must create the channel. The format used by this parameter is:

```
<internal_topic_1>, <topic_1_DD_time_window>;
<internal_topic_2>, <topic_2_DD_time_window>;
...
```

If the *Duplicates Detection Time Window* parameter is not provided for a given topic, the duplicate detection feature will not be used for it.

- *InternalSendingQueueName*: the internal sending queue name

Additionally, to retrieve the JSON object and include it in the message body, the tenant configuration in U4TMS must include:

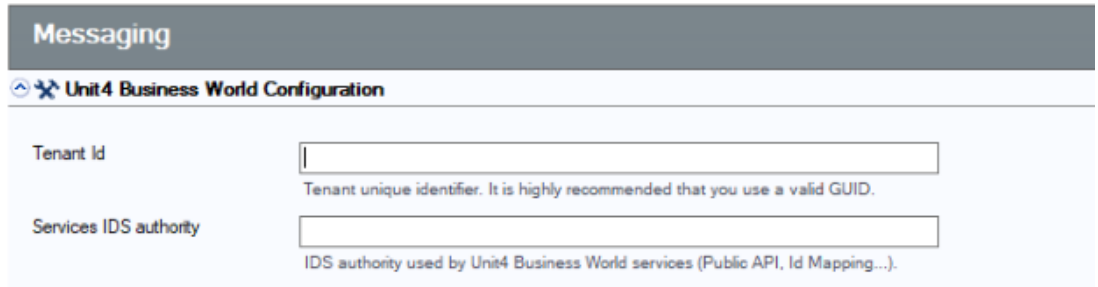
<i>PublicApiBaseUri</i>	The internal document retrieval service base URI
<i>PublicApiVersion</i>	The internal document retrieval version. If not provided, it will be set to <b>v1</b> .
<i>PublicApiAuthenticationType</i>	The internal document retrieval authentication type. It can be <b>basic</b> for basic user/password authentication or <b>token</b> for using IDS token authentication.
<i>PublicApiUser</i>	The internal document retrieval service client name
<i>PublicApiPassword</i>	The internal document retrieval service client password/IDS client secret. This parameter will be stored as a secure parameter.



<i>PublicAPIRequiredScope</i>	The IDS scope for the internal document retrieval service
<i>SoapBaseUri</i>	The SOAP base URI
<i>SoapAuthenticationType</i>	The SOAP authentication type. It can be <b>basic</b> for basic user/password authentication or <b>token</b> for using IDS token authentication.
<i>SoapUser</i>	The SOAP client name
<i>SoapPassword</i>	The SOAP client password/ids client secret. This parameter will be stored as a secure parameter.
<i>SoapRequiredScope</i>	The IDS scope for SOAP
<i>IdMappingBaseUri</i>	The ID Mapping base URI
<i>IdMappingVersion</i>	The ID Mapping version. If not provided, it will be set to <b>v1</b> .
<i>IdMappingAuthenticationType</i>	The ID Mapping authentication type. It can be <b>basic</b> for basic user/password authentication or <b>token</b> for using IDS token authentication.
<i>IdMappingUser</i>	The ID Mapping client name
<i>IdMappingPassword</i>	The ID Mapping client password/IDS client secret. This parameter will be stored as a secure parameter.
<i>IdMappingRequiredScope</i>	The IDS scope for ID Mapping
<i>ServiceIdsAuthority</i>	The Business Worldservices IDS authority URI. it's essential If any of these services needs token authentication.
<i>MessageHubValidEventTypes</i>	Used for setting a list of valid <i>EventTypes</i> . The list must contain elements separated by commas. If this parameter is not set or if its value is null or empty, all event types will be accepted.

## Configuring tenants on premise

For premise installation, the tenant is configured in the **Messaging** configuration window in AMC, under **Unit4 Business World Configuration** (Figure 7).



The screenshot shows a configuration window titled "Messaging" with a sub-section "Unit4 Business World Configuration". It contains two input fields:

- Tenant Id**: A text input field with a placeholder. Below it, the text reads: "Tenant unique identifier. It is highly recommended that you use a valid GUID."
- Services IDS authority**: A text input field with a placeholder. Below it, the text reads: "IDS authority used by Unit4 Business World services (Public API, Id Mapping...)"

Figure 7: Unit4 Business World Configuration in the AMC Messaging configuration window

The parameters included in **Unit4 Business World Configuration** are:

- *Tenant Id*: a unique identifier of the product installation. It sets the value of the common parameter `TENANT_ID`.
- *Services IDS authority*: a URL for accessing the IDS authority that will validate the identity for accessing the Public API, SOAP and Id Mapping services. It sets the value of the common parameter `IDS_AUTHORITY`.

## Configuring U4BW Publish/Subscribe API

To send messages from U4BW to U4MH, configure the **U4BW Publish/Subscribe API** by setting the following parameters in the **Messaging** AMC window, in the **Internal Service Bus Configuration** section (see Figure 8):

- *Service Bus connection string*: a connection string for the U4BW Internal Bus, which is the same as the one used by the Publish/Subscribe API
- *Internal sending queue name*: a name of the internal sending queue

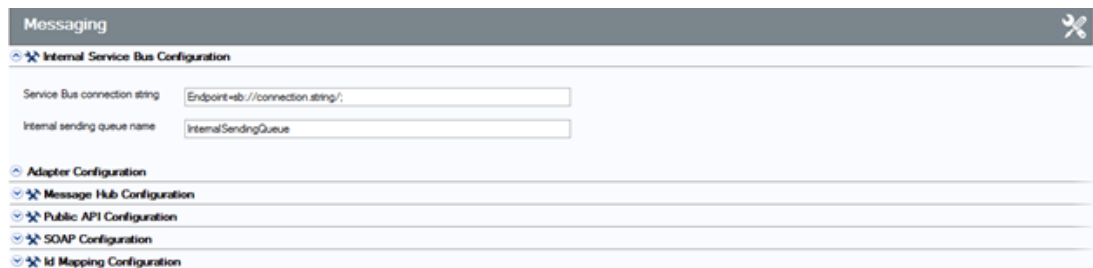


Figure 8: The AMC Messaging configuration window with the Internal Service Bus configuration parameters

In the **Adapter Configuration** section you can configure the following parameters:

Subsection	Parameter
Message Hub Configuration	<ul style="list-style-type: none"> <li>• <i>Valid event types</i>: a comma-separated list of EventType names that can be received by U4BW Adapter. If no value is provided, all EventTypes will be accepted.</li> </ul>
Public API Configuration	<ul style="list-style-type: none"> <li>• <i>Public API base Uri</i>: the internal document retrieval service base URI</li> <li>• <i>Public API version</i>: the internal document retrieval version. If not provided, it will be set to <b>v1</b>.</li> <li>• <i>Public API required scope</i>: the IDS scope for the internal document retrieval service.</li> <li>• <i>Public API service account name</i>: a name of the web service account that includes the internal document retrieval service login credentials. The web services accounts can be created in the <b>TAG200</b> window of Unit4 Business World</li> </ul>

SOAP Con- figuration	<ul style="list-style-type: none"> <li>• <i>SOAP base Uri</i></li> <li>• <i>SOAP required scope</i>: the IDS scope for SOAP</li> <li>• <i>SOAP service account name</i>: the name of the web service account that includes the SOAP login credentials. You can create web services accounts in U4BW TAG200 window.</li> </ul>
Id Mapping Configuration	<ul style="list-style-type: none"> <li>• <i>Id Mapping base Uri</i></li> <li>• <i>Id Mapping version</i>: if not provided, it will be set to v1.</li> <li>• <i>Id Mapping required scope</i>: the IDS scope for Id Mapping</li> <li>• <i>Id Mapping service account name</i>: a name of the web service account that includes the Id Mapping login credentials. You can create web services accounts in the <b>TAG200</b> window of Unit4 Business World</li> </ul>

Apart from these parameters, the internal topic configuration is set with the system value `MH_VALID_TOPICS` in the table *asysvalues*. The names of the topics are defined in the *Description1* column and in case the *Duplicates Detection Time Window* parameter is used, it is specified in seconds in the *Description2* column. Since it is not possible to modify system values, they are already configured by default.

## Running Unit4 Business World Adapter on premise

The U4BW Adapter runs using the AMC. For this reason, it is necessary to configure the *Data Source* and the *Business Server*, and to add the U4BW Adapter as a new **Service Process** (see Figure 9).

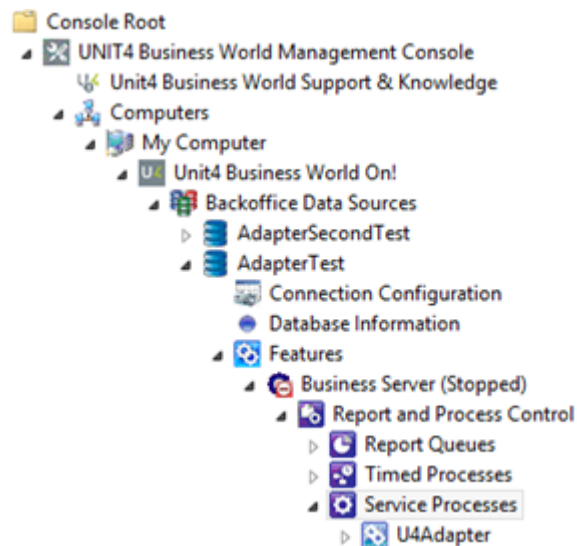


Figure 9: Add U4BW Adapter to Business Server Services

U4BW Adapter will automatically start and stop with Business Server. Its log will be stored in the *Business Server logs* folder, with the name **U4Adapter.log**.

If U4BW Adapter is launched without any process parameter, it will run the **cloud** installation. In order to run the **premise** installation, it is necessary to include the argument **/console** (or **/c**). AMC will automatically include the *Server Queue* and the *Data Source* arguments to obtain the tenant configuration parameters from the database.

Additionally, the service runs using its executable directly, located in the binary folder. Again, if the service is launched without any argument, it will run the cloud installation, launching it using the **/console** (or **/c**) argument will run the premise one. To read the tenant configuration from the database, manually enter the *Server Queue* name preceded by **-q** (for example, if the Server Queue name is U4Adapter, the service argument should be **-qU4Adapter**) and the *Data Source* name.

Each receiving assembly loaded will have its own log file. Its location will be set by the Business Server logger configuration file. If this configuration is not found or it has not been set, these logs will be saved in the executable location.

## Sending the messages to Unit4 Message Hub

In order to send the received message from U4BW there are some conditions to be fulfilled. First of all, U4BW must send the message to the bus configured in the *ServiceBusConnectionString* sensitive data parameter using the `TENANT_ID` as *Session Id*. Otherwise, U4BW Adapter will not receive those messages. The U4BW Publish/Subscribe API automatically copies this property to the message. In case that any of these parameters are changed during U4BW runtime, a restart will be necessary.

Secondly, the message must be valid according to the U4BW Adapter criteria: containing a non-null or empty *TenantId* and a non-null or empty property named *EventType*. The U4BW Publish/Subscribe API sets both properties, being the *EventType* the *ObjectId* of the object modified within U4BW.

The last condition is that the message *EventType* must exist and have subscribers in Message Hub. For more information about how to create a new subscription in U4MH, follow <https://docs-external.u4pp.com/message-hub/>.

If the message does not meet all of these conditions or any error happens during the sending process (network error, authentication error...), it will be put in the dead-letter queue of *InternalSendingQueue* including the *DeadLetterReason* and *DeadLetterErrorDescription* in the message properties and will not be sent to Message Hub. In case that the *EventType* does not exist, it will be created too, although the message will be dead-lettered.

The existing *EventTypes* are stored in a local cache (regardless of whether they have any subscribers or not) in order to improve U4BW Adapter's performance. The cache parameters are configured in the Adapter App.config file (see [Configuring U4BW Adapter service](#) for more information).

The body of the message can contain the information of the object modified, provided by the Public API, which credentials are set by its service configuration. If this configuration is not set or it is not correct, the message will be sent with its original body.

## Programming a receiver assembly for U4BW Adapter

To create a new receiver assembly that can be read by U4BW Adapter in order to perform any operation on message reception or in case that an exception occurs, it is necessary to add a new project in the *Fundamentals* solution called **Fundamentals.AdapterReceivers**. This project should generate a DLL file, which name ends with **AdapterReceiver.dll**.

In this project, at least one class must inherit from the *IAdapterReceiver* class provided in **Adapter.Receiver** and include an attribute called *ReceiverSubscription* with three parameters: a name of the *SourceSystem*, where messages are going to be received, a type of the *EventType* (**MessageType.Document** or **MessageType.Event**) to subscribe to and a comma-separated list of these *EventTypes*.

Since this class inherits from *IAdapterReceiver*, it must implement the functions *OnMessageReceived*, which includes the callback function to execute when a message is received, and *OnException*, which contains the code to run in case that any exception occurs when processing the received message.

Once the executing U4BW Adapter reads the custom receiver DLL, it will create the necessary subscriptions in order to provide the connection with U4MH. If neither the selected *SourceSystem* nor *EventType* exists, no subscription for it will be created. After *keep alive time* is configured, U4BW Adapter will try to create this subscription again.

## Example of use

This section describes an example of how a custom Unit4 Business WorldAdapter receiver can be created in the premise installation.

### Sending a message to Unit4 Message Hub

To send a message to U4MH follow the steps below:

1. In the **TAG200** window, create a new web service account for basic authentication named **testPublicApi** with a valid U4BW user and password.

2. Configure the tenants to set up the adapter service (see [Configuring tenants on premise](#)).

3. Configure the U4BW Adapter behaviour using the **App.config** file located in the binary folder. For this example, the default parameters will be used for the cache and runtime (except from the logging level, that will be set to *Verbose*):

```
<appSettings>
  <add key="SourceSystemName" value="u4bw"/>
  <add key="Logger.MinimumLevel" value="Verbose"/>
</appSettings>
```

4. In AMC, create a new *DataSource* (for this example, it will be called *AdapterTestSending*). In **Business Server configuration** node, open **Messaging** and configure the U4BW Adapter in order to use the open hub (see [Running Unit4 Business World Adapter on premise](#)).

3. Restart U4BW in order to apply the changes.

4. Add the U4BW Adapter service (see [Running Unit4 Business World Adapter on premise](#)).

5. Create a subscription for the selected *EventType* in U4MH. This can be done through Swagger following the steps provided by the U4MH documentation (<https://docs-external.u4pp.com/message-hub/>). For this test, the sending *SourceSystem* will be **u4bw**, the receiving one will be **u4bwtest** and we will use the *AppId custrec* for receiving the U4BW



**customers** event types (identified in the Message Hub with the ID 1199) from the *TenantId* set in U4BW .

In this example, the *EventType* will be created before sending the messages. In case that a message for a non-existing *EventType* is received by U4BW Adapter, it will be dead-lettered and a new *EventType* created. It will be necessary to manually add a subscription to it.

**POST**

/v2/sourceSystems/{sourceSystemName}/receivers/{appId}/tenants/{tenant}/eventTypes/{eventSourceSystem}/{messageType}/{eventTypeName}/version/{version}

Start subscribe to an event type by adding it to the subscription

Response Class (Status 200)  
OK

Model | Example Value

```
{
  "SourceSystemName": "string",
  "EventTypeName": "string",
  "MessageType": "DocumentMessage",
  "Version": "string"
}
```

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
sourceSystemName	<input type="text" value="Your Source System Name"/>	Source system name	path	string
appId	<input type="text" value="Your Receiver Name"/>	Receiver application ID	path	string
tenant	<input type="text" value="Your Tenant Name"/>	Tenant ID	path	string
eventSourceSystem	<input type="text" value="The Source System of the Event Type"/>	Event type owner	path	string
messageType	<input type="text" value="DocumentMessage"/>	Message type (DocumentEvent or MessageEvent)	path	string
eventTypeName	<input type="text" value="The Event Type Name"/>	Event type name	path	string
version	<input type="text" value="The Event Type version"/>	Version	query	string

6. Modify an existing customer or create a new one from U4BW using the **TCU002** window. A message will be automatically sent to U4MH and it will be recovered in the queue using U4Messaging SDK or through Service Bus Explorer.

7. Another way of sending a message to the Message Hub without using U4BW interface is using U4Messaging SDK.

### Receiving a message from Unit4 Message Hub

This example describes how to create a new custom receiver. It includes a new log entry every time a message from *SourceSystem u4sm* and *EventType customer* arrives, and will

complete the message. Other messages, sent for the *EventType* **personnel** will not be allowed so they will not be received.

1. Create a new library project in *Fundamentals.AdapterReceiver* solution, named **U4.Test.AdapterReceiver**. Like other U4BW projects, it is necessary to change its default project target to \$(SolutionDir)Targets\platform.csharp.targets. Include in this project the references to *Serilog*, *U4.MessageHub.ConnectorSdk* and *Adapter.Receiver*.

2. Create a new class, named **MyCustomReceiver** that extends *IAdapterReceiver*

3. In this class, include the attribute:

```
ReceiverSubscription("u4sm", MessageType.Document, "customer, personnel")
```

4. Implement the *OnException* and *OnMessageReceived* functions. One possible implementation is the one that logs the message reception, throws an exception while processing a message if it includes the *hasException* property or complete the message otherwise.

```
using System;
using System.Threading.Tasks;
using Adapter.Receiver;
using Serilog;
using U4.MessageHub.ConnectorSdk;

namespace U4.Test.AdapterReceiver
{
    [ReceiverSubscription("u4sm", "customer")]
    public class Mireceiver : IAdapterReceiver
    {
        public void OnException(Exception ex, ILogger logger)
        {
            logger.Error(ex, "Heeeeeelp!!!!");
        }

        public async Task OnMessageReceived(IU4Message message,
            IAdapterConfiguration config)
        {
            config.Logger.Information("A message arrived!!!");
            if (message.Properties.ContainsKey("hasException"))
            {
                await Task.Run(() => { });
                throw new Exception("Something went wrong");
            }

            config.Logger.Information("That was a good message");
            await message.CompleteAsync();
        }
    }
}
```

5. Compile the project. It generates the *U4.Test.AdapterReceiver DLL* in the binary folder.

6. In AMC create a new DataSource (for this example, called **AdapterTestReceiving**) and in **Business Server Configuration** node open **Messaging**. Next, set the valid *EventTypes* list value to **customer**. That way, only the customer messages will be allowed. For this example, any service will be used but they can be configured in the **Messaging** window as shown in [Configuring tenants on premise](#).

7. Add the U4BW Adapter service and run it following the steps in [Running Unit4 Business World Adapter on premise](#).

8. Send a message from the **u4sm SourceSystem** either using *Service Bus Explorer* or *U4.MessageHub.ConnectorSdk* for the event type **customer**. The message will be received and the log will be stored in the *DataSource* log file.

Sending a message for *EventType* personnel will not trigger any event. Sending a message for **customer** with the property *hasException* will trigger the *OnException* function and its log will be stored in the receiver's log.

