# Platform SDK Developer's Guide

## Friendly Reaction to Unsupported Messages

12/14/2025

# Friendly Reaction to Unsupported Messages

## Java

## Overview

This feature allows Platform SDK to deliver protocol messages which are unknown for the current protocol version (that is, messages with an unsupported message ID). This allows a user application to receive and react to "abstract" messages from the server which have no corresponding protocol Event class.

Note: Prior to release 8.5.0 of Platform SDK, there was no way to receive or react to unsupported messages.

Unsupported messages can be received as an asynchronous or unsolicited event by calling `MessageHandler.onMessage()`. The received message only has one protocol attribute declared: a protocol-specific Reference ID. It is also possible to use this attribute to receive unsupported messages with a synchronous request as shown below:

```
Message response = protocol.request(req);
```

Providing a friendly reaction to unsupported messages is optional. The feature is enabled by default, but may be disabled for all protocol types or for particular protocols in case of backward compatibility issues.

> **Tip**
>
> This feature is implemented for protocols based on Genesys-proprietary protocols messages. This includes most of the Platform SDK protocols, excluding the following XML-based ones: Chat/Callback/Email, and ESP-based UCS/EspEmail protocols).

> **Tip**
>
> This feature is designed to receive "unknown messages" from a server, not for sending these unsupported messages to a server.

## The Protocol Unknown Message

A new Platform SDK internal "protocol unknown message" class was introduced in Platform SDK release 8.5.0 to represent unsupported messages.

Genesys recommends that users do not rely on this specific class, its specific attributes or properties when handling the message. In this case, the following basic Message attributes are valuable:

- `MessageId`
- `ProtocolId`
- `Endpoint`
- `ProtocolDescription`

This base API will help provide "forward compatibility", when user application gets newer Platform SDK version which is extended with a particular message support. To ensure backwards compatibility in the future, this message does not support any protocol attributes except for `ReferenceId`.

Previously, the scenario for adding support of new protocol messages to Platform SDK required the following steps:

1. collecting technical details of new protocol message(s)
2. making a feature request to Genesys for the Platform SDK to be extended
3. waiting for development and testing to be completed
4. getting the extended version of Platform SDK and using it to update your application

This feature allows your applications to support unknown events without waiting for this process to be completed.

In this case, the attribute subscription functionality may be helpful. Your application should subscribe for needed attributes on specific message(s) using `AttributeSubscriptionList`. For example:

```
// Initialize protocol:
<AnyServer>Protocol protocol = new <AnyServer>Protocol();
protocol.set...

// Initialize attribute subscription:
final int unsupportedMessageId = 123;
final String attrId = "9";
AttributeSubscriptionList subscriptionList = new AttributeSubscriptionList();
subscriptionList.addAttribute(unsupportedMessageId, attrId);
subscriptionList.applyToContext(protocol.connectionContext());

// Initialize MessageHandler:
protocol.setMessageHandler(new MessageHandler() {
        public void onMessage(final Message message) {
            if (message.messageId() == unsupportedMessageId) {
                Object valAttr9 = message.getMessageAttribute(attrId);
                // do something with raw value of the attribute
            }
            // ...
        }
});
```

```
// Open protocol connection:
protocol.open();

// Send some request to the server to initiate responding:
protocol.send(rq);
```

This code is designed to let your application behave in the same way after Platform SDK is updated to include message support for the added messages.

## Backward Compatibility

Automatically enabling this feature may cause a change in behavior for some Platform SDK protocols.

Genesys servers usually maintain backwards compatibility, so receiving unsupported messages is not expected in normal situations. However, this feature may have an effect in scenarios when your application tries to open a protocol client connection to a server of the wrong type. So there may be some cases where you need to disable this feature to keep your applications working as expected. This section describes how to disable the feature for such cases.

In Platform SDK for Java, there is a new `PsdkCustomization` utility class that contains an option to enable or disable this feature. Option values for this class can be configured in three ways:

- a specific configuration file
- JVM system properties
- explicit calls to the `PsdkCustomization` API

The flag to enable this feature is "branchable" for particular protocols. This means that it is possible to disable the feature for specific protocol types, while keep the default value of enabled for all others. Or it is possible to disable the feature for all protocols, and then enable it only for specific types.

The following example shows how to disable the feature by default and then enables it for StatServerProtocol objects using two of the available methods:

**Using PsdkCustomization API**

```
 PsdkCustomization.setOption(PsdkOption.DisableUnknownProtocolMessageDelivery, "true");
 PsdkCustomization.setOption(PsdkOption.DisableUnknownProtocolMessageDelivery,
"Reporting.StatServer", "false");
```

**Using JVM System Properties**

```
-Dcom.genesyslab.platform.disable-unknown-incoming-messages=true
-Dcom.genesyslab.platform.Reporting.StatServer.disable-unknown-incoming-messages=false
```

## .NET

## Overview

This feature allows Platform SDK to deliver protocol messages which are unknown for the current protocol version (that is, messages with an unsupported message ID). This allows a user application to receive and react to "abstract" messages from the server which have no corresponding protocol Event class.

Note: Prior to release 8.5.0 of Platform SDK, there was no way to receive or react to unsupported messages.

Unsupported messages can be received as an asynchronous or unsolicited event using the `Received` event. The received message only has one protocol attribute declared: a protocol-specific Reference ID. It is also possible to use this attribute to receive unsupported messages with a synchronous request as shown below:

```
IMessage response = protocol.Request(req);
```

Providing a friendly reaction to unsupported messages is optional. The feature is enabled by default, but may be disabled for all protocol types or for particular protocols in case of backward compatibility issues.

> **Tip**
>
> This feature is implemented for protocols based on Genesys-proprietary protocols messages. This includes most of the Platform SDK protocols, excluding the following XML-based ones: Chat/Callback/Email, and ESP-based UCS/EspEmail protocols).

> **Tip**
>
> This feature is designed to receive "unknown messages" from a server, not for sending these unsupported messages to a server.

## The Protocol Unknown Message

A new Platform SDK internal "protocol unknown message" class was introduced in Platform SDK release 8.5.0 to represent unsupported messages.

Genesys recommends that users do not rely on this specific class, its specific attributes or properties when handling the message. In this case, the following basic Message attributes are valuable:

- `MessageId`
- `ProtocolId`
- `Endpoint`

- `ProtocolDescription`

This base API will help provide "forward compatibility", when user application gets newer Platform SDK version which is extended with a particular message support. To ensure backwards compatibility in the future, this message does not support any protocol attributes except for `ReferenceId`.

Previously, the scenario for adding support of new protocol messages to Platform SDK required the following steps:

1. collecting technical details of new protocol message(s)
2. making a feature request to Genesys for the Platform SDK to be extended
3. waiting for development and testing to be completed
4. getting the extended version of Platform SDK and using it to update your application

This feature allows your applications to support unknown events without waiting for this process to be completed.

In this case, it may be helpful to add a message handler to identify unknown messages. For example:

```
protocol.Received += (sender, e) => // assign message handler
{
  var args = e as MessageEventArgs;
  if ((args != null) && (args.Message != null))
  {
    switch (args.Message.Id){
      case 2854:{ // unknown message is handled by identifier
        // TODO: process message with id 2854
        break;
      }
      default:{
        // TODO: do something with others incoming messages
        break;
      }
    }
  }
};
```

This code is designed to let your application behave in the same way after Platform SDK is updated to include message support for the added messages.

## Backward Compatibility

Automatically enabling this feature may cause a change in behavior for some Platform SDK protocols.

Genesys servers usually maintain backwards compatibility, so receiving unsupported messages is not expected in normal situations. However, this feature may have an effect in scenarios when your application tries to open a protocol client connection to a server of the wrong type. So there may be some cases where you need to disable this feature to keep your applications working as expected. This section describes how to disable the feature for such cases.

Backwards compatibility can be preserved in .NET by using an application configuration file. The example below shows how to disable this feature for Configuration Protocol:

```
<configuration>
  <appSettings>
    <add key="ConfServerProtocolUnknownMessageEnabled" value="false"/>
  </appSettings>
</configuration>
```

Each protocol has an individual key which can be added into the application configuration file with a value of "false" to restore behavior to the way it was in earlier versions of Platform SDK. The key name can be formed using the following rules:

```
<key  name> ::= <protocol name><suffix>
<protocol name> ::= "ConfServer" | "MessageServer" | "TServer" | ... | etc.
(equal to IProtocolDescriptionSupport.ProtocolDescription.ProtocolName)
<suffix> := "ProtocolUnknownMessageEnabled"
```

To disable this feature for all protocols used in your application, update the configuration file and assign a value of false to the `ProtocolUnknownMessageEnabled` key, as shown here:

```
<configuration>
  <appSettings>
    <add key="ProtocolUnknownMessageEnabled" value="false"/>
  </appSettings>
</configuration>
```