Timeouts Under the Hood

Although the default timeout values are enough for most use cases, this guide will provide you with the information required to setup several types of timeout configurations that may affect the functionality of your OutSystems applications, and the OutSystems Platform in general, allowing you to control how the applications handle longer requests.

Context

Due to the web nature of the OutSystems Platform, running over web application servers and databases, is implicit the existence of time limits to several interconnections. Let's face it, how many of you never hit a timeout while developing, deploying or running an OutSystems application? In that regard, I think it's time to talk about the several timeout events that can influence the proper development, deploying and runtime of OutSystems applications.

So, without further delay, let's meet the candidates. For the runtime level timeouts we have the SQL Query timeout, the application request timeout, the web service request timeout, the web server connection timeout, the session timeout and timer's timeout. While for deployment level timeouts we have the Development Environment Upload/Download timeout besides any runtime level timeouts that may also influence the deployment process.

Please note that usually, if you're having timeout events then either you have a very sluggish system, which can benefit from an upgrade or extended resources, or you have a very unoptimized logic, data models or slow integrations. Of course that depends on the application's purpose and dataset volumes, and the timeout settings should be adjusted accordingly.

SQL Query Timeout
**Timeout Description**

Being supported by a database backbone (either Microsoft SQL, Oracle or MySQL) in nature, this is the most common timeout event that can occur in the OutSystems Platform and their applications, specially in low performance database servers or large data volume environments. This SQL Query timeout defines the maximum amount of time that an SQL statement can take to be fully executed by the database engine, and the resulting dataset be returned to the calling application. It influences OutSystems applications simple and advanced queries, as well as the entity actions (Create, Update, Get, GetForUpdate and Deleted). It also influences the OutSystems Platform system queries, scattered along the LifeTime and Service Center applications and OutSystems Services. At the application level, it's possible to define this timeout value in the Development Environment while in the development stage of an application module for simple or advanced queries, however, the entity actions and the OutSystems Platform system queries do not have a timeout value definition that can be changed individually and follow a platform-wide default query timeout.

**Timeout Symptoms**

The symptoms detected when this timeout value is reached are usually a runtime error either in the applications or during deployment procedures, and a logged entry in Service Center. The log entry, for the most part looks very much like:

```
Message :Error in advanced query GetConsumers in Espace (SELECT ... ): Timeout expired. The timeout period elapsed prior to completion of the operation or the server is not responding.
Stack: at ssServiceCenter.Actions.QueryEspace_UpdateCacheGetConsumers(HeContext heContext, Int32 maxRecords, String qpkey)
at ssServiceCenter.Actions.UserActionEspace_UpdateCache(HeContext heContext, Int32 inParameSpacedId, Boolean inParamSkipConsumers, Boolean inParamOnlyConsumers)
at ssServiceCenter.Actions.WebSrvcServiceStudioPublish(HeContext heContext, String inParamusername, String inParampassword, Int32 inParamversionid, RLHEMessageRecordList& outParammessages)
at ssServiceCenter.WebServices.ServiceStudio.Publish(String username, String password, Int32 versionid, WORCHEMessageRecord[]& messages)
```

In this case there's a explicit reference to a timeout and from the error stack we can assert that is occurring in a SQL query, in this example, the Advanced
Query `QueryEspace_UpdateCacheGetConsumers`. However, this is not always the case. Sometimes the timeout error appears masked under a different error, usually a very nasty runtime error, like the one below:

```plaintext
Error in advanced query DeleteTestResults in DeleteInstanceFromDatabase (DELETE ...):
Object reference not set to an instance of an object.
  CommandDelete(HeContext heContext, Int32 inParamInstanceId)
  Submitwidget491643_Click(Object sender, EventArgs e)
Exception has been thrown by the target of an invocation.
at System.Reflection.RuntimeMethodInfo.InternalInvoke(Object obj, BindingFlags invokeAttr,
  Binder binder, Object[] parameters, CultureInfo culture, Boolean isBinderDefault, Assembly
caller, Boolean verifyAccess)
at System.Reflection.RuntimeMethodInfo.InternalInvoke(Object obj, BindingFlags invokeAttr,
  Binder binder, Object[] parameters, CultureInfo culture, Boolean verifyAccess)
at System.Reflection.RuntimeMethodInfo.Invoke(Object obj, BindingFlags invokeAttr,
  Binder binder, Object[] parameters, CultureInfo culture)
  recTableTableRecords1_Select(Object sender, DataGridCommandEventArgs e)
at System.Web.UI.WebControls.DataGrid.OnItemCommand(DataGridCommandEventArgs e)
at System.Web.UI.WebControls.DataGrid.OnBubbleEvent(Object source, EventArgs e)
at System.Web.UI.Control.RaiseBubbleEvent(Object source, EventArgs args)
at System.Web.UI.WebControls.DataGridItem.OnBubbleEvent(Object source, EventArgs e)
at System.Web.UI.Control.RaiseBubbleEvent(Object source, EventArgs args)
  RaisePostBackEvent(String eventArgument)
at System.Web.UI.Page.RaisePostBackEvent(IPostBackEventHandler sourceControl, String
  eventArgument)
at System.Web.UI.Page.RaisePostBackEvent(NameValueCollection postData)
at System.Web.UI.Page.ProcessRequestMain()
```

In this rare case we appear to have a serious crash, which we immediately react with horror and fear when we find our selves in the presence of the infamous Object reference not set to an instance of an object error. But fear not, because this is in fact just a consequence of a timeout event that just exceeded the limit. Although this scenario is not as common as the previous one, it can occur when an SQL reader closes due to the timeout event, associated with some "bad" timing when it happens to be still under access, generating this null reference exception.

### Timeout Setting

For the application simple and advanced queries, is possible to specify a timeout value in seconds, during application module editing (development phase). When these queries do not have a timeout value specified, they take into account the platform wide default query timeout instead. Additionally, any system query from LifeTime, Service Center or OutSystems Services, also takes this default value as the timeout.
This default query timeout can be specified on the OutSystems Platform Configuration Tool on the database tab, and by default is defined with 30 seconds. Note that changing this value will influence the timeout value for all non-application queries.

To run the tool you can execute the ConfigurationTool.exe application located in the Platform Server installation directory. Always remember to exit the tool with the Ok button to commit your changes.

Application Request Timeout

Timeout Description

Every web request done to an OutSystems application takes its time to process by the application server (either ASP.NET over IIS or JBoss) and is ruled by a Request Timeout that prevents the existence of everlasting web requests. So basically, when a web request is taking too long to process and reaches the defined request timeout, the thread that’s running it will be abruptly interrupted, generating a runtime error and avoiding waste of resources.

Timeout Symptoms

As expected, every application-not-working-anymore runtime error that occurs will be logged in Service Center, under the Error Logs page. Since this error depends on the amount of time the request is being processed for, it can occur practically anywhere in the application request: on the page_load, on page_render, in SQL queries, in extension or eSpace methods, etc. However, the error message is always very particular. In the ASP.NET version case, the message Thread was being aborted is the dead giveaway of this timeout event. For example, the following error stack is cause by a web request timeout event that occurred right when a SQL query was being executed:

Module Name: Web service met
Message: **Thread was being aborted.**
Stack: at OutSystems.NssOMLProcessor.CssOMLProcessor.MssCompileeSpaceAsync(Boolean ssDebug, Byte[] ssOML, String ssServerKind, RLHEMessageRecordList& ssMessages, Int32& ssErrorCount, String& ssErrorText, String ssEspaceName, Int32 ssEspaceVersionId)
at ssServiceCenter.RssExtensionOMLProcessor.MssCompileeSpaceAsync(HeContext heContext, Boolean ssDebug, Byte[] ssOML, String ssServerKind, RLHEMessageRecordList& ssMessages, Int32& ssErrorCount, String& ssErrorText, String ssEspaceName, Int32 ssEspaceVersionId)
at ssServiceCenter.Actions.ssCompileFlow3004763(HeContext heContext, Boolean ssDebug, Byte[] ssOML, String ssServerKind, RLHEMessageRecordList& ssMessages, Int32& ssErrorCount, String& ssErrorText, String ssEspaceName, Int32 ssEspaceVersionId)
at ssServiceCenter.WebServices.ServiceStudio.Compile(String username, String password, Int32 versionid, WORCHEMessageRecord[]& messages)

As you can see, the error message is the dead give away ...

**Timeout Setting**

In the ASP.NET version of the OutSystems Platform, this timeout event value is defined in the machine.config file under the **httpRuntime** section, with the parameter **executionTimeout**. By default, this value is not set and it will default to 100 seconds. In the OutSystems Platform's Installation Checklist document, you'll find the instructions for setting the executionTimeout for your application server. Please obtain the installation checklist for your version and application server stack in [OutSystems Downloads](https://outsystems.com) page.

**Web Service Request Timeout**

**Timeout Description**

The web services is a very common way of integrating with different applications and external systems. The applications can invoke methods remotely through web services. These invocations are merely XML information being transferred forth and back from a web server through standard web requests/responses. Apart from the inherited web requests timeout events, the web service itself will also have an invocation timeout event. This timeout event defines the total elapsed amount of time since the web service invocation until the complete reception of its response.

**Timeout Symptoms**

The typical symptoms when this timeout event occurs are quite straightforward. In an OutSystems application, when calling a Web Service action, if the web service takes too long, it may generate the
following timeout error.

Module_Name: Web service met
Message: The operation has timed-out.

In this case, the WebClientProtocol method is the key to identify this timeout event has the web service invocation timeout.

Timeout Setting

When developing your eSpace with the Development Environment, upon adding your Web Reference to the eSpace, for each method/action you can define the Timeout In Seconds property, that specifies the maximum time value in seconds that the web service method invocation can take, prior to issue a timeout event. If there is no value specified in this property, the Web Reference method timeout is 100 seconds.

Timers Timeout

Timeout Description

Timers are OutSystems applications’ synchronous jobs that execute a specified application logic on scheduled instants, invoked by the OutSystems Scheduler Service. The timer's timeout event occurs when the elapsed time since the timer's invocation and its completion surpasses the specified value.

Timeout Symptoms

When a timer reaches its timeout settings, an error log will be registered in the Error Logs page of Service Center with an error message and error stack similar to:

Module Name: Scheduler Service:


Since it's the OutSystems Scheduler Service that invokes the timer of the application to run, it will also catch the error exception associated with the timeout event occurrence, which will abort the timer's execution. In this case, the timer execution took 1439 seconds, before being aborted.

Timeout Setting

When developing your eSpace with the Development Environment, and creating the timer objects, for each one of them its possible to specify the Timeout in Minutes property. This value is the maximum elapsed period of time that the OutSystems Platform has to completely execute the action associated with the timer. For more information on how timers are handled by the OutSystems Platform please check the Service Studio help topic How Timers are Handled.

Email Sending Timeout

Timeout Description

The OutSystems Platform allows to send emails directly form your OutSystems applications. The email sending process is done in two steps:

1. In the first step, the email is rendered and added to a queue. Rendering involves a special web request to a specific endpoint that renders the “web page” with the email content. This content is then stored to the database. This request is made synchronously in the logic that sends the email.
   E.g. if you send an email with the click of a button in a page, the email is rendered and queued before the page returns.

2. In the second step, Scheduler service picks emails from the queue and sends them to the server using the SMTP configurations in the server.

The first step is managed by a timeout while rendering the page to send by email, and when the page takes too long to render, it may affect the ability of your application to send that email.
Timeout Symptoms

When the timeout value is reached, you get an error message like this:

```
Error creating Email. The operation has timed out
  at OutSystems.HubEdition.RuntimePlatform.Email.EmailHelper.HttpGetContent(String ssUrl,
   String method, String contentType, String userAgent, Cookie cookie, QueryParameter[]
   parameters, String& ssContent, String& ssContentEncoding)
  at OutSystems.HubEdition.RuntimePlatform.Email.EmailHelper.HttpPost(String ssUrl,
   QueryParameter[] parameters, String userAgent, Cookie cookie, String& ssContent, String&
   ssContentEncoding)
   SendEmailRequest(String url, String from, String to, String cc, String bcc, Int32 activityId, Int32
   tenantId, Boolean storeContent, EmailType type)
```  

The error message is quite explicit.

Timeout Setting

When the email being sent is taking too long to render, you need to reduce the time it takes to render
the email because there’s no timeout setting that you can configure to overcome the limitation of the
page rendering process of the email engine.

Note that there is no use in resorting to HTTPRequestHandler.SetRequestTimeout in your page
request to try to extend the duration of email processing: the timeout is not related to the length of
the request to render the email; instead, the limit is in the method that calls the email
(OutSystems.HubEdition.RuntimePlatform.Email.EmailHelper.HttpGetContent) is limited in time to
ensure best practices.

To put it simply: it's the caller that is not waiting long enough for the request to be ready. The
solution, is to reduce the time it takes to render the page.