# Notifications

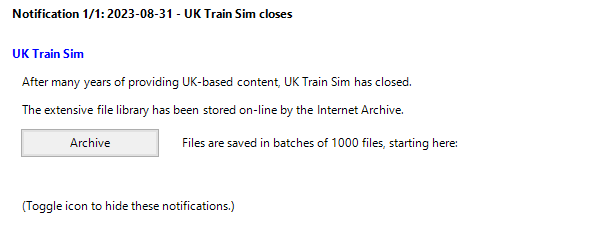
Notifications messages may be static or responsive. Responsive messages change depending on the user’s installation, settings and content. For example, if the user’s Update Mode is set to Stable, then a message intended to announce updates will report new updates for the Stable Version but not for the Testing Version.

## Privacy

Notifications are fetched from the Open Rails web server and all users get the same notifications. They are just presented differently depending on the user’s settings. Note that no information is returned to the Open Rails web server.

## Static Notifications

A notification which is static might be used to convey a simple message for all users, such as:

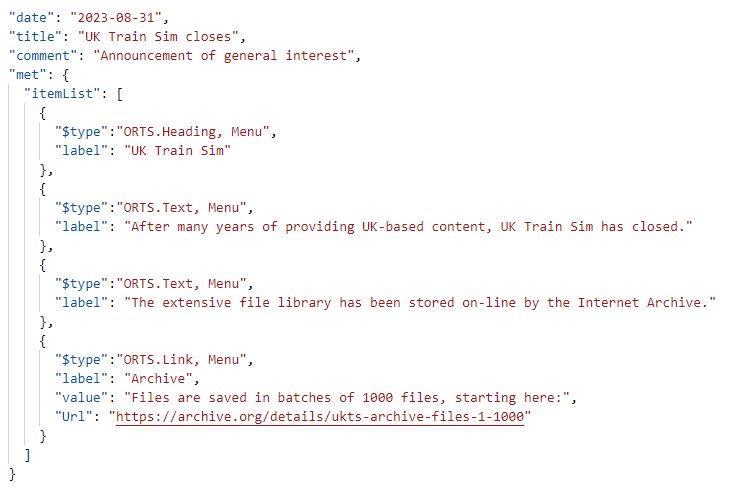


The JSON must specify a date and title which appears in the top line.

The “comment” is optional.

The “met” section is required and provides a list of items of several possible types which are specified in the “$type” field. The construction “ORTS.<ClassName>, Menu” is recognised by the Newtonsoft JSON reader as a class in the Menu executable.

This example makes use of Heading, Text and Link classes. The Link class provides a button which launches the default browser and opens it at the given URL.

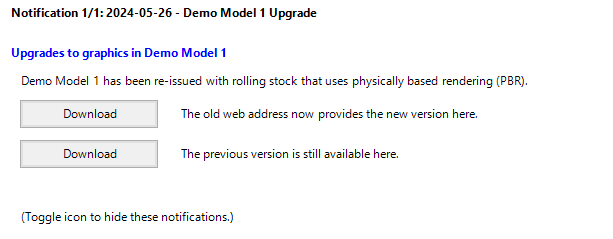


Items may also be of type “Record” and type “Update” as described below. There are optional fields for font color and for indent from the left.

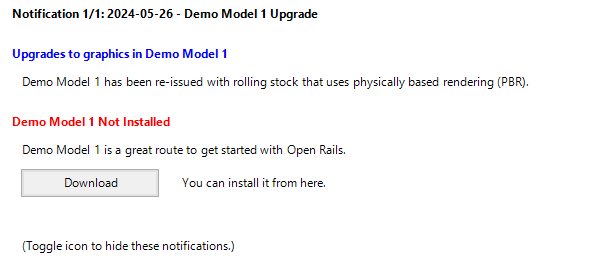
## Responsive Notifications

An example of a responsive notification is one that announces news about a route that the user already has installed.

If the user has the Demo Route 1 installed (as an installation profile), then he is shown this notification:



But if he hasn’t, then he is shown a different one:



### Parameters

The different behaviour is achieved by testing whether the installed\_routes which the user has installed includes the route folder “SCE” – as used by Demo Model 1.

The installed\_routes are automatically evaluated as a string of route folders, which may prove to contain “SCE”. “{{installed\_routes}}” is recognised as a parameter because it is enclosed by “{{ . . }}” and is evaluated automatically.

This behaviour is achieved with the following JSON:



First we have a “prefixItemList”. This is optional and provides content that appears above the “met” section. It may contain the same type of items as the met’s “itemList” does, such as Heading, Text and Link.

The “met” section contains an “itemList” as before, but is now followed by a “checkIdList” which names one of more checks that must succeed before the met’s “itemList” will be shown.

The “comment” in checkList is optional.

If the checks do not succeed, then there is an optional “unmetItemList” which will be shown in place of the met’s “itemList”.

Note that an optional item has been used to set the Heading color to “red”. The Windows system colors are recognised[[1]](#footnote-1).

These lists may be followed by an optional “suffixItemList” similar to the “prefixItemList”.

The JSON is split into 2 sections, the “notificationList”, which contains all the notifications, and the “checkList” which specifies the checks that must succeed if a “met” is to be shown.

The “checkList” allows a number of checks, each identified with an “id”, so a single check may satisfy the “met” part of several notifications.

“anyOfList” allows multiple checks to be considered.

Checks are either of type “ORTS.Includes, Menu” or “ORTS.Excludes, Menu” and are executed in sequence.

The first “Include” to succeed will end the checking process and lead to the met’s “itemList” being shown and the first “Exclude” to succeed will end the checking process but lead to the “unmetItemList” being shown. If no checks succeed, then the “unmetItemList” is shown.

The “ORTS.Includes, Menu” contains an “allOfList” which provides the checks that must all succeed if the Includes is to be successful. In this example, there is only one check - “ORTS.Contains, Menu”.

This constraint requires that “{{installed\_routes}}” contains “SCE”.

As noted in the “comment” field, “SCE” is the name of the folder containing the route “Scottish Central Express”. “{{installed\_routes}}” is a parameter which is evaluated to a string containing all the route folder names in the installation profiles.

The Contains operator is not case-significant.

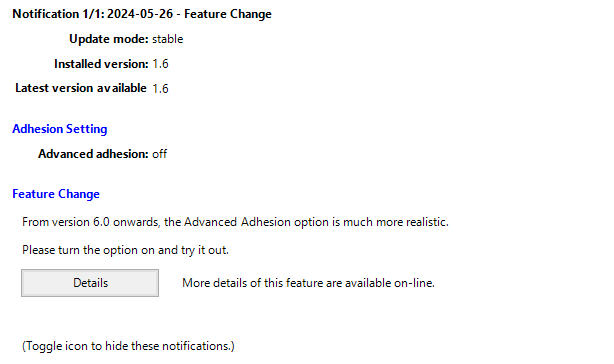
A “ORTS.NotContains, Menu” alternative is also supported.

## User Settings

Notifications can also change behaviour based on the settings which a user chooses in the “Options” menu.

In this example, we are assuming that adhesion has been greatly improved in Stable Version 1.6 and this notification is prompting users to try it out.

Only users who have the setting Advanced Adhesion turned off will see this notification. The notification has been configured so that users with Advanced Adhesion turned on will see no notification at all.



Pressing the “Details” button launches the user’s browser to show the specified webpage.

In this JSON, the parameter “{{Settings.UseAdvancedAdhesion}}” appears twice; once as a confirmation to the user of its current value - “off” and again in the checkList as a check to see if it actually is “off”.

If the check fails, then the notification is not shown at all. This behaviour is specified simply by omitting the element “unmetItemList”.

In this example, the check is for 2 conditions, that the setting is “off” and also that the “{{installed\_version}}” contains the value “1.6”. This notification will not be shown while the user stays at version 1.5 and will disappear again once the user upgrades to version 1.7.

The JSON element “allOfList” indicates a logical “and”, so that the “met” section will only be shown if all the properties have the correct values.



This example also introduces the type “ORTS.Record, Menu” which is used to set out individual records aligned as in a table format.

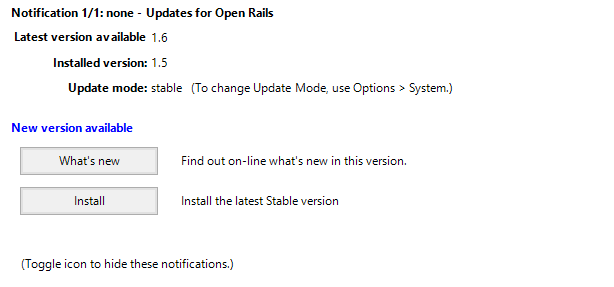
Note: the parameter for user settings is case-sensitive (in Pascal case) and the possible names are listed in <https://github.com/openrails/openrails/blob/master/Source/ORTS.Settings/UserSettings.cs>

The other parameters are not case-sensitive and a full list is appended.

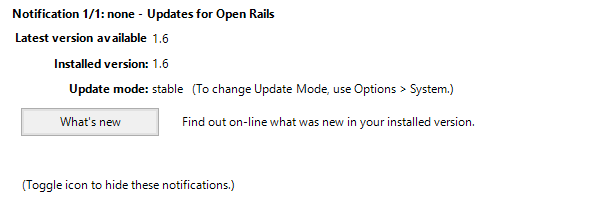
## Update Version

Notifications are also used to advise that a new version is available. The same arrangement is used for each of the “update modes” – stable, testing, unstable and none.

If a new version is available, then the notification might be shown as:



Once the user has upgraded, this is no longer appropriate and the notification is shown as:



This example introduces a new JSON element “ORTS.Update, Menu” which is used here to provide the button “Install” and its action.

The check with id=”not\_updated” uses the “Excludes” so that if the check passes, then the “unmetItemList” will be shown instead of the met’s “itemList”. The JSON element “anyOfList” indicates a logical “or”, so the 2 checks in this list will be considered independently.

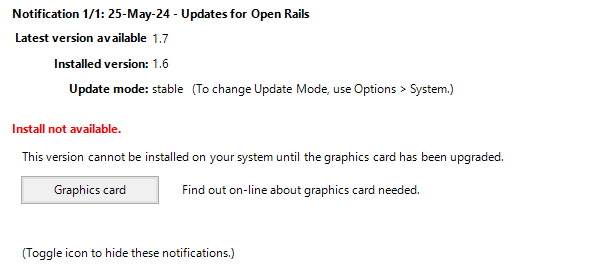


The full JSON file contains 3 notifications with different links but sharing the same check with id=”not\_updated”, one for each of the “update modes” - stable, testing and unstable. The 2 notifications which are not appropriate for the user’s current update mode are automatically excluded by the Notifications code.

## Compatibility Testing

An upgrade which includes the Physics Based Rendering (PBR) provided by the Graphics Library Transission Format (glTF) requires hardware that supports Direct3D version 10.0 or greater. Previously Open Rails has only required Direct3D version 9.3.

By adding a second check, the notification of a new version can also indicate that an upgrade of the system is required, such as:



This example is very similar ot the previous one, but there are now 2 check ids listed in the “checkIdList” of the notification, one for the version and one for the system information Direct3D.



## Trialling notifications.json

The “notificationList” is downloaded and extracted from the JSON file at <https://static.openrails.org/api/notifications/menu.json>

In order to test new versions of this file, the Notifications code will first look for a file in the Program folder of Open Rails – Program\notifications\_trial.json

If this exists, then it will be used instead of the published file. The JSON can be checked against the schema published at <https://static.openrails.org/api/notifications/schema.json> but note that no checking is done by the Notifications code.

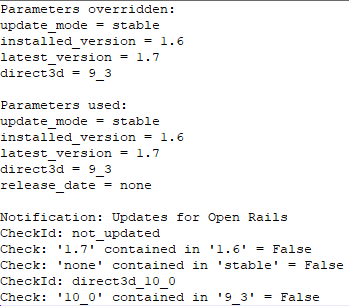
As part of trialling new version of this file, it may be helpful to simulate system information or user settings or installed routes that do not exist on the trial system. Another JSON file can be added to the Open Rails Program folder to simulate these.

This file is named Program\notifications\_trial\_parameters.json and contains a list of parameters and values. The following example was used to produce the previous notification image.



If the file Program\notifications\_trial.json exists, then a log file will be generated also in the folder - Program\notifications\_trial\_log.txt

For example:

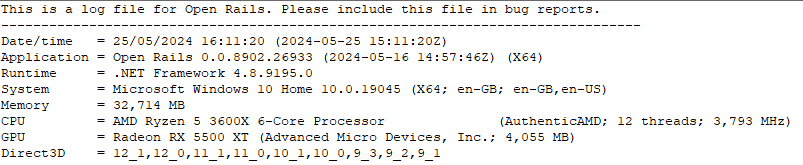


## List of System Information

This is similar to the list of system information that appears at the top of OpenRailsLog.txt:

* installed\_version
* runtime
* system
* memory
* cpu
* gpu
* direct3d

OpenRailsLog.txt sample:



1. <https://learn.microsoft.com/en-us/dotnet/api/system.windows.media.colors?view=windowsdesktop-8.0> [↑](#footnote-ref-1)