Creatio Academy

Existing additional feature

Feature Toggle mechanism

Version 7.18

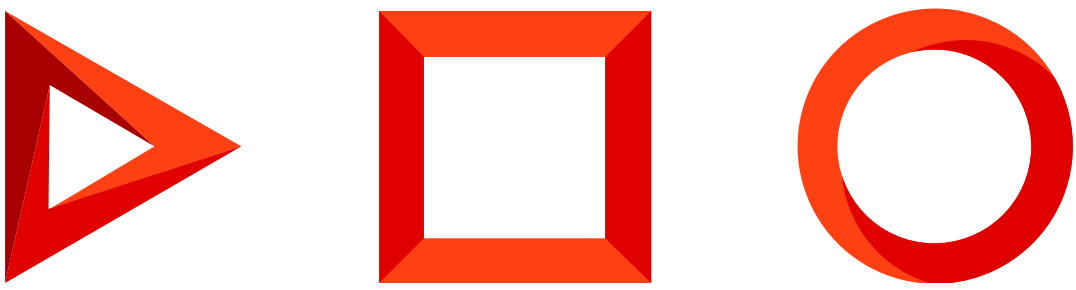


Table of Contents

Feature Toggle mechanism

Advanced

Feature toggle is a software development technique. The **purpose** of the feature toggle is to manage the additional feature status in a live application.

Feature toggle lets you use continuous integration while preserving the working capacity of the application and hiding features you are still developing. The application source code contains the additional feature block and the conditional operator that checks the feature status.

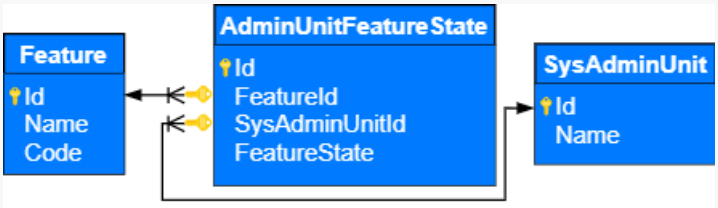
Learn more about the Feature toggle mechanism in Wikipedia.

Store the feature details

Creatio stores the feature details in the following database **tables**:

- `[Feature]`. Stores the list of features you can toggle. Empty by default.
- `[AdminUnitFeatureState]` (the `[FeatureState]` column). Stores the is the additional feature status (turned on/off). The `[AdminUnitFeatureState]` table connects the `[Feature]` and the `[SysAdminUnit]` database tables. `[SysAdminUnit]` is the additional feature status for Creatio users and user groups.

View the relationship chart between tables that store the additional feature details in the figure below.



View the primary columns of the `[Feature]` database table below.

Primary columns of the `[Feature]` table

Column	Type	Description
`[Id]`	`uniqueidentifier`	The unique ID of the record.
`[Name]`	`nvarchar(250)`	The name of the feature.
`[Code]`	`nvarchar( 50 )`	The code of the feature.

View the primary columns of the `[AdminUnitFeatureState]` database table below.

Primary columns of the `[AdminUnitFeatureState]` table

Column	Type	Description
`[Id]`	`uniqueidentifier`	The unique ID of the record.
`[FeatureId]`	`uniqueidentifier`	The unique ID of the record from the `[Feature]` table.
`[SysAdminUnitId]`	`uniqueidentifier`	The unique ID of the user record.
`[FeatureState]`	`int`	The status of the feature (`1` : turned on, `0` : turned off).

Turn on the additional feature

You can manage the additional feature status in several **ways**:

- for current Creatio user
- for all Creatio users

Turn on the feature for current Creatio user

1. Open the [*Features*] page that lets you use to turn the feature on selectively.

   Template of the feature page URL

   ```
   [Creatio URL]/0/Nui/ViewModule.aspx#BaseSchemaModuleV2/FeaturesPage
   ```

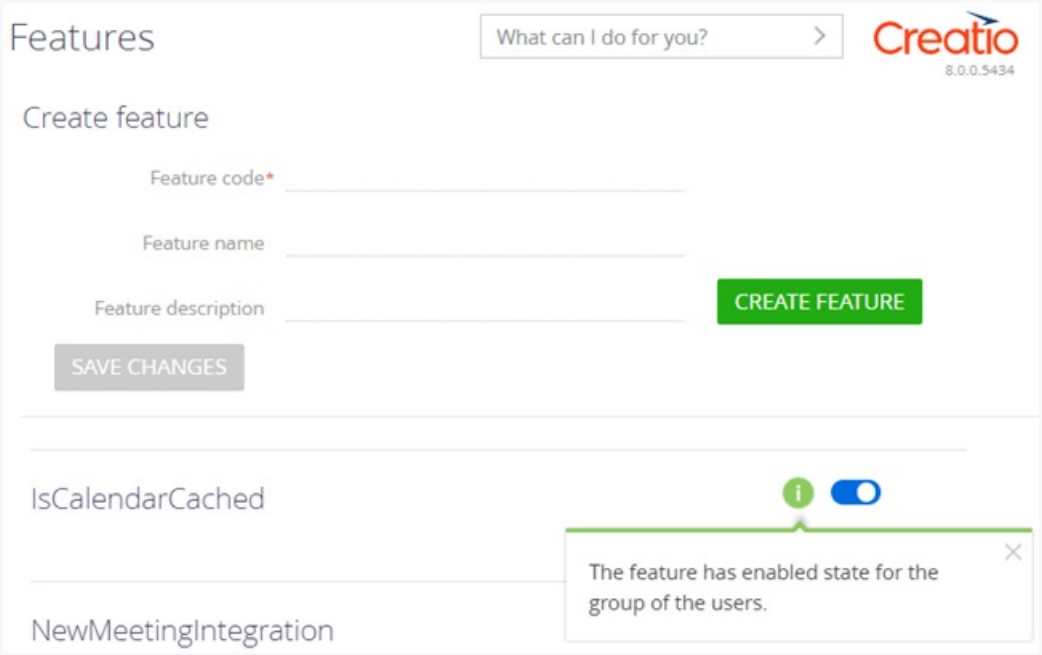
   Example of the feature page's URL

   ```
   http://mycreatio.com/0/Nui/ViewModule.aspx#BaseSchemaModuleV2/FeaturesPage
   ```

2. Turn on the corresponding switch for the needed feature.
3. Click [*Save changes*].

The tooltip displays the status of the feature for the user group of the current user. Changing the feature status on the page does not change the feature status for the user group of the current user. In this case, Creatio adds an extra rule only for the current user. This rule has a higher priority than the rule for the user group.

View the example of the [*Features*] page in the figure below.



Turn on the feature for all Creatio users

To **turn on the feature for all Creatio users**, execute a database query. You can use one of example queries below.

Example of an SQL query

MSSQL

```
DECLARE @featureCode varchar(max) = 'NewEmailSettingsWithoutAutoSynchronization', @featureId uni
set @featureId = (select top 1 Id from Feature where Code = @featureCode);
IF @featureId is null
    BEGIN
        insert into Feature
            (Name, Code)
        values
            (@featureCode, @featureCode);
        set @featureId = (select top 1 Id from Feature where Code = @featureCode);
    END;
delete from AdminUnitFeatureState where FeatureId = @featureId;
insert into AdminUnitFeatureState
    (SysAdminUnitId, FeatureState, FeatureId)
    values
        ('A29A3BA5-4B0D-DE11-9A51-005056C00008', 1, @featureId),
        ('720B771C-E7A7-4F31-9CFB-52CD21C3739F', 1, @featureId);
```

PostgreSQL

```
do $$
    DECLARE
        featureCode varchar(100) = 'EmailIntegrationV2';
        featureId UUID;
    BEGIN
        featureId = (select "Id" from "Feature" where "Code" = featureCode limit 1);
        IF featureId is null THEN
            insert into "Feature"
                ("Name", "Code")
            values
                (featureCode, featureCode);
            featureId = (select "Id" from "Feature" where "Code" = featureCode limit 1);
        end IF;
    delete from "AdminUnitFeatureState" where "FeatureId" = featureId;
    insert into "AdminUnitFeatureState"
        ("SysAdminUnitId", "FeatureState", "FeatureId")
        values
            ('A29A3BA5-4B0D-DE11-9A51-005056C00008', 1, featureId),
            ('720B771C-E7A7-4F31-9CFB-52CD21C3739F', 1, featureId);
end $$;
```

Oracle

```
DECLARE
    existObject NUMBER := 0;
    v_featurecode VARCHAR2(50) := 'LoadAttachmentsInSameProcess';
    v_featureid VARCHAR2(38);
BEGIN
    SELECT COUNT(1) INTO existObject FROM "Feature" WHERE "Code" = v_featurecode;
    if existObject = 0 THEN
        INSERT INTO "Feature" ("Name", "Code") VALUES (:v_featurecode, :v_featurecode);
    END IF;
SELECT "Id" INTO v_featureid FROM "Feature" WHERE "Code" = v_featurecode AND rownum <= 1;
DELETE FROM "AdminUnitFeatureState" WHERE "FeatureId" = v_featureid;
INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId")
    VALUES ('{A29A3BA5-4B0D-DE11-9A51-005056C00008}', 1, v_featureid);
END;
/
```

`featureCode` specifies the name of the needed feature.

You can use SQL queries to turn on the feature for the user group, turn off the feature for current user and vice versa.

Run the SQL query that has `0` in the `[FeatureState]` column of the `[AdminUnitFeatureState]` table to **turn off**

the feature for all Creatio users.

Attention. If you turn the feature on using a database query, turn it off using another query.

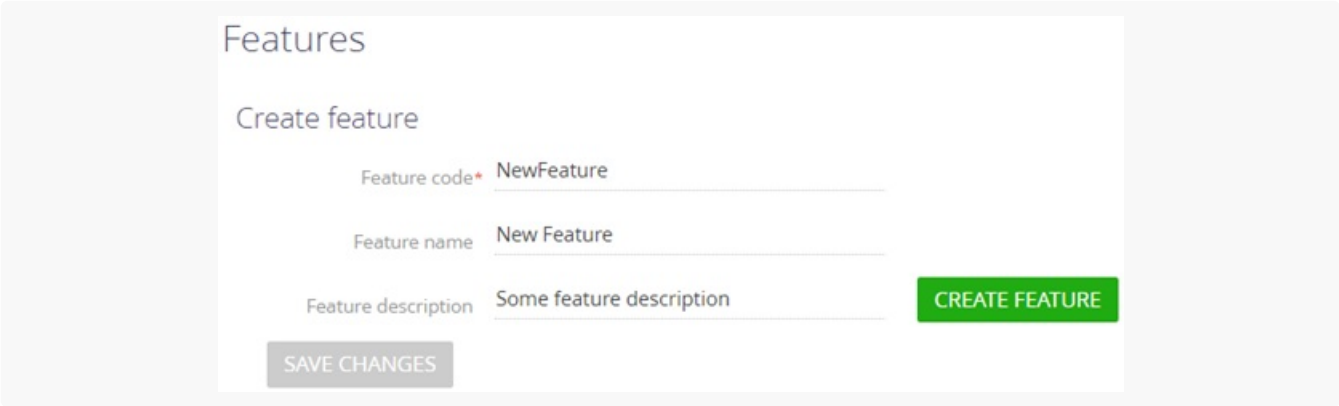
Implement a custom feature

1. Add **a custom feature**.

   a. Open the [*Features*] page that lets you turn the feature on selectively.

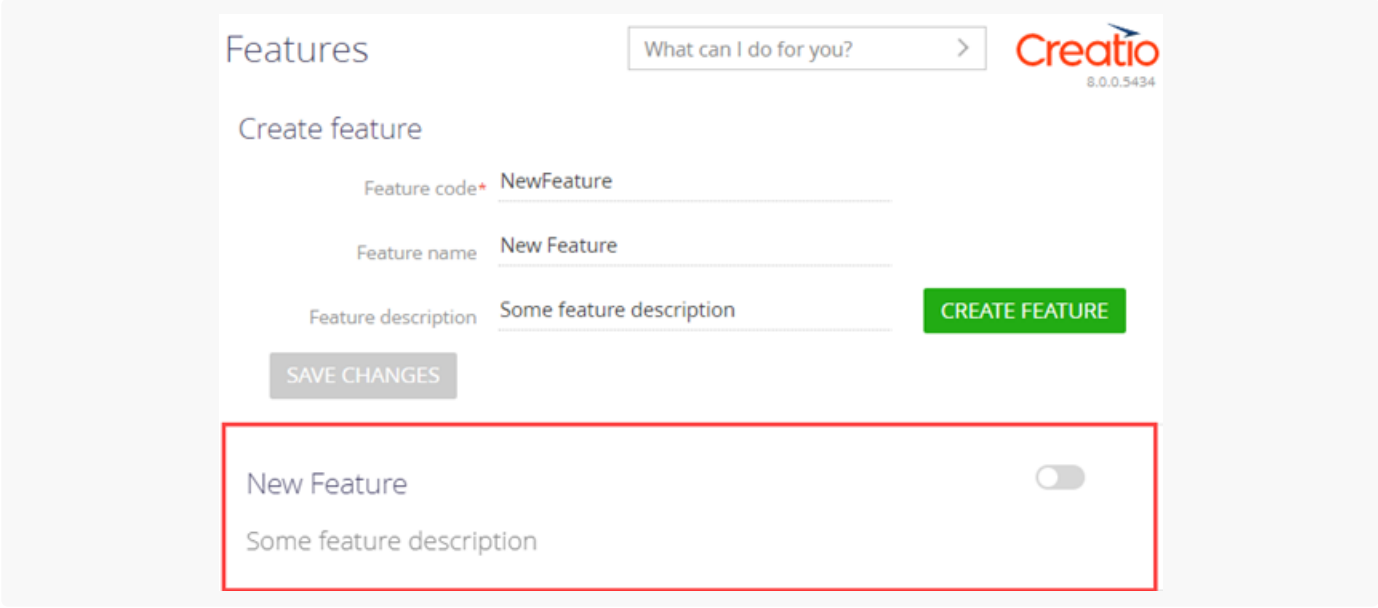
   b. Fill out the **properties of the feature to add**:

   - Enter the custom feature code in the [*Feature code*] property. Required.
   - Enter the custom feature name in the [*Feature name*] property.
   - Enter the custom feature description in the [*Feature description*] property.



   f. Click [*Create feature*].

   As a result, Creatio will add the custom feature to the [*Features*] page list.



2. Implement **a custom feature in the source code**. Implement the custom feature in the block of the conditional operator that checks the feature status (i. e. the `[FeatureState]` column value from the `[AdminUnitFeatureState]` database table).

   You can implement a custom feature in several **ways**:

   - In the front-end. To do this, follow the guide in a different section: Implement a custom feature in the front-end.
   - In the back-end. To do this, follow the guide in a a different section: Implement a custom feature in the back-end.

Implement a custom feature in the front-end

1. Create a view model schema. To do this, follow the guide in a separate article: Client module.
2. Add the source code in the Module Designer. You can use the template below.

   Template to implement a custom feature

   ```
   /* Method that defines the additional feature. */
   someMethod: function() {
       /* Check if the feature is connected. */
       if (Terrasoft.Features.getIsEnabled("FeatureCode")) {
           /* Implement the additional feature. */
           ...
       }
       /* Implement the method. */
       ...
   }
   ```

   The `getIsFeatureEnabled` method of the `BaseSchemaViewModel` base view model schema lets you streamline the workflow. As such, you can replace the `Terrasoft.Features.getIsEnabled` method with the `this.getIsFeatureEnabled("FeatureCode")` method.
3. Click [*Save*].

Refresh the page to add the custom feature to the client code and display the feature on the browser page.

Implement a custom feature in the back-end

1. Create a [*Source code*] schema. To do this, follow the guide in a separate article: Source code (C#).
2. Add the source code in the Source Code Designer.

   The `Terrasoft.Configuration.FeatureUtilities` class provides a set of extension methods to the `UserConnection` class. These methods let you use the `Feature toggle` functionality in the source code schemas of the Creatio back-end. You can use the template below.

   Template to implement a custom feature

```
...
/* The namespace that lets you toggle a feature. */
using Terrasoft.Configuration;
...
/* The method that implements an additional feature. */
public void AnyMethod() {
    /* Check if the feature is connected. */
    if (UserConnection.GetIsFeatureEnabled("FeatureCode")) {
        /* Implement the additional feature. */
    }
    /* Implement the method. */
    ...
}
```

3. Set the feature status by calling the `SetFeatureState()` method. You can use the template below.

Template to call the `SetFeatureState()` method

```
UserConnection.SetFeatureState("FeatureCode", FeatureState);
```

FeatureUtilities class C#

Medium

The `Terrasoft.Configuration` namespace.

The `Terrasoft.Configuration.FeatureUtilities` class provides a set of extension methods to the `UserConnection` class. These methods let you use the `Feature Toggle` functionality in the source code schemas of Creatio back-end. The `FeatureUtilities` class also declares the enumeration of feature statuses (the `[FeatureState]` column of the `[AdminUnitFeatureState]` database table).

Note. Use the .NET class reference to access the full list of the constructors, methods and properties, parent classes, and implemented interfaces of the `UserConnection` class.

Methods

`static int GetFeatureState(this UserConnection source, string code)`

Returns the feature status.

Parameters

source	The user connection.
code	The feature code.

`static int GetFeatureState(this UserConnection source, string code, Guid sysAdminUnitId)`

Returns the feature status.

Parameters

source	The user connection.
code	The code of the feature.
sysAdminUnitId	The unique record ID.

`static bool GetIsFeatureEnabledForAnyUser(this UserConnection source, string code)`

Returns the feature status for any user.

Parameters

source	The user connection.
code	The code of the feature.

`static bool GetIsFeatureEnabledForAllUsers(this UserConnection source, string code)`

Returns the feature status for all users.

Parameters

source	The user connection.
code	The code of the feature.

`static Dictionary<string, int> GetFeatureStates(this UserConnection source)`

Returns all feature statuses.

Parameters

source	The user connection.

```
static List<FeatureInfo> GetFeaturesInfo(this UserConnection source)
```

Returns data about all features and their statuses.

Parameters

source	The user connection.

```
static void SetFeatureState(this UserConnection source, string code, int state, bool forAllUsers
```

Sets the feature status.

Parameters

source	The user connection.
code	The code of the feature.
state	The new status of the feature.
forAllUsers	Sets the status of the feature for all users.

```
static void SafeSetFeatureState(this UserConnection source, string code, int state, bool forAllU
```

Sets the feature status or creates the feature if it does not exist.

Parameters

source	The user connection.
code	The code of the feature.
state	The new status of the feature.
forAllUsers	Sets the status of the feature for all users.

```
static void CreateFeature(this UserConnection source, string code, string name, string descripti
```

Creates the feature.

Parameters

source	The user connection.
code	The code of the feature.
name	The name of the feature.
description	The description of the feature.

`static bool GetIsFeatureEnabled(this UserConnection source, string code)`

Checks if the feature is turned on.

Parameters

source	The user connection.
code	The code of the feature.

`static bool GetIsFeatureEnabled(this UserConnection source, string code, Guid sysAdminUnitId)`

Checks if the feature is turned off.

Parameters

source	The user connection.
code	The code of the feature.
sysAdminUnitId	The unique record ID.

FeatureState enumeration

The `FeatureState` enumeration defines the feature statuses (the `[FeatureState]` column of the `[AdminUnitFeatureState]` database table).

Disabled	0	The feature is turned off.
Enabled	1	The feature is turned on.
Established	2	The feature is established.