Whenever a user saves a macro expression, the system automatically adds a security signature. Signatures prevent macros from accessing or displaying data for which the author is not authorized.

Every macro signature contains an identifier of the macro's author and a hash of the given expression. To increase the level of security, the hash function used when creating macro signatures appends a salt to the input (a sequence of additional characters). The salt value depends on the configuration of the application's environment, so the signatures are in most cases not valid when deploying macros to other instances of Kentico.

You can recognize signed macro expressions by the **#** character, which the system automatically inserts before the closing **%}** parentheses when saving the text containing the macro.

```
{% CurrentUser.Children["cms_category"][0].CategoryName #%}
```

If the execution of the macro requires any permissions, the system resolves the macro only if the **user specified by the signature** has the appropriate permissions.

Permissions are only checked when resolving macro expressions that:

- access objects that are members of another object, for example: *{% CurrentDocument.DocumentPageTemplate %}*
- access collections of objects, for example: *{% CurrentUser.Children["cms_category"][0].CategoryName %}*

The system carries out a security check for each access to any collection, not just for the final macro result.

Unsuccessful security checks prevent the system from resolving macros. If you encounter such problems, you can view the event log or the macro debug log, which provide information about performed security checks and their results.

Tip: You can confirm whether your macros have valid signatures by searching for expressions in **System -> Macros -> Report**.

See: Searching for macros

Assigning macro signature identities

Every macro signature contains an identifier of the user who entered and saved the expression. By default, the identifier is the user name.

Alternatively, administrators can assign special macro signature identities to users, which then replace the user name in the signatures of all macro expressions created by the given users. The same macro identity can be shared by any number of users. Each identity has an *effective user*, which the system checks instead of the original user when evaluating permissions while resolving macro expressions.

The following examples show how a signed macro expression is stored in the database, including the full signature (both types):

- User name: *{%CurrentSite.SiteName|**(user)administrator**|(hash)9056b7b76629a47a660c40cc2e5b0d92a13d0f9bce178847ca412c3a585552a1%}*
- Macro identity: *{%CurrentSite.SiteName|**(identity)GlobalAdministrator**|(hash)e3402fe14374ab15d119dddbb6c831ce3d2ef55bd8e70ce8eb80994f4e6ad18b%}*

Signature identities can be useful in environments where you transfer data containing macro expressions between multiple Kentico instances, for example when using staging or continuous integration. With the default user name signatures, you may encounter problems with invalid macros if you do not have exactly the same set of users across all instances – macro signatures are not valid if the signed user does not exist on the given instance. By assigning an identity with the same name to users across different instances, you can avoid the need to synchronize all users across all instances or manually re-sign macros after transferring data.

Security information

- Only users with the *Global administrator* privilege level can manage macro signature identities and assign them to users.
- Identities are not suitable for highly secure environments that require signatures to identify the exact author of each macro. If multiple users share the same macro identity, signatures cannot be used to determine which of the users is the author.

To create a macro signature identity:

1. Open the **System** application.
2. Select the **Macros -> Identities** tab.
3. Click **New macro identity**.
4. Enter an **Identity name**.
5. Assign an **Effective user**.
6. Click **Save**.

Notes

- The *Identity name* value is added into macro signatures instead of the user name of the author. You ***cannot change*** the name of a macro identity after you save it (the change would make macro expressions signed by the given identity invalid).
- If you do not assign a valid *Effective user*, the system evaluates macros signed by the identity with the restricted permissions of a public user.

To assign a macro identity to a user:

1. Edit the required user in the **Users** application.
2. On the **General** tab, select a **Macro signature identity**.
3. Click **Save**.

All macros entered by the user are now signed with the selected identity instead of the user's name. This does NOT retroactively change the user's signatures for already existing macro expressions. You cannot assign more than one macro identity to each user, but the same identity can be shared by any number of users.

Disabling the security signature for specific macros

You can stop the system from adding macro signatures by manually adding the **@** character before the closing %} sequence of a macro:

```
{% CurrentPageInfo.DocumentName @%}
```

Unsigned macros are always evaluated with the permissions of a public user. As a result, expressions that require any permissions will not resolve correctly when unsigned.

The advantage of unsigned macros is that they do not store an identifier of the author or a security hash. This allows you to:

- Add macros into fields that have a limited character count (unsigned macro expressions have the same internal length as the visible number of characters)
- Create macros that are easier to deploy to other Kentico instances (they do not rely on a specific user or identity, and the environment's hash salt value)

Configuring the hash salt for macro signatures

The system appends a salt to the input of the hash function that creates macro signatures.

By default, new instances of Kentico use a randomly generated GUID as the hash salt. The installer sets the custom salt value through the **CMSHashStringSalt** key in the *appSettings* section of the web.config file. Instances without this web.config key use the application's main database connection string as the salt (the exact **CMSConnectionString** value in the web.config file). We strongly recommend using a custom salt (CMSHashStringSalt).

A custom salt provides the following benefits:

- Stability (you do not need to re-sign macros if your connection string changes)
- You can use the same salt for other instances of Kentico, which makes deployment easier and allows content staging of macros

The best option is to set the hash salt value before you start creating content for your website.

> **Warning**
>
> Changing the salt causes all current hash values to become invalid, including the signatures of macros. Having invalid macros throughout the system may lead to problems on your website, and also ***prevents the Kentico administration interface from working correctly***. You need to re-sign macros immediately after changing the hash salt value.
>
> In addition to macro signatures, the system uses the **CMSHashStringSalt** value for other hash functions. Changing the hash salt on a website that already has defined content may break dialog links and images on your website. If you encounter such problems, you need to re-save the given content (the system then creates the hashes using the new salt).

To change the salt value for your application, edit the value of the **CMSHashStringSalt** key in your web.config file. You can use any string as the value, but the salt should be random and at least 16 characters long. For example:

```
<add key="CMSHashStringSalt" value="e68b9ad6-a461-4707-8e3e-ece73f03dd02" />
```

Re-signing macros

If the hash salt value used by your application changes, the security signatures of existing macro expressions become invalid. For example, you may encounter problems with unresolved macros:

- After setting a new custom salt via the **CMSHashStringSalt** web.config key.
- When using Content staging to transfer data containing macros to an instance of Kentico with a different hash salt.
- If your application does not have a custom hash salt, and the connection string has changed (for example when moving to a different server or after setting a new database password).

To re-sign individual macros, find the expression in the user interface and save the value (you can use the macro report tool to find the location). The system creates a new signature for the macro based on the application's current environment.

You can also repair invalid macro signatures by re-signing all macros in the system:

1. Go to **System -> Macros -> Signatures**.
2. Fill in the **Old salt** field.

If you leave the **Sign all macros** option *disabled*, the system checks the original signatures before re-signing macros. Only macros that have a valid signature under the old salt are re-signed and the identifiers of the macro authors remain unchanged. You need to enter the old salt that was used to generate the security hash of the existing macro expressions in the system.

- If your application uses a custom hash salt, enter the original value of the **CMSHashStringSalt** web.config key.
- If the original hash salt was a connection string, enter the value in format:
  Persist Security Info=False;database=DBName;server=ServerName;user id=DBUser;password=pwd; Current Language=English;Connection Timeout=240;

If you enable **Sign all macros**, the macro re-signing process skips the signature integrity check and creates new signatures for all macros. You do not need to enter the old salt value in this case. The new signatures contain the name or identity of the user who started the re-signing procedure.

Security warning: With the *Sign all macros* option enabled, the resigning process also includes macros that are unsigned or have invalid signatures. If your system's data contains invalid macros added by users with insufficient authorization, such macros receive valid signatures and represent potential security threats.

3. Type in the **New salt** that will be used to re-sign the macros.
   - By default, the field automatically loads the current application's hash salt value. To enter a different value, disable the **Use current salt** option.

     Important

     In order for the system to correctly validate macro signatures, the new hash salt value must match the application's current salt (the **CMSHashStringSalt** key value or connection string). Only change the **New salt** value if you are planning to change your application salt.

4. Click **Update macro signatures**.

The system replaces the security signature in all occurrences of macros based on the new salt.