SQL Sentry Message Editing

Last Modified on 06 December 2021

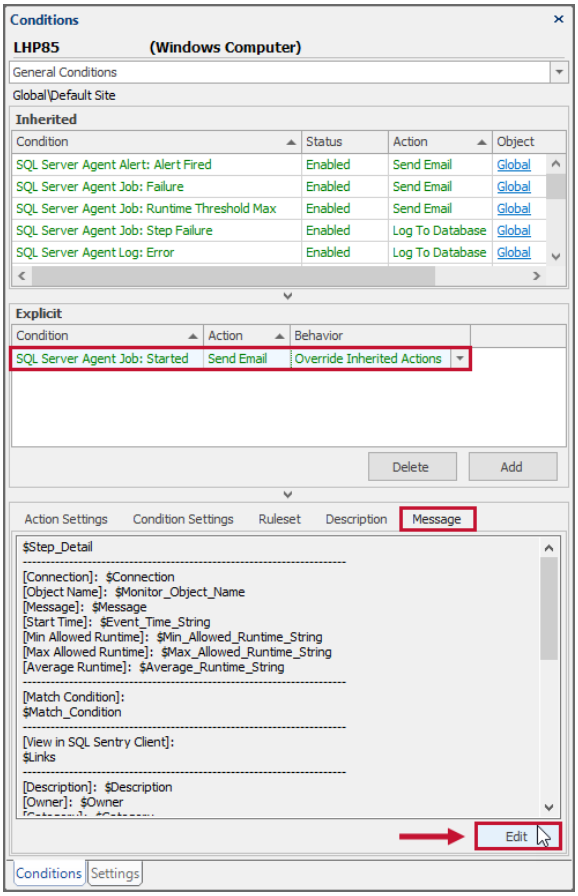
This feature allows you to fully customize the messages that are generated by SQL Sentry. Add or remove information, change how the information is formatted, and change the order that the information is displayed.

Additional Information: See these blogs for more details and examples:

- Customizing Your Alert Emails in SQL Sentry
- Customizing SQL Sentry Message Text

Access the Message Editor

To access the **Message Editor**, select the **Condition** that you'd like to change in the **Conditions** pane (**View** > **Conditions**), select the **Message** tab on the bottom of the pane, and then select **Edit**.



How does customization work?

Understanding XML

To understand how customizing messages works in SQL Sentry, it's important to understand the following about XML:

- The format of the XML that describes the format of the messages
- The elements added to the XML
- How these elements are related

At the root level (the **Message** element), each message formatted XML document (called a **Template**) uses the **Imports** attribute to inherit from another template. That template inherits from another template and so on, creating a chain of **Parent/Child** relationships. These chains are used extensively throughout the **Template** files to reduce the repetition of reused elements. The middle pane of **Message Editor** combines these chains into the single, complete XML document that's used to generate the final message. Sometimes the finale **Template**, sometimes referred to as a **Leaf Template**, has very little information because almost all of the XML is in the inherited **Templates**. In these cases, it can be useful to look at the complete XML to understand what's in the message and how to change it.

Inside the root message element, there are three parts to a template:

- the **Format** element
- the **TokenSets** element

- the **Body** element

See the following table for more information about each element and its contents.

Element Name	Element contents
Format element	Contains definitions for section, item, and token styles.
tokenSet element	Contains a list of all of the **tokenSets** associated with the **Template**.
Body element	Contains all of the sections, items and tokens that describe the message format itself.

Note: All of these elements are combined with their imported parent elements during the import process so that inheritance is used.

The **Format** element is a list of style definitions that are referred to in the body of the **Template**. **Leaf templates** often don't contain a **Format** element because they rely on a group of standard inherited styles. Add a format element if one doesn't exist or add to an existing one to define new styles. Only one format element is allowed per template.

The **tokenSets** element is a list of all of the **tokenSets** associated with a **Template**. A **tokenSet** is a logical grouping of tokens that exists in SQL Sentry. You can't add **tokenSets** to messages because the data for the tokens in the **tokenSet** needs to be created and assigned in the program. A list of all of the tokens in the **tokenSets** is available in the **Message Editor**. Only one **tokenSets** element is allowed per template.

The **Body** element describes all of the things that a message is made of. These things can be categorized as items (and things that inherit from items) and sections, which are groups of items. Only one body element is allowed per **Template**.

Items and sections

An **item** is the base element in the body section. **TokenItems**, **textItems**, **breakItems** and **blankLineItems** are all specialized types of items. See the following table for more information about these items:

Item/Section Name	Item/Section Description
TokenItems	Special items that represent a name/value pair. A common example in SQL Sentry messages is **[Condition]: SQL Server: Blocking SQL**. The name (**Condition**) is a description of the value (**SQL Server: Blocking SQL**). The value for these **tokenItems** is set during the program execution.
MultiTokenItems	Special **tokenItems** that convert one or more **tokenItems** into a single token with specialized formatting. They have a converter attribute that's the name of a registered converter, and they reference one or more **tokenItems**.

Item/Section Name	Item/Section Description
TextItems	A special type of item that displays a literal text value.
BreakItems	A specialized **textItem** with pre-set text for ease of use. The **breakItem** takes whatever the current section format break is and places that in the message.
blankLineItems	A specialized **textItem** with pre-set text for ease of use. The **blankLineItem** inserts a blank line in a message.
section element	A way to group the above items. Sections can contain other sections. Nested sections are indented using the **indentStyle** from the current **sectionStyle**.

Item and Section Attributes

All items have the following attributes:

Item/Section Attribute	Description
id	A unique string identifying the item.
visible	A boolean that determines if the item is visible in the message.
before	The id of the item, token or section that this item should be placed before.
after	The id of the item, token or section that this item should be placed after.
requires	The id of an item, token or section that must be visible for this item to be visible.
indent	A boolean that determines if the item should be indented.
itemStyleId	The id of the itemStyle (format) that should be used for this item.

TextItems have one additional attribute:

TextItem Attribute	Description
text	The text that will be displayed in the final message.

TokenItems have additional attributes:

TokenItem Attributes	Description

TokenItem Attributes	Description
name	The token name to be displayed in the message. If none is assigned, a default name based on the item id is displayed.
hideIfNoValue	A boolean that when set to true, hides the token if no value is assigned.
tokenItemStyleId	The id of the tokenStyle (format) that should be used for this token.

MultiTokenItems have all of the tokenItem attributes plus an additional attribute and element:

Attribute:

MultiTokenItems Addittional Attribute	Description
converter	The name of the registered converter.

Element:

MultiTokenItems Addittional Element	Description
token	An element that represents a token to be passed to the converter. It has a single attribute—id—that's the id of a valid token. MultiTokenItems may have one or more token elements.

Sections have similar attributes to items:

Section Attribute	Description
id	A unique string identifying the item.
visible	A boolean that determines if the item is visible in the message.
before	The id of the item, token or section that this item should be placed before.
after	The id of the item, token or section that this item should be placed after.
requires	The id of an item, token or section that must be visible for this item to be visible.
indent	A boolean that determines if the item should be indented.
sectionStyleId	The id of the sectionStyle that should be used for this section.

itemStyleId Section Attribute	The id of the itemStyle (format) that should be used for this item. Description
tokenItemStyleId	The id of the tokenStyle (format) that should be used for this token.

Styles

Styles are a very important part of how message customization works. There are three types of styles:

- Section styles
- Item styles
- Token styles

Note: Styles are defined in the format section of the XML. The styles are then referenced by their IDs in the sections, items and tokens. These styles use format strings to identify how to change an item, token or section when the message is built. The format strings use keywords to represent the data so you can add formatting content around them.

For example, token styles use **%itemname%** to represent the name of a token and **%itemvalue%** to represent its value. To create the finished message value of **[Condition]: SQL Server: Blocking SQL**, use the format string: **[%itemname%]: %itemvalue%.** If the desired finished value is **Condition-->SQL Server: Blocking SQL** the format string needs to be changed to **%itemname%-->%itemvalue%**. You can also add whitespace characters such as newline (\r\n) or tab (\t).

Style Attributes

Section styles (sectionStyle elements) have three attributes:

Section Style Attribute	Description	Keyword	Example
id	A unique string identifying them.	NA	NA
indentStyle	A format string to determine how a nested section should be indented.	%sectionitem% represents each item in the section.	%sectionitem% places four spaces before each item in the section.
formatStyle	A format string to determine how a section should be formatted.	%sectionitems% represents the entire section as a whole.	%sectionitems% ----------\r\n places a line of dashes as a break after the section.

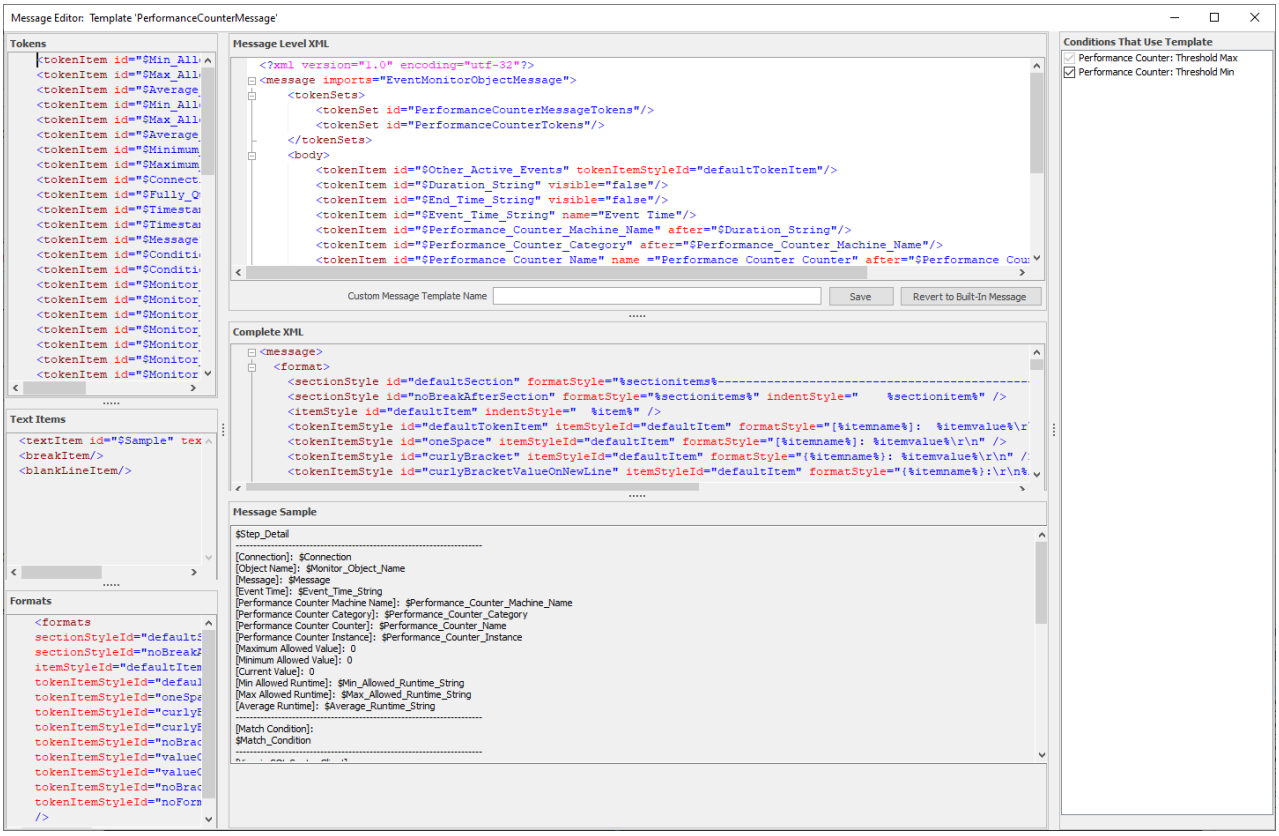
Item styles (itemStyle elements) have two attributes:

Item Style Attribute	Description	Keyword	Example
id	A unique string identifying this style.	NA	NA
indentStyle	A format string to determine how an item should be indented (when the indent attribute is set to true).	%item% represents the item.	\t%item% places a tab character in front of the item.

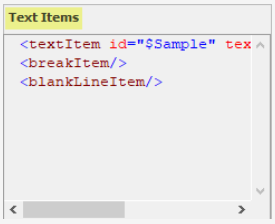
Token styles (tokenItemStyle elements) extend item styles and have three additional attributes:

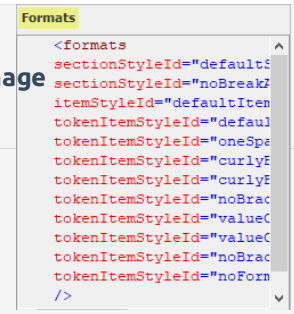
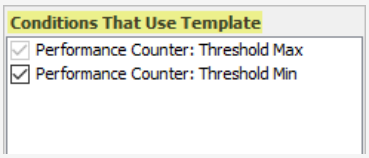
Token style Attribute	Description	Keyword	Example
formatStyle	A format string to determine how a token should be formatted.	%itemname% represents the token name; %itemvalue% represents the value of the token.	[%itemname%]: %itemvalue% creates the standard "[Name]: Value" token style.
itemStyleId	The id of the itemStyle this tokenStyle references. Because a token is an item underneath, the format needs to specify the item style as well.	NA	NA
itemValueStyle	A .NET format string for the item value. This string gives you the ability to format the token value itself using .NET format string which is useful for date or number formatting.	NA	0.## formats a decimal value to a maximum of two decimal places.

Customizing a Message



Sections of the Message Editor

Message Editor Section	Description	Image
Tokens pane	Displays a list of tokens that are available for use in the message.	
Text Items pane	Contains various XML elements that can be copied and pasted into the **Message Level XML** pane to add custom text and line breaks to the message.	

Message Editor Section	Description	Image
Formats pane	Contains XML attributes that can be used with an element to adjust the formatting of the element.	
Message Level XML pane	The **Message Level XML** pane is where you can edit the message. Changes made in this pane are reflected in the **Complete XML** and **Message Sample** panes.	
Complete XML pane	Displays the final product of the template displayed in the **Message Level XML** pane plus all of the templates that are imported by it. You can use this pane as guideline when making changes to the **Message Level XML**.	
Message Sample pane	Displays a mock-up of what the complete XML becomes.	
Conditions That Use Template pane	Displays a list of conditions that are currently using the template you're editing. The check boxes give you the ability to quickly save a new template and apply it to multiple conditions.	

Editing a Message

Adding Additional Information

New items can be added inside or outside of a section. Available nodes are listed in the **Tokens** pane and can be copy-and-pasted, highlighted-and-dragged, or typed directly into the editor. Set the **before** or after **attributes** to place the item in a particular spot in the message.

Removing or Hiding an Item

If an item is only in the **Message Level XML**, you can delete that node from the XML. If the item is in an imported template, you can hide the item in the finished message by setting the **visible** attribute to **False**. You can also set the **hideIfNoValue** attribute to **True** which hides the item if there's no value assigned to it.

Moving an Item

Items are displayed in the final message as they are placed in the XML, top to bottom, unless **before** or **after** attributes are placed on the items or sections.

Note: This quickly gets confusing with the inheritance chain, so the easiest way to move an item is to set its **before** or **after** attribute.

Adding, Hiding, or Moving a Section

Sections are a lot like items (by design). Add one by typing in a new

, hide one by setting its **visible** property to **False** and move one by setting its **before** or **after** attribute.

Changing a Token Name

Change the displayed name of a token by setting its **name** property.

Indenting an Item

Indent an item by setting the **indent** attribute for an item to true.

Note: All items have an inherited default indent style.

For custom indents, create a new **itemStyle** with your custom indent and apply that style to a body, section or item.

Formatting a Token

Each template has several inherited token styles that are listed in the **Formats** pane of the **Message Editor**. Copy-and-paste or highlight-and-drag these attributes into the **body**, **section** or **token item**. For custom formatting, define a new **tokenItemStyle** in the **Format** section and set the **formatStyle** string to suit your needs. Reference this new style by **id** in the attributes of the **body**, **section** or **tokenItem**.

Example:

t**okenItemStyle** definition to get a token like **{Condition}-->Name of Condition**:

Formatting a Section

Each template has at least one inherited section style that's listed in the **Formats** pane of the **Message Editor**. Copy-and-paste or highlight-and-drag the attribute into the **body** or **section**. For custom formatting, define a new **sectionStyle** in the **Format** section and set the **formatStyle** string to suit your needs. Reference this new style by **id** in the attributes of the **body** or **section**.

Example section:

Style definition to replace the standard line of dashes after a section with a line of equal signs:

Formatting a Token Value

Define a .NET format string to format a token value itself. This is especially useful in formatting dates and number values. Define a new **tokenItemStyle** in the format section, set up the **formatStyle** and **itemStyleId** attributes (or copy them from another style) and then set the **itemValueStyle** to a .NET format string.