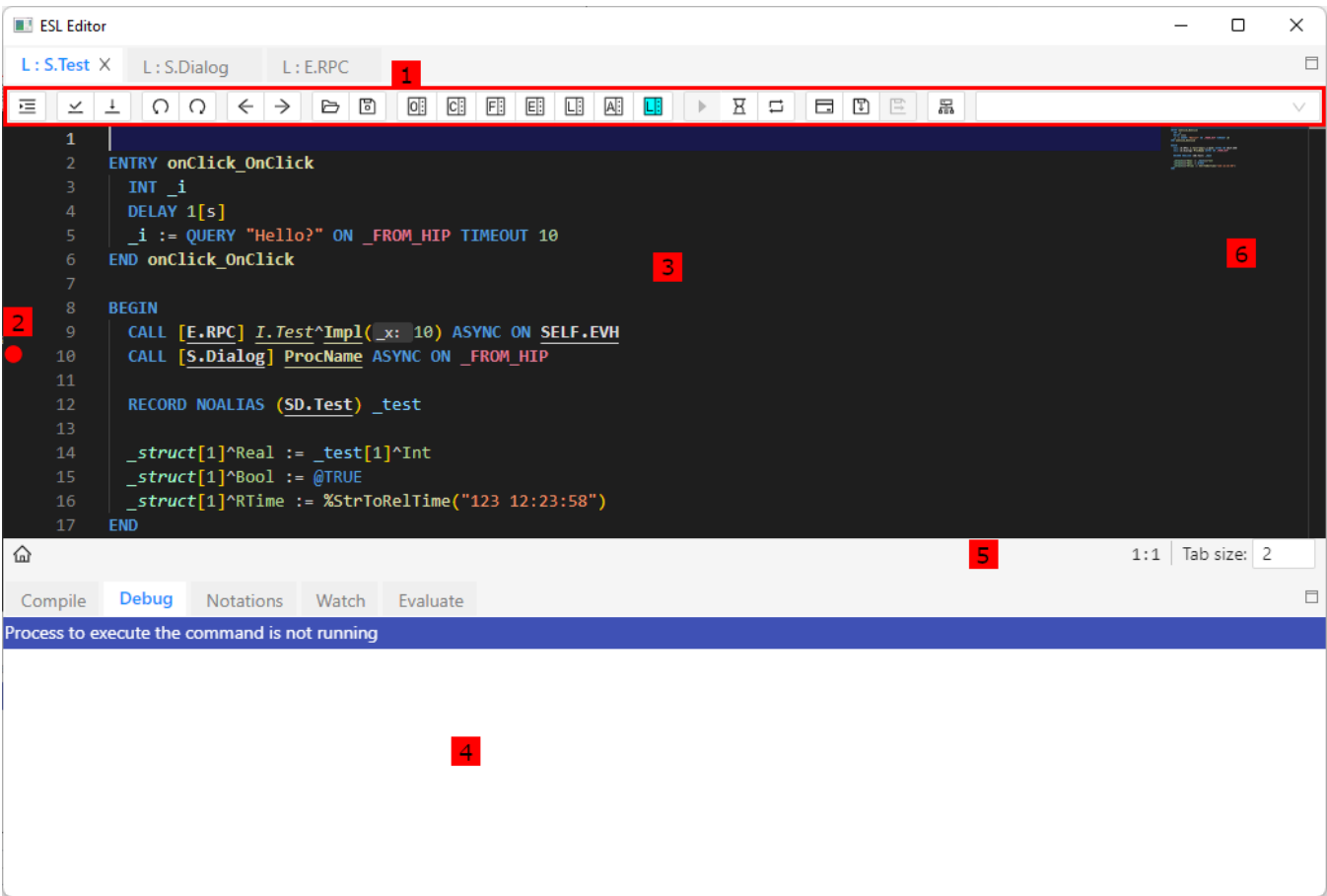




Script Editor

Script editor

The figure below shows the ESL script editor window. Each script is opened in its own tab.



1	Toolbar.
2	The breakpoint is set when debugging a script  .
3	The source text of the script - individual actions that form the script.
4	<div>The part of the script editor that contains the following displaying:<ul style="list-style-type: none">• <i>Compile</i> - errors that occurred during the compilation (syntactic, semantic check),• <i>Debug</i> - debugging logs,• <i>Notations</i> - comments existing in the script,• <i>Watch</i> - values of local variables• Evaluate - expression evaluationTo resize this part and the area for editing the script text point the mouse cursor to the margin between the parts and when the mouse cursor changes its shape to , then press the left mouse button and drag the border (left mouse button has to be still pressed) to the desired position.</div>
5	Status bar
6	Minimap for fast navigation in code

Notes:

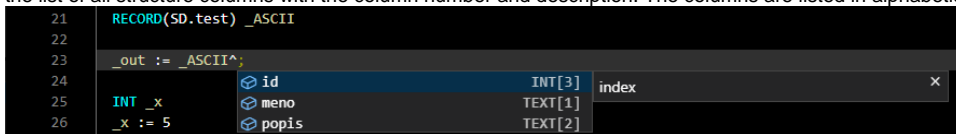
When the script editor is closed all the tabs will be closed and the window will hide.

The rules of the tab closing:

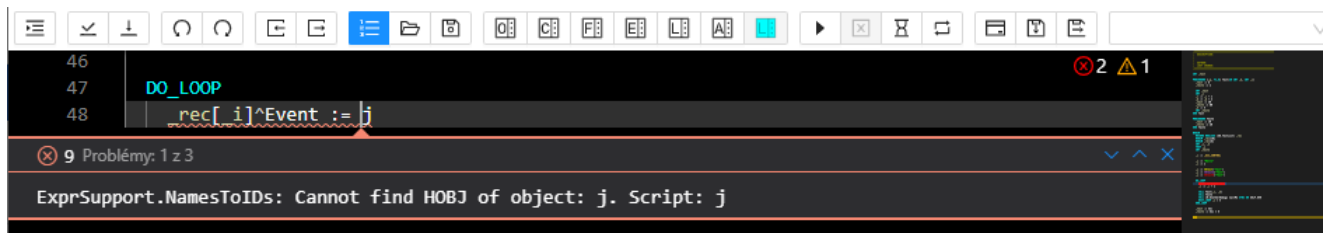
- script of the active picture (opened in [D2000 GrEditor](#)) does not display the question of whether to save the script providing that there were made some changes (the script is occupied by the edited picture),
- script of the event (opened in [D2000 CNF](#) or [D2000 GrEditor](#)) displays the question of whether to save the script providing that there were made some changes

ESL editor features:

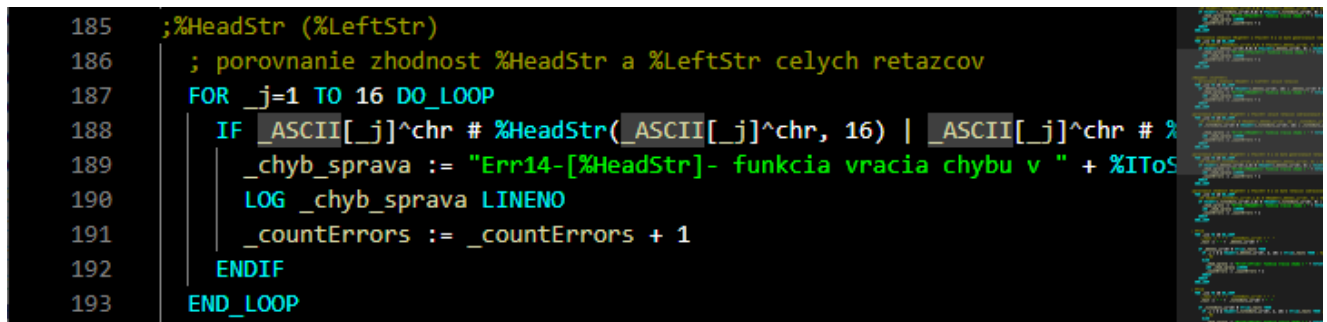
- Automatic color differentiation of keywords: **END**.
- Automatic color differentiation of texts: correctly written: **"Text"**, incorrectly written: **"Zly text"**.
- Semantic coloring - different colors for local, global, predefined and scheme variables, different colors for objects, procedure names etc...
- Static code analysis
- After a successful **Syntax check**:
 - Typing the symbol '^' after the name of an object of **Structured variable** type or after the name of a local variable of **RECORD** type shows the list of all structure columns with the column number and description. The columns are listed in alphabetical order.



- Pointing the mouse cursor to a local variable, or a structured variable item displays information about its type and the place where it is declared (in the debugging mode, the value of the variable is shown).
- After an unsuccessful **Syntax check** or **Compilation**, double-clicking an error from the lists of errors (5) moves the mouse cursor to where the error occurred. Another possibility for navigation between errors or warnings is by clicking on an icon in the upper right corner of the editor, which is shown when there are some errors or warnings. After clicking on this icon, a line with an error is shown with an error description. Every other click on the icon shows the next error/warning.



- Pressing **F1** (help) when the cursor is pointed to the name of a function/action opens the help document for the given function/action/picture event.
- Quick navigation in a script using a minimap (code outline). To move quickly in the script, you can click and drag the gray part in the minimap to move the currently displayed part of the script in the editor. The minimap contains highlighted pieces of code, such as the line with the cursor position. In the case of a failed syntax check or compilation, error lines are highlighted here in red, or warning lines are highlighted in orange. After placing the cursor in the editor on a local variable or procedure, all occurrences of the given variable/procedure within the script are displayed in gray behind the slider.



- Automatic suggestions occur under the following conditions:
 - After entering the "\"" character after the object or variable, when a list of object attributes is displayed
 - After entering the character "^" if this character is located after an object of type structured variable or after a local variable of **RECORD** type
- Autocompletion and suggestions in the editor after pressing the **CTRL + SPACE** keyboard shortcut. After pressing **CTRL + SPACE** again, it is possible to display a more detailed description (function - description of function, action - syntax for action, variables - description of variable, constants - description of constant). It is possible to navigate in the list of suggestions using the arrows, after pressing **ENTER** the given action, variable, function, constant, or attribute will be added. Based on the conditions described below, it contains a list of proposals:
 - If there is a space in front of the cursor in the editor, actions, variables, functions, and constants are suggested
 - If the cursor is preceded by a text beginning with "%", the functions are suggested, and the list of suggestions is filtered based on the text after "%"
 - If the cursor is preceded by a text beginning with "_", the proposed variables (local, global, and schema variables in the case of a schema script) are filtered, and the list of suggestions is filtered based on the text before the cursor.
 - If the cursor is preceded by a text beginning with "@", constants are suggested, and the list of suggestions is filtered based on the text after "@"
 - If the cursor is preceded by a text beginning with a-zA-Z, action keywords are suggested

- If the line starts with the keyword of the action, the current options will be displayed first in the list, followed by all the options for completion provided by the editor (variables, functions, constants).
- If an object identifier (schema object or system event server identifier) is specified for the CALL action, RPC procedures and interfaces that are defined for the object are proposed.

```

27
28  _out := %Time
29
30  %TimeFromItems(INT, INT, INT,...  TIMA  TEXT %TimeToStrEx(TIMAS, TEXT, NONE)
31  %TimeToStr(TIMAS, TEXT)          TEXT
32  %TimeToStrEx(TIMAS, TEXT, NONE)  TEXT  Converts absolute time to text string
33  %TimeToStrMono(TIMAS, TEXT, IN... TEXT  by defined mask
34  %AddTime(TIMAS, TIMR, NONE)      TIMA
35  %HI_TimerEnable(BOOL)            BOOL
36  %ModTime(TIMAS, TIMR, NONE)      TIMR
37  %NewTime(TIMAS, INT, INT, INT,... TIMA
38  %RelTimeToStr(TIMR, TEXT)        TEXT
39  %SubTime(TIMAS, TIMR, NONE)      TIMA
40  %SubTimesLocal(TIMAS, TIMAS, NO... TIMR
    %SubTimesMono(TIMAS, TIMAS)    TIMR

```

```

23  CALL [E.SmallRPC]
24
25  CALL
26  CALL
27  CALL
28  CALL
29  Dec(INT _i)
30  Inc(INT _i)
31  Rand

```

- Automatic display of function parameters after entering the function name and the first parenthesis. The currently typed parameter is underlined and highlighted. The popup with the description of the function parameters can be displayed even after placing the cursor in the editor between the function parameters and pressing the keyboard shortcut CTRL + SHIFT + SPACE

```

9  PROCEDURE TestValue(IN INT _value, IN INT _status, BOOL _out)
10  IF _value = _status THEN
11  | _out := @TRUE
12  ELSE
13  | _out := @FALSE
14  ENDIF
15  END TestValue
16
17  RPC PROCEDURE CallMe TestValue(IN INT _value, IN INT _status, BOOL _out)
18  CALL TestValue(10, _st)

```

- After typing the "^" character after the interface name, a list of procedures that are defined for that interface is displayed. After selecting an item from the list, the procedure definition is added to the ESL Editor.

```

34  I.Interface^
35
36  END CallMe
37
38  BEGIN
    RegisterClient(IN TEXT, IN INT, IN INT, INT)
    SendMessage(IN INT, IN TEXT, INT)
    UnRegisterClient(INT)

```

- Preview the definition or references of a local variable and procedure, without the need to change the position in the script. By activating, the nested editor is displayed in the desired position with the possibility of editing, while previewing references, it is possible to gradually switch between individual occurrences in the right part. This functionality is invoked via the popup menu Peek Peek Definition (ALT + F12) or Peek Peek References.

```

502 ELIF ( _j = 5 & %GetStrItem( _text1, _j, ";" ) = %SubStr( _ASCII[ _j-1]^chr,1,11)) THEN
503 ELIF ( _j = 6 & %GetStrItem( _text1, _j, ";" ) = %SubStr( _ASCII[ _j-2]^chr,13,4)) THEN

Odkazy - Odkazy (18)
391 _text1:= _text1 + _txtInOut[_j]^chr
392 ;%FindStr
393 ; vsetky slova spojime do jednej premennej
394 _text := ""
395 _text1 := ""
396 FOR _j=1 TO 16 DO_LOOP
397 _text := _text + _ASCII[_j]^chr
398 _text1:= _text1 + _txtInOut[_j]^chr
399 END_LOOP
400 IF %LenStr( _text ) # 256 THEN
401 | _chyb_sprava := "Err22-[%LenStr]- funkcia vracia chybu. "
402 LOG _chyb_sprava LINENO
403 _countErrors := _countErrors + 1
404 ENDIF
405 FOR _j=1 TO 255 DO_LOOP ; postupne hladanie znakov z text1 v text2
406 IF 1/%FindStr( _text, %SubStr( _text1, _j, 1 ) ) < 18 THEN
504 ELIF ( _j > 6 & _j < 18 & %GetStrItem( _text1, _j, ";" ) = _ASCII[_j-2]^chr) THEN
505 ELSE

```

- Folding / unfolding parts of the script such as: procedure body, BEGIN-END, IF-ELSE, IF-ELSIF, IF-ENDIF, FOR-END_LOOP ... The icons for folding/unfolding are located to the right of the line number, while the icons for folding are shown only if we move the cursor over this part of the editor.

```

29 FOR _j=1 TO 16 DO_LOOP
30   _ASCII[_j]^chr := ""
31   FOR _i=0 TO 15 DO_LOOP ...
34 END_LOOP

```

- Automatic indentation
When writing the ESL script and moving to the new line, the ESL editor automatically indents the current line (according to the first non-zero line) and moves the cursor on the given position. The indent size is automatically detected from the current script, but can also be adjusted in the status bar (Figure area 6).

Features of automatic indentation:

- When moving to a new line after selected actions (eg RPC, PROCEDURE, PUBLIC, FOR, DO_LOOP, IF ...) in the ESL editor, the text is automatically indented by the defined size.
 - When moving to a new line after selected actions (eg END_LOOP, ENDIF ...) in the ESL editor, the text is automatically indented by a defined size to the left of the previous line.
 - When you press CTRL + K and then CTRL + F, the highlighted text is formatted according to the previous properties. The indent of the first line in the marked set of lines is decisive for this action.
 - Pressing SHIFT + ALT + F will format the entire text
- Column text selection is possible by pressing SHIFT+ALT and then by dragging the mouse

```

1 *****
2 ; DESCRIPTION:
3 ;
4 ;
5 ; AUTHOR:
6 ; LAST CHANGE:
7 *****

```

- Basic keyboard shortcuts:

Shortcut	Action
CTRL+F1	Display the complete list of actions together with keyboard shortcuts
F1	Display editor help, if the cursor is over the function name, help for the function is displayed
CTRL+SPACE	Show suggestions
CTRL+SHIFT+SPACE	Show quick info for procedure and function parameters

CTRL+K CTRL+C	Add a comment to the current line
CTRL+K CTRL+U	Delete a comment from the current line
CTRL+/ 	Add/remove a comment from the current line
SHIFT+ALT+A	Add/remove a comment from the current selection
CTRL+F2	Rename all occurrences
F2	Rename a symbol within its validity
CTRL+F	Search in the script
ENTER	Find another occurrence
SHIFT+ENTER	Find the previous occurrence
SHIFT+ALT+F	Format the entire script
CTRL+K CTRL+F	Format the selected area
CTRL+F12	Go to definition
CTRL+G	Go to line
ALT+F8	Go to the next problem
SHIFT+ALT+F8	Go to the previous problem
SHIFT+F12	Go to references
CTRL+SHIFT+O	Go to symbol - displays the option to go to functions, variables, parameters defined in the script
CTRL+SHIFT+F8	Check script syntax
CTRL+F8	Compile the script
CTRL+S	Compile and save the script
CTRL+1	Opens a list of D2000 system objects .
CTRL+2	Opens a list of predefined constants .
CTRL+3	Opens a list of functions .
CTRL+4	Opens a list of actions .
CTRL+5	Opens a list of local variables .
CTRL+6	Opens a list of value attributes of an object or local variable.
F8	Switch to debug mode

Popup menu

The popup menu containing these items can be displayed over the script source text (part 4 on the picture). Click by right mouse button or push the **Menu** key on the keyboard.

Go to Definition	Ctrl+F12
Go to References	Shift+F12
Go to Symbol...	Ctrl+Shift+O
Peek	>
<hr/>	
Rename Symbol	F2
Change All Occurrences	Ctrl+F2
Format Document	Shift+Alt+F
<hr/>	
Editor settings	
Show all instances	
<hr/>	
Cut	
Copy	
Paste	
<hr/>	
Command Palette	Ctrl+F1

- **Go to Definition** - cursor is automatically moved to the declaration of the identifier. When using "Go to definition" on the remote procedure ([RPC/PUBLIC](#)), the ESL editor automatically opens the script, which contains the definition of procedure, and moves the cursor on the definition.
- **Go to References** (SHIFT + F12) - a nested editor is displayed with the option to switch between the individual references of the local variable /procedure.
- **Go to Symbol...** (CTRL + SHIFT + O) - a selection box with filtering of all symbols (local variables, procedure parameters, procedures) is displayed, after selecting the symbol and pressing the ENTER key the cursor is moved to the symbol definition.
- **Peek** - here are two options:
 - **Peek Definition** (ALT + F12) - the nested editor is displayed at the position of the local variable/procedure definition.
 - **Peek References** - a nested editor is displayed with a reference to a local variable/procedure, with a list of individual references in the right part.
- **Rename Symbol** (F2) - renames the name of the local and global variable defined in the script within its scope.
- **Change All Occurrences** (CTRL + F2) - this allows replacing all occurrences of the specified text within the script.
- **Format Document** (SHIFT + ALT + F) - adjusts the formatting of the document, applying the currently set indentation.
- **Editor settings** - displays the ESL editor settings dialog (font and colors).
- **Show All Instances** - shows the list of all running instances of the edited [ESL script](#).
- **Cut** (CTRL + X) - copies the selected content to the clipboard and removes it from the editor,
- **Copy** (CTRL + C) - copies the selected content to the clipboard,
- **Paste** (CTRL + V) - pastes the contents of the clipboard.
- **Command Palette** (CTRL + F1) - displays all available actions in the editor

Editor settings

General settings

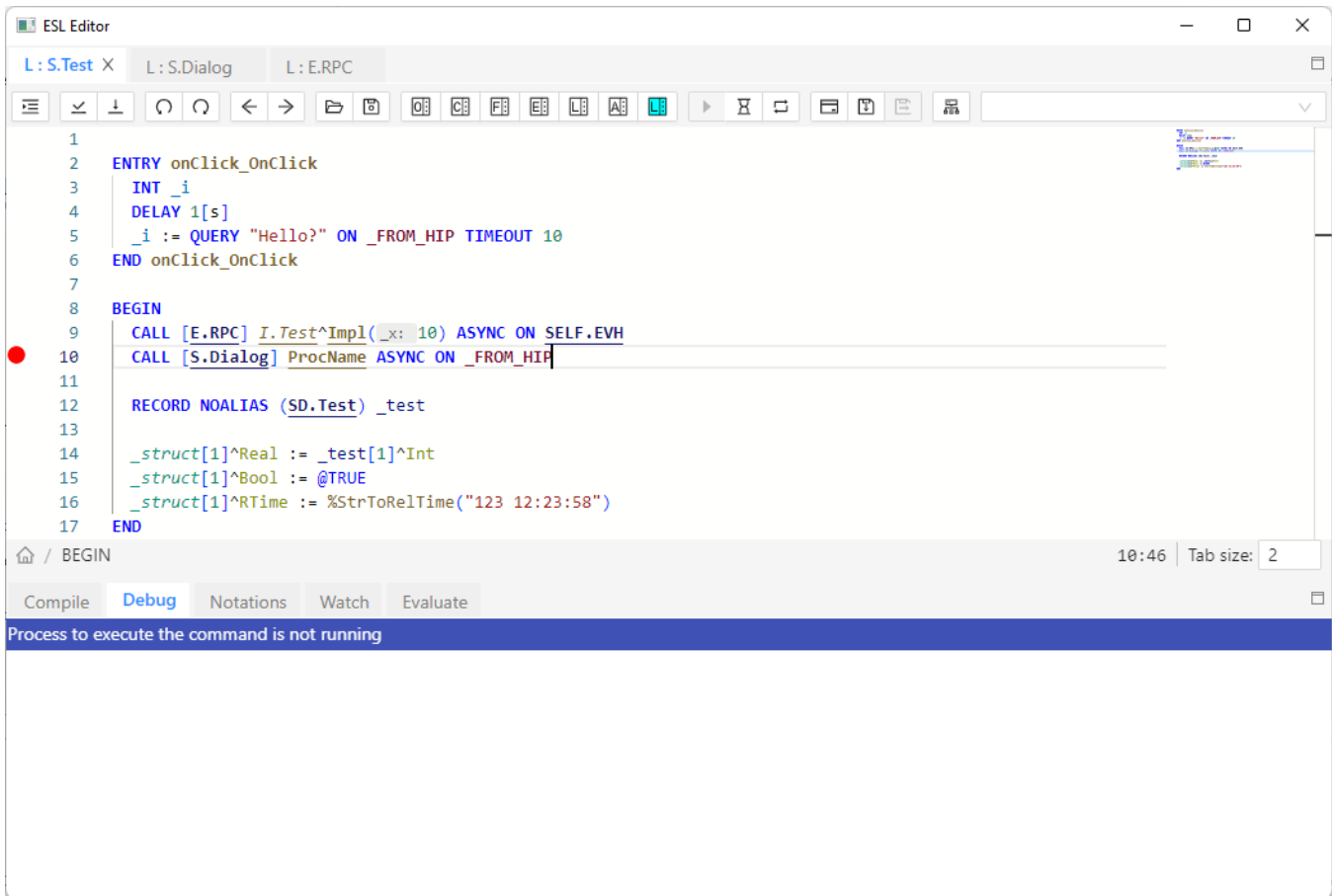
Allows user personalize editor by modifying various settings

Static analysis

Allow user enable/disable different types of code validations and set their severity. Severity affects how error is displayed in editor

Color settings

For better orientation in the ESL script, the editor uses a different coloring scheme for each type of the individual text tokens. There are three editor color themes - dark, light and high-contrast theme.



ESL Editor within *String* detects the references to the [dictionary](#). If some reference to a dictionary (key), which has not been defined yet, is identified, it will be colored as an *Error* (see the dialog above).

Example:

The key `D2_ActAlarm` exists in the dictionary, but `D2_ActAlarmAAAAA` does not exist.

```
16 _t := "D2_ActAlarm"  
17 _t := "... {!D2_ActAlarm} . text."  
18 _t := "... {!D2_ActAlarmAAAAA} . text."
```