Chapter

44 *Using the SDL Editor*

This chapter describes the functionality, menus, dialogs and windows of the SDL Editor. You can also find information about how to use the editor for creating and drawing SDL (Specification and Description Language) diagrams.

General

Editor Information

The editor described in this chapter handles SDL diagrams.

Another editor is capable of handling four different types of diagrams, namely the Object Model (OM) diagrams, State Chart (SC) diagrams, Message Sequence Chart (MSC) diagrams, and High-level MSC (HMSC). That editor is described in chapter 40, *Using Diagram Editors*.

SDL Diagrams

The SDL Editor can handle any number of SDL diagrams of any type. Virtually all of the Z.100 recommendation is supported.

The SDL Editor supports three kinds of SDL diagrams:

- Interaction Diagrams
- Flow Diagrams
- Overview Diagrams

The editing functionality that is available depends on what type of diagram the SDL Editor is handling.

This chapter describes the functionality the SDL Editor provides when you edit interaction and flow diagrams.

In Figure 350–Figure 352, you can find an example of interaction diagrams (in the example there is one system diagram and two block diagrams) and the resulting overview diagram.

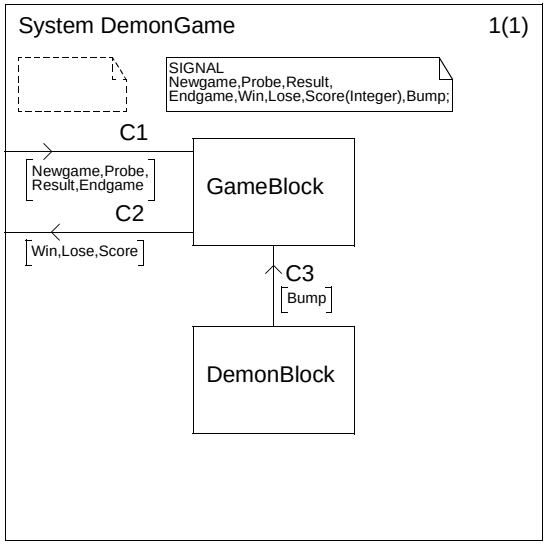


Figure 350: System diagram

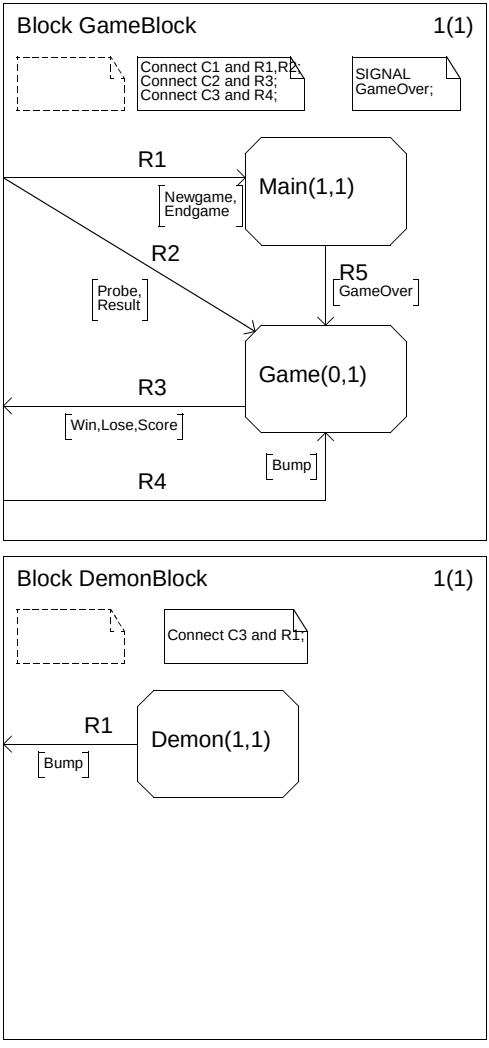


Figure 351: Block diagrams

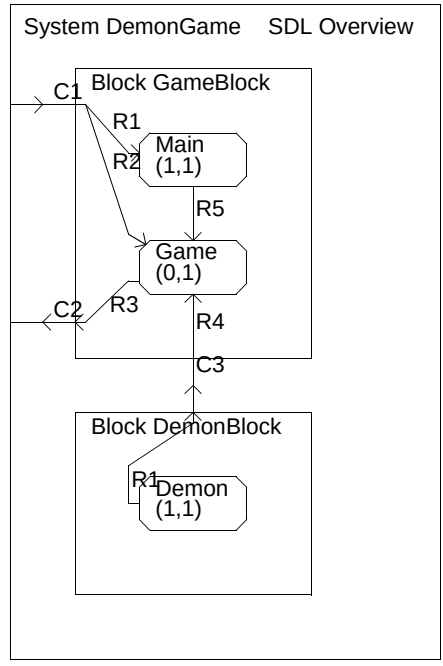


Figure 352: Resulting overview diagram

Interaction and Flow Diagrams

The SDL Editor works with SDL diagrams in graphical form, SDL/GR, and allows graphical editing of *symbols*, *lines* and *text*.

You may open multiple windows in an SDL Editor to be shown at the same time; each window will hold an SDL page. This can be useful if you want to edit different diagrams and pages and refer between them, or if you want to see the interface between two processes on different hierarchical levels. This is the normal mode in which diagrams are edited.

Interaction diagrams are: *system, system type, block, block type, substructure, package* diagrams.

Flow diagrams are: *process, process type, service, service type, procedure, operator, macro* diagrams.

Overview Diagrams

You can also use the SDL Editor to view *SDL overview diagrams*. An SDL overview diagram consists of a number of diagrams that are built up into a diagram subtree in which the symbols are drawn in a nested fashion, giving a more comprehensive overview of a required set of diagrams. You may have multiple SDL overview pages associated to one SDL system, at different SDL levels, and thus presenting different views of the system. You generate overview diagrams in the Organizer (using the *SDL Overview* command), and you can open the overview diagrams for display in the SDL Editor. Various parameters allow you to generate overview diagrams to meet your requirements.

Once an SDL overview diagram is generated, you cannot modify it in the same way that you would normally edit interaction and flow diagrams (unless you regenerate it). However, you can do limited editing on the text of overview diagrams in the SDL Editor.

Essentially, the editor works in read-only mode with regard to graphical operations. General restrictions in editing Overview diagrams are:

- There is no *symbol menu.*
- Only *textual elements* can be modified, not symbols and lines.

SDL Pages

The pages that the SDL Editor displays are always contained within an SDL diagram. An SDL diagram can contain any number of pages, but, must contain at **least one page**, and can be any of the following types.

- *Block interaction, process interaction, service interaction, package* (the *interaction pages*)
- *Graph, service, procedure, macro, operator* (the *flow pages*)
- *Overview*

Relationship between Diagrams and Pages

Pages and diagrams must be associated with each other according to the rules of SDL. There follows a list of what type of pages can be added to different diagrams and what type of pages can be pasted into those diagrams.

Diagram type	Page types that can be added to the diagram	Page types that can be pasted to the diagram
system	block interaction	block interaction package
block	block interaction process interaction	block interaction process interaction package
substructure	block interaction	block interaction package
service	service	service graph procedure macro operator
process	graph service interaction	graph service interaction service procedure macro operator
procedure	procedure	procedure service graph macro operator
system type	block interaction	block interaction package
block type	block interaction process interaction	block interaction process interaction package
service type	service	graph procedure macro operator

Diagram type	Page types that can be added to the diagram	Page types that can be pasted to the diagram
process type	graph service interaction	graph service interaction service procedure macro operator
macro	macro	macro service graph procedure operator
operator	operator	operator service graph procedure macro
package	package	package block interaction
overview	overview (one page only)	-

SDL Page Order

The SDL pages that are contained in an SDL diagram are listed and handled according to the order they were added when they were created.

This order is reflected in some of the menu choices that are related to SDL pages. Also, the structure displayed by the Organizer will adopt the same order. See "Chapters" on page 46 in chapter 2, *The Organizer*.

SDL pages can be renamed and rearranged by the *Edit* menu choice in the *Pages* menu.

Tracing Simulations (Graphical Trace)

Graphical trace, GR trace, is a method to follow the execution of transitions in SDL/GR source diagrams. It is mainly intended to be utilized together with the simulator's *Step-Symbol* command and should normally only be used for a small number of processes to limit the amount of information displayed. The GR trace facility will always show the next SDL symbol within the transition to be executed. After a nextstate or stop operation (e.g. between two transitions), the nextstate or stop symbol is still selected.

The GR tracing is activated from the SDL Simulator. The SDL Editor selects the symbol currently being executed. The Simulator automatically highlights the symbol and displays it. You can use either SDL to see what is being traced, or MSC to see the interaction between processes.

If you use an MSC to trace the simulation, the instances concept of MSC is mapped to the instances concept of SDL processes. The mapping rules which govern how SDL events are transformed into MSC symbols, lines and textual elements are described in "Mapping Between SDL and MSC" on page 1682 in chapter 40, *Using Diagram Editors*.

GR trace will take place in an SDL Editor window containing the appropriate diagram. If the SDL Editor does not hold the current page, but if the diagram can be found in the Organizer, then the page will be opened by the SDL Editor.

If the Organizer cannot find the diagram, or the SDL Editor cannot find an appropriate symbol to select, due to, for example, modifying the diagrams after the generation of the simulator, error messages will be issued by the Organizer or SDL Editor, but the simulation program will continue to execute without tracing graphically.

Setting Breakpoints in Simulations

During simulations, the SDL Editor can show all breakpoints that have been set in the Simulator. It is also possible to set and remove breakpoints directly in the SDL Editor by using the special *Breakpoints* menu that will be shown while this function is operating. The functionality is activated by the *Connect-To-Editor* command in the Simulator.

Compliance with ITU Z.100

The SDL Editor complies with the SDL rules as defined in the ITU Z.100 recommendation. Virtually all parts of the recommendation are supported by the SDL Editor. More information on the SDL support compared to Z.100, see "Compatibility with ITU SDL" on page 9 in chapter 1, *Compatibility Notes, in the Release Guide*.

Syntax Rules when Editing

Some of the SDL syntax rules are enforced during editing when you create diagrams, add pages to these diagrams, add symbols and lines to your pages and edit the text inside the symbols and lines.

Next follow the syntax checks that are performed by the SDL Editor:

- The SDL Editor checks that names on diagrams and pages adhere to the SDL definition.
- The type of a page that you add to a diagram must comply with the diagram type.
- The set of the symbols you are allowed to add to a page is in accordance to the page type. Also, the SDL Editor ensures that symbols are interconnected in a way that is syntactically correct.
- The text that is entered into a symbol or a line is instantly checked for syntax errors. When an error is located a red bar will underline the text where the error occurs.

Turning Syntax Checking On and Off

The syntax checking for entered texts, for what symbols you are allowed to use, and how you can interconnect symbols can be disabled and enabled again.

To disable or enable syntax checking:

1. The operation applies on one diagram at the time. Therefore, make sure you are editing the correct diagram.
2. Select the *Diagram Options* menu choice from the *View* menu. In the dialog which is issued, toggle the *Layout Syntax Check* and/or *Textual Syntax Check* button on or off and click *OK*.

Note: No Retroactive Syntax Checking

Turning the switch from off to on will not perform a retroactive syntax check for not allowed interconnection of symbols. Only the objects that you insert while syntax checking is enabled are checked. However, the syntax check on texts will be performed on all the texts when syntax check is set to on.

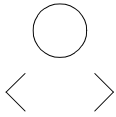
Syntax checking in Kernel Heading and Reference symbol

The syntax check for texts in the kernel heading or any reference symbol is always on independent of the diagram options setting.

Pasting and Syntax Checking

Only symbols that are valid according to what is displayed within the symbol menu may be pasted into a page. Non-valid symbols will be omitted whether or not syntax checking is enabled.

Identical Symbols – Different Syntax



SDL contains some symbols which appear identical, but are distinguished by their syntax. The symbols are:

- The **in connector** and the **out connector**
- The **continuous signal** and the **enabling condition**.

When layout syntax checking is on, the SDL Editor automatically determines from the context what kind of symbol it is. If syntax checking is off, the symbols are always treated as in connector and enabling condition, respectively.

Note: No Layout Check means Syntax errors

When disabling the layout syntax check and using connectors in the diagram mean that it will not be possible to analyze the diagram without errors as all connectors are treated as in connectors. Thus never turn off this switch if you want to analyze the diagram.

SDL Grammar Help

The SDL Editor provides a versatile context-sensitive support function – the *Grammar Help* window. The grammar help windo*w* is available to assist you when entering and editing the syntax of SDL text elements

that are correct according to Z.100 definition. It simplifies writing statements with the correct SDL syntax.

The SDL Grammar Help facility is described in "Using Grammar Help" on page 1906.

Signal Dictionary Support

The SDL Editor also provides a means of access to a *signal dictionary* where the signals that are already defined can be accessed. The Signal Dictionary window provides functions for listing SDL signals that are available when looking up or down in the SDL hierarchy or by looking at the current diagram. Also, you can ask the tool to list signals that are used in the current diagram or defined and visible according to the SDL scope rules.

Furthermore, the signal dictionary has the ability to import messages from a Message Sequence Chart (each MSC message will be mapped to an SDL signal) and to import an external signal dictionary.

For more information on this topic, see "Using the Signal Dictionary" on page 1917.

The SDL Editor User Interface and Basic Operations

The SDL Editor User Interface

The SDL Editor consists of a main window and three auxiliary windows:

- The *Grammar Help window* where SDL textual grammar support helps you entering correct textual expressions.
- The *Signal Dictionary window* which gives you access to signals already defined in the SDL hierarchy.
- The *Entity Dictionary Window* which provides you a name reuse facility as well as information for link endpoints.

The SDL Editor also contains the *symbol menu*, where you select the symbols (objects) that are to be inserted into the page, and the *text window*, where you may edit textual objects.

The general Telelogic Tau user interface is described in chapter 1, *User Interface and Basic Operations*.

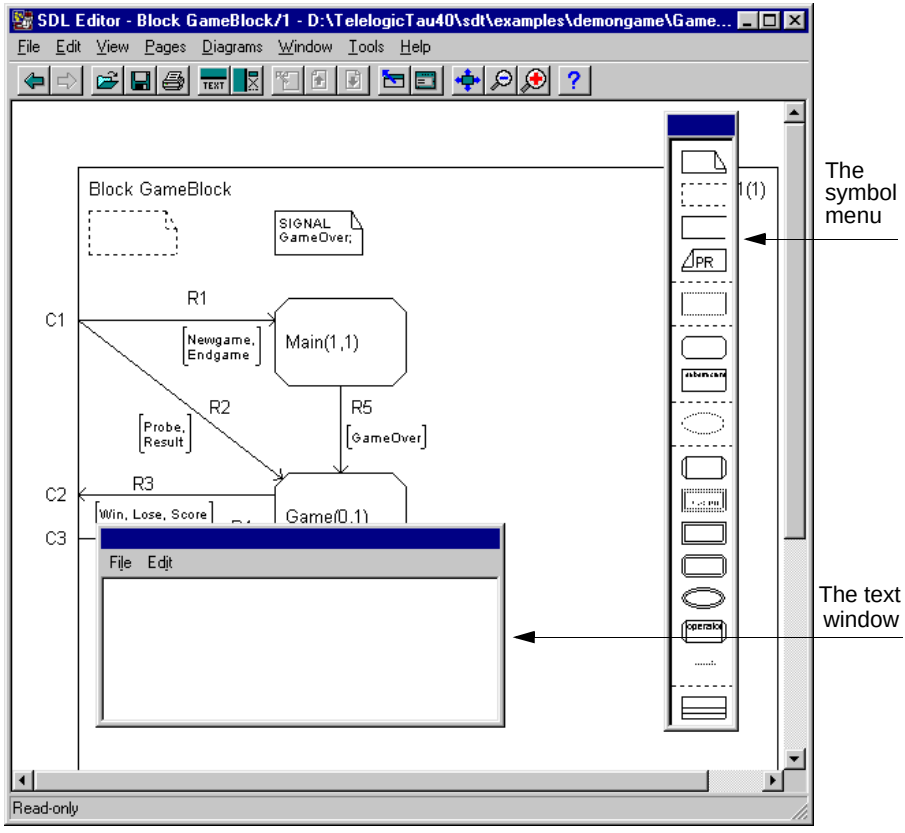


Figure 353: The SDL Editor window **(in Windows)**

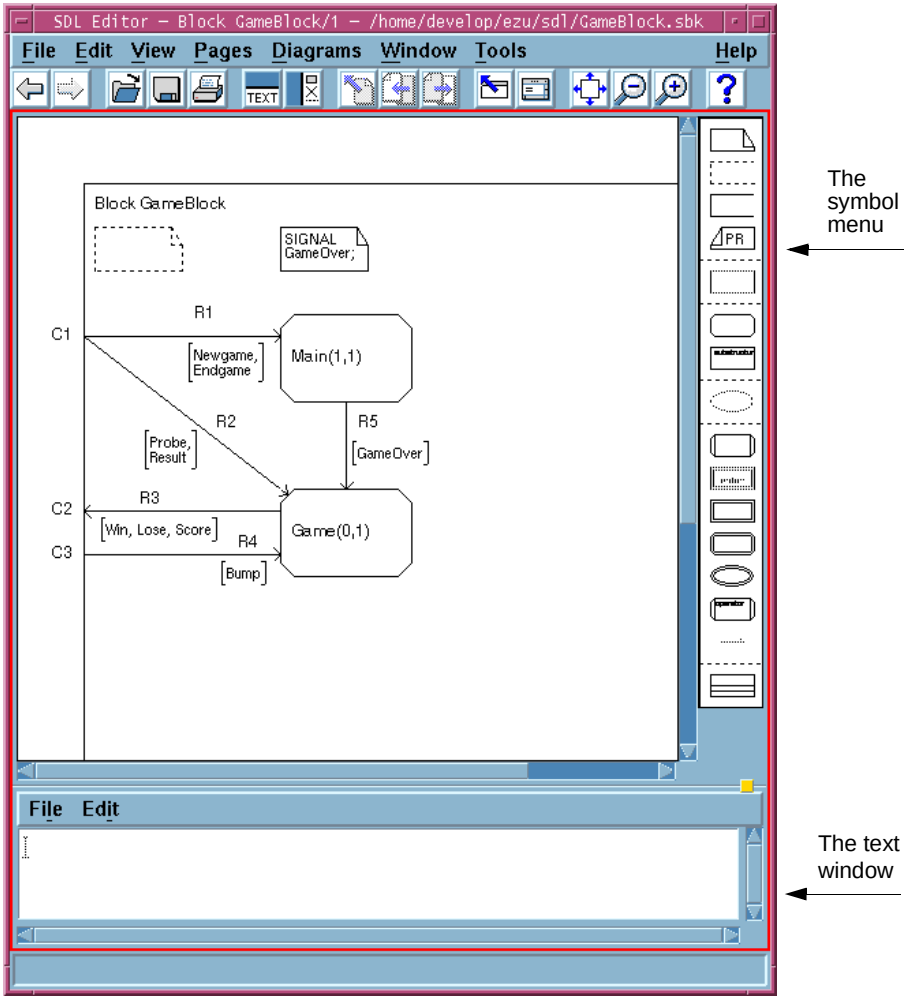


Figure 354: The SDL Editor window **(on UNIX)**

Drawing Area

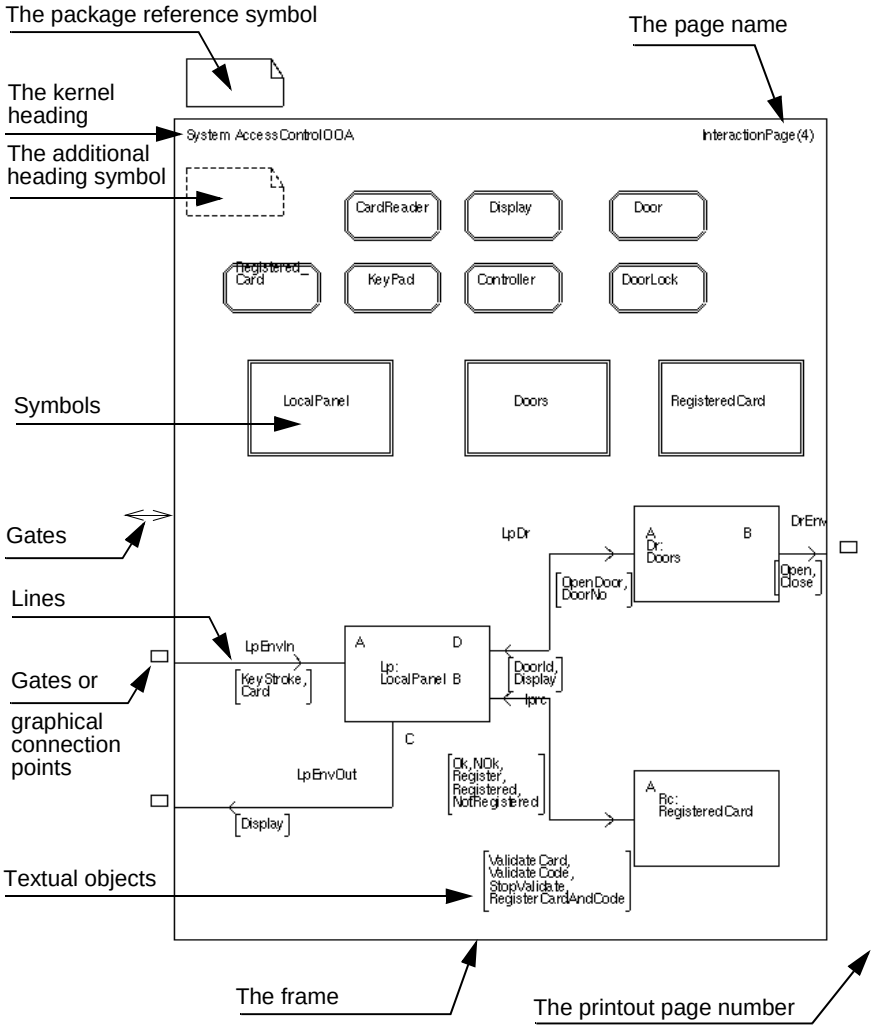


Figure 355: The Drawing Area

The *drawing area* (see Figure 355) is the part of the window that displays the symbols, lines and text that constitute an SDL page (an SDL diagram can contain multiple pages). Most of the graphical notation that is used in the SDL Editor is inherited from the Z.100 definition. In addition, some tool specific notations have been added to facilitate the work with the SDL suite tool set in general, and the SDL Editor in particular.

The Drawing Area Boundaries

The drawing area is delimited by its boundaries, which correspond to the size of the SDL page. Within a diagram, each page has an individual size. No objects are allowed to be placed outside these boundaries. The drawing area uses a light background color, while the area outside the drawing area uses a grey pattern.

The Frame

You can select and resize the frame. All objects, except the package reference symbol, gates and printout page number must reside entirely within the frame.

The Kernel Heading

The kernel heading identifies the diagram type and its name. Alternatively, an SDL qualifier expression can be used. You can edit but not move the kernel heading. The SDL Editor performs a textual syntax check based on the grammar used in this heading.

The SDL suite supports the following syntaxes for the kernel heading:

Case	Syntax
1.	`SYSTEM <name> [: <type expression>]`
2.	`BLOCK <identifier>`
3.	`PROCESS <identifier> [<number of instances>]`
4.	`SERVICE <identifier>`
5.	`<procedure preamble> PROCEDURE <identifier>`
6.	`SUBSTRUCTURE <identifier>`
7.	`MACRODEFINITION <name>`

Case	Syntax
8.	SYSTEM TYPE <identifier>
9.	[<virtuality>] BLOCK TYPE
10.	[<virtuality>] PROCESS TYPE
11.	[<virtuality>] SERVICE TYPE
12.	OPERATOR <operator identifier>
13.	PACKAGE <name>

For an explanation and reference to the notation used in the table above, see the Z.100 recommendation.

The kernel heading is repeated trough all pages contained in an SDL diagram.

The Page Name

The page name identifies the name of the SDL page and, within parentheses, the total number of SDL pages contained in the current SDL diagram.

The contents of the page name are assigned by the tool. You can not select, move, or edit the page name.

Printout Page Number

This object is created by the SDL Editor to inform you about the physical page numbering that will be the result of an SDL page which is larger than the paper format that is defined. You can neither select nor edit it. Page numbering follows a “down first, then right” fashion.

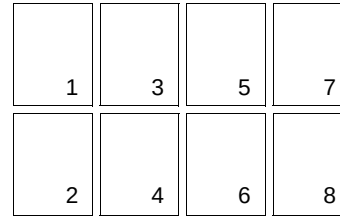


Figure 356: The page numbering in the SDL Editor

Grids

The SDL Editor uses two grids for an easy positioning of symbols, lines and textual elements.

- The symbol grid, that can be set by an SDL Editor preference, has a minimum resolution of 5 * 5 mm. All symbols adhere to the symbol grid.
- The line and text grid has a resolution of 2.5 * 2.5 mm, i.e. half the symbol grid. All lines and textual objects adhere to the line grid.

Default Size of Symbols

The symbols are assigned default sizes when added to the drawing area:

- For interaction pages, the predefined size of a symbol is by default 30 * 20 mm.
- For flow pages, the predefined size of a symbol is by default 20 * 10 mm. The width / size relationship is always 2:1.

You can change the default symbol size in the SDL Editor preferences.

> **Note:**
>
> The size of a class symbol is automatically calculated by the SDL Editor, and may thus differ from the default size.

Color on Symbols

For each symbol type there is a preference for setting the color of the symbol. It is only the graphical part of the symbol and not the associated text(s) that will use the color setting. **On UNIX**, the setting is only valid on screen and all symbols will use the black color when printed on paper. **In Windows**, when using MSW Print the colors will be sent to the printer as well. For more information, see *"The SDL Suite Specific Preferences" on page 260 in chapter 3, The Preference Manager*.

Keyboard Accelerators

In addition to the standard keyboard accelerators, described in "Keyboard Accelerators" on page 35 in chapter 1, *User Interface and Basic Operations*, the SDL Editor features the following:

Accelerator	Reference to corresponding command
Ctrl+D	Opens the next page in the diagram, similar to "<Page Name>" on page 1954
Ctrl+G	"Split Text" on page 1943
Ctrl+I	"Insert Paste" on page 1941
Ctrl+L	"Show High-Level View/Show Detailed View" on page 1949
Ctrl+M	"Mark as Important/Mark as Detail" on page 1944
Ctrl+U	Opens the previous page in the diagram, similar to "<Page Name>" on page 1954
<Delete>	"Clear" on page 1942 (i.e. remove, delete)
Ctrl+1	"Show Organizer" on page 15 in chapter 1, *User Interface and Basic Operations*
Ctrl+2	"Connect to Text Editor" on page 1959

Quick-Buttons

In addition to the generic quick-buttons described in "General Quick-Buttons" on page 24 in chapter 1, *User Interface and Basic Operations*, the SDL Editor tool bar contains the following quick-buttons:

Text window on / off
Toggle the text window between visible and hidden, as described in "Window Options" on page 1950.

Symbol menu on / off
Toggle the symbol menu between visible and hidden, as described in "Window Options" on page 1950.

Reference page
Open the page where this diagram is referenced, similar to "Edit Reference Page" on page 1954.

Log Window
Pop up the Organizer Log window.

Previous page
Open the previous page in the diagram, similar to "<Page Name>" on page 1954.

Next page
Open the next page in the diagram, similar to "<Page Name>" on page 1954.

Toggle scale
Toggle the scale between a scale to show the complete page in the window and a scale of 100%.

Scrolling and Scaling

You can scroll the view vertically and horizontally by using the scrollbars. The view may also be scrolled automatically when you move the cursor beyond the current view, for example when you move an object or add a symbol.

If you move the cursor close to the edge of the current view, the automatic scrolling is slow. If you move it further beyond, the scrolling is quicker.

You can scale the view by specifying a scale or by zooming in and out.

To specify a scale:

1. Select *Set Scale* from the *View* menu.
2. In the dialog that will be opened, you can either:
   - Use the slider for setting the scale and then click the *Scale* button.

   - Click the *Overview* button to adjust the drawing area to the size of the window. (This has the same effect as the *Scale Overview* quick-button.) The smallest scale is 20%.

To zoom in or out:

- Click the quick-button for *zoom in* or *zoom out*.

Moving Selection with Arrow Keys

You can move the selection using arrow keys in the SDL editor if there is one single symbol or line selected. (In this context, we regard a flow symbol together with its attached flow lines to be one symbol.)

Note that you can only move the selection when in symbol editing mode. In text editing mode, the arrow keys will move the cursor position within the text. To get into symbol editing mode, click on the symbol outside of a text area. If you are in text editing mode, you can go to symbol editing mode by pressing the escape key. If you are in symbol editing mode, you can go to text editing mode by pressing the tab key. Pressing the tab key immediately after that will either select another text for the same symbol to edit or go back to symbol editing mode.

You can move to the next page by moving forward from the last symbol on the current page. You can move to the previous page by moving backward from the first symbol on the current page.

Two shortcuts are available for fast-forward moving to colored symbols (symbols not using black as a border color and white as a background color) on pages in the current diagram:

- Shift+arrow key to move to the next/previous colored symbol.
- Shift+alt+arrow key to move to the next/previous colored symbol on another page than the current page.

Symbol Menu

The *symbol menu* contains the SDL symbols that you can place into the drawing area.

On UNIX, the *symbol menu* is a fixed-sized, non-moveable auxiliary window, associated with the drawing area and placed to the right of it. Each editor window has its own symbol menu.

In Windows, the symbol menu is a fixed-sized, moveable window that can be placed anywhere on the screen, not necessary within the limits

of the editor window. A single symbol menu is shared by all instances of the editor currently running.

The symbol menu can be made invisible and visible again with a menu choice, *Window Options*, or a quick-button **In Windows:** When visible, the symbol menu will always be placed on top of the editor window.

Basically, when you select a symbol in the symbol menu and click it into the drawing area it is added to the diagram. How to work with symbols is described in "Working with Symbols" on page 1835.

The contents of the symbol menu depends upon the type of diagram that is displayed in the SDL Editor window and the settings of the SDL Editor preferences depends upon whether the symbols should be used or not.

- For *process, process type, procedure, macro definition* and *operator diagrams* (the *flow* diagrams), the default symbol menu looks like the left picture in Figure 357.
- For interaction diagrams (*system, block, substructure, service, package, system type, block type* and *service type*), the default symbol menu looks like the right picture in Figure 357.

For a reference to these symbols, see "Symbols on Interaction Pages" on page 1819 and "Symbols on Flow Pages" on page 1821.

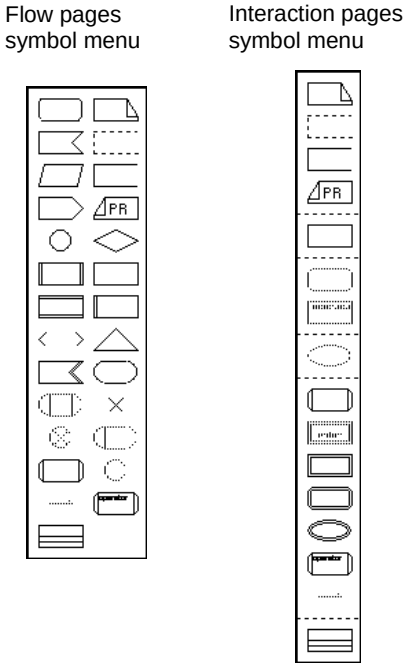


Figure 357: The Symbol menu

Syntax Checking on SDL Symbols

If layout syntax checking is enabled, some of the symbols displayed in the *symbol menu* may be dimmed. This indicates that those symbols are not valid within the current context, and that you cannot select them.

- When layout syntax checking is disabled, you can select all symbols.

- When layout syntax checking is enabled, the following rules apply:

Type of page	Unusable symbols (dimmed)
System / System type	process reference service reference block substructure reference system type reference gate
Block diagram / Block interaction page	process reference service reference block substructure reference system type reference gate
Block diagram / Process interaction page	block reference service reference system type reference gate
Service	procedure start procedure return macro inlet macro outlet gate
Process diagram / Graph page	procedure start procedure return macro inlet macro outlet gate
Process diagram / Service interaction page	block reference process reference block substructure reference system type reference block type reference process type reference gate

Type of page	Unusable symbols (dimmed)
Procedure	start stop macro inlet macro outlet gate
Block type diagram / Block interaction page	process reference service reference block substructure reference system type reference
Block type diagram / Process interaction page	block reference service reference system type reference
Macro	gate
Service type	procedure start procedure return macro inlet macro outlet
Process type diagram / Graph page	procedure start procedure return macro inlet macro outlet
Process type diagram / Service interaction page	block reference process reference block substructure reference system type reference block type reference process type reference

Type of page	Unusable symbols (dimmed)
Operator	state input save output procedure call enabling condition / continuous signal create request priority input procedure reference start stop macro inlet macro outlet gate
Package	block reference process reference block substructure reference service reference gate

Working with Diagrams

This section describes the methods you use when performing the following tasks on SDL diagrams:

- *Creating a Diagram*
- *Including a Diagram*
- *Opening a Diagram*
- *Saving a Diagram*
- *Saving a Copy of an SDL Diagram*
- *Closing a Diagram*
- *Printing a Diagram*
- *Displaying an Opened Diagram*
- *Reorganizing a Diagram*.
- *Transforming the Type of a Diagram*

Creating a Diagram

You can create diagrams from within other diagrams. When created, they form part of the current active system in the SDL Editor, and their presence is instantly reflected in the Organizer. This is achieved by inserting SDL diagram *reference symbols.*

You can also create diagrams as separate unrelated new entities. They will however not be incorporated in the Organizer.

Creating a Related Diagram from Another Diagram

1. Open the SDL diagram from where to start.
2. Insert the SDL reference symbol.
3. Assign the symbol a name.
4. Deselect the reference symbol. The Organizer's diagram structure is updated accordingly.
5. Double-click the newly added reference symbol. You should specify what action to perform, either:
   - Create a new diagram in the SDL Editor from scratch.
   - Copy an existing file.

   Select an option and click *OK*.

The SDL Editor will now create a new diagram, possibly a new window, and display the newly created diagram.

6. Save both the parent and the child diagram later on in your editing session, to make changes permanent.

Example in an Organizer Structure

As an example, in the Organizer structure in Figure 358, a system is shown with only one block diagram.

Figure 358: A basic system shown in the Organizer

Suppose the system diagram is being edited. We now add a new block reference symbol, name it Block_B and deselect it. The Organizer diagram structure becomes:

Figure 359: Having added a reference symbol

The SDL suite shows, with the `[unconnected]` identifier, that the newly added reference symbol is not yet connected to any physical file.

Now, double-click the Block_B symbol. Specify whether you want to create a new file or to connect the newly created block to an existing file.

Make sure the *Show in Editor* radio button is on and click *OK*. In some cases a new dialog appears, where you are to specify the name and type of the first SDL page to add to the diagram (an SDL diagram must contain at least one SDL page).

Assign a page name (see "Adding a Page" on page 1888) and click *OK*.

A new block diagram with the name Block_B is created. The SDL Editor responds by issuing a window showing this diagram. Also, the Organizer's diagram structure is updated, showing the new diagram icon, inserted at the corresponding place in the SDL hierarchy.

Figure 360 on page 1805 shows the result:

- A newly created diagram, empty except for the diagram heading
- The resulting diagram structure
  - The parent diagram needs to be saved (marked with a gray pattern)
  - The child diagram needs to be saved.

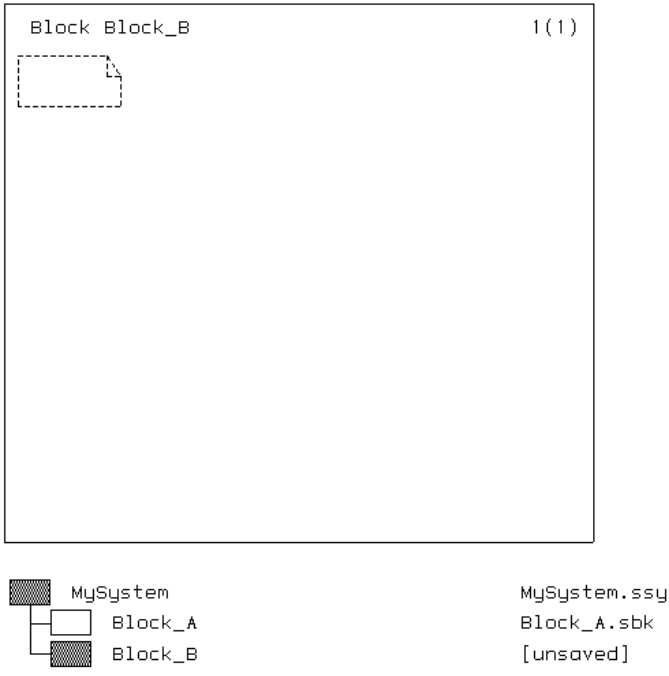


Figure 360: The newly created diagram and the resulting diagram structure

Creating an Unrelated New Diagram

1. Select *New* from the *File* menu of the SDL Editor. A dialog will be opened.
2. Enter the name of the new diagram in the *Diagram Name* field. The name should be in keeping with SDL naming conventions.

3. Choose the type of diagram from the *diagram type* menu.
4. Click *New*. In some cases a new dialog box is shown where you can add a page, as each diagram must have at least one page. See the section "Adding a Page" on page 1888 for this.

   A window on the newly created diagram is opened.

Including a Diagram

Including a Diagram into the SDL Hierarchy

When you have created a diagram, it needs to be brought into the SDL hierarchy to be available for future selection in the Organizer. Otherwise, you will only be able to open it from the SDL Editor.

1. Create the diagram using the *New* command from the SDL Editor. *Save* the diagram on file.
2. Open the diagram that will have a reference to the newly created diagram.
3. In the diagram you just opened, add a diagram reference symbol with a type and name that matches the newly created diagram. A new icon appears in the Organizer structure.
4. With the Organizer *Connect* command, connect the new icon with the file on which the diagram is stored. See "Connect" on page 91 in chapter 2, *The Organizer*.

Once a reference symbol is placed in the newly created page of the new diagram in the SDL Editor, the Organizer is updated to reflect this. This indicates that the SDL hierarchy is updated.

Including a Diagram into the Organizer

If a diagram is not managed by the Organizer (i.e. not contained in the Organizer structure) you must first include it into the Organizer, before you can start the SDL Editor.

1. Select the place in the Organizer where you want to add the SDL diagram. It will be added as a new root diagram after a selected document or chapter, or at the top level of a selected module.
2. Select the Organizer's *Add Existing* command from the *Edit* menu. A file selection dialog will be opened.
3. Select a SDL diagram file.

   The SDL diagram is added to the Organizer structure, and opened in the SDL Editor.

Opening a Diagram

There are several ways you can open a diagram for editing, depending on where you are and what you need to access.

Lock Files and Read-Only Mode

When you open a diagram or document file, i.e. *a.ssy*, a lock file *a.ssy.lck* is created. If another user tries to open the same diagram or document file before you close it, a dialog will appear informing the other user that the file is in use.

There are two choices in the dialog:

- Open the file in read-only mode.
- Reset the lock and open the file in read-write mode. (This alternative should only be used if the existing lock file is obsolete.)

The read-write mode is the normal editing mode. In read-only mode, you are not allowed to make any changes to the diagram.

You will also enter read-only mode if you open a diagram file that you do not have write permission for. The read-only mode is indicated by the words *read-only* in the window title.

If you want to edit a diagram that is in read-only mode, here are two alternative actions:

- Change the file access rights in the file system to give you write permission for the diagram file. Then change from read-only mode to read-write mode with the *Revert Diagram* operation.
- Use *File >Save As* to save the diagram in a new file with write permission.

Opening a Diagram File from the SDL Editor

1. Select *Open* from the *File* menu or click the quick-button for *Open*. A file selection dialog is opened.
2. Select a file in the files list and click *OK*. If the file is already opened by the SDL Editor, a message is issued.

> **Note:**
> The *status bar* at the bottom the SDL Editor provides information about the diagram type and name stored on the file which is selected in the file list.

Opening a Diagram from the Organizer

You can open a diagram in different ways:

- Select an SDL diagram icon or the text line next to it in the Organizer. Select *Edit* from the Organizer's *Edit* menu.
- Double-click on an SDL diagram icon in the Organizer.

Opening a Diagram – via Organizer – to Show Analyzer Reports

When analysis has been performed, you can show the source of Analyzer error reports in an SDL Editor.

- Select *Show Error* from the *View* menu in the *Organizer Log Window*. An SDL Editor window is opened with the source of error (i.e. an SDL symbol) selected.

Opening a Diagram via a Reference Symbol

You can open a diagram from the SDL Editor by double-clicking on a reference symbol, or by using the *Navigate* command.

1. Locate the SDL reference symbol referring to the diagram you want to open.
2. Double-click the symbol or select it and choose the *Navigate* command from the *Tools* menu. The diagram is opened.

Opening a Diagram from the Simulator

If you have performed a GR trace in the SDL Simulator, invoking the Show-Next-Symbol or Show-Previous-Symbol simulator commands will display either the next symbol to be executed or the previously executed symbol in a diagram in an SDL Editor window.

Opening a New (Nonexisting) SDL Diagram

To open a new SDL diagram, you either use an unconnected SDL diagram icon or add a new SDL diagram icon.

Opening an Unconnected SDL Diagram Icon

To open an existing but unconnected SDL diagram icon:

1. Select an unconnected SDL diagram icon in the Organizer.
2. Double-click the SDL diagram icon. A dialog is opened.
3. Make sure the *Show in editor* option is set. Click *OK*. A new, empty diagram is opened.
   - To copy the contents of an existing SDL diagram file before opening the new diagram, select the option *Copy existing file* and enter a filename.

Opening a New SDL Diagram Icon

To add a new SDL diagram icon to the Organizer and opening that icon:

1. Select the place in the Organizer where you want to add the SDL diagram. The diagram will be added as a new root diagram after a selected document or chapter, or at the top level of a selected module.
2. From the Organizer's *Tools* menu, select the *Editors > SDL Editor* command. A new SDL diagram icon is added to the Organizer structure, and the new diagram is opened in the SDL Editor.
   – Alternatively, select the Organizer's *Add New* command from the *Edit* menu. A dialog is opened. Select a SDL diagram type, enter a name in the *New document name* field and click *OK*. The *Add Page* dialog is displayed, see "Adding a Page" on page 1888.

Opening an Existing SDL Page

To open a specific page, the SDL pages must be visible in the Organizer diagram structure.

1. To display page icons in the Organizer, select the Organizer's *View Options* command from the *View* menu. In the dialog, make sure *Page symbols* is highlighted and click *Apply*.
2. Select the icon representing the SDL page in the Organizer.
3. Double-click the SDL page icon. The SDL diagram page is displayed.

Saving a Diagram

There are several ways to save a diagram. The effect is saving any changes made to either one specific diagram, or all diagrams modified during the current session.

Saving a Diagram in the SDL Editor

- Select *Save* from the *File* menu or click the quick-button for *Save.*

  If the diagram is newly created, a file selection dialog is issued where you can specify the file to save the diagram on.

Saving All Diagrams in the SDL Editor

You can make a global save of all the diagrams that are open in the current session and have been modified.

- Select *Save All* from the *File* menu. All diagrams which are modified are saved.

Saving a Copy of an SDL Diagram

You can save the diagram being edited into another file. You may either continue working with the original file or with the newly created copy.

Continue Working with the New Copy

1. Select *Save As* from the *File* menu. A file selection dialog is issued.
   - The file name which is suggested is the same file as the original was stored on, appended with an integer suffix (1, ..., N) in order to build up a unique file name.
2. Click *OK*. The SDL diagram is copied to the specified file, the file containing the original is closed, and the window where you made the save remains open for editing the newly created file.

Continue Working with the Original

1. Select *Save a Copy As* from the *File* menu. A file selection dialog is issued.
   - The file name which is suggested is the same file as the original was stored on, appended with an integer suffix (1, ...,N) that guarantees a unique file name.
2. Click *OK*. The SDL diagram is saved on the specified file, and the window where you made the save remains open for editing.

Closing a Diagram

Closing a diagram also means closing all instances of all windows displaying any page contained in that diagram.

- Select *Close Diagram* from the *File* menu.

  If you have modified the diagram, a *Close* dialog is shown where you can save the modifications before closing the diagram.

Printing a Diagram

You can print an entire SDL diagram or a selection of pages or objects contained in the diagram. You print multiple SDL diagrams from the Organizer.

See *"The Print Dialogs in the SDL Suite and in the Organizer" on page 308 in chapter 5, Printing Documents and Diagrams*.

Displaying an Opened Diagram

You can open the diagram that is currently read by the SDL Editor. (The diagram that you would like to see may reside in a window which is not on top of the screen.)

The *Diagrams* menu shows all SDL diagrams and pages (up to a maximum of the last nine to have been used). If more than nine pages are open, a tenth menu choice, *List All*, provides access to a list dialog where all diagrams and pages are listed.

Reorganizing a Diagram

The SDL Editor can reorganize the layout of an SDL diagram, for instance if you run out of space on any of the pages that build up the diagram.

To reorganize an SDL diagram:

1. Display the diagram you want to reorganize.
2. Select *Tidy Up* from the *Tools* menu. A dialog is opened.
3. Click *Tidy Up*. The SDL Editor reorganizes the diagram (this may take some time, depending on how complex your diagram is).
4. Check the diagram before continuing working with it (you can undo the operation from the *Edit* menu).

Transforming the Type of a Diagram

A useful operation when e.g. transforming to a more object-oriented version of the SDL system is to change the SDL88 diagrams into type diagrams and make instantiations of these types. The following shows some examples how to perform such an operation and similar ones

where you want to change the type of a diagram or want to work with an SDL structure that is almost a copy of an already existing hierarchy.

Example 1

Assume that the system S consists of block B, which contains process P. The exercise to solve is to transform the block into a block type BT and put this into a package P and instantiate the block B from BT.

There is no direct command to accomplish this task in the SDL suite; instead there are many solutions, more or less complicated and time consuming. The proposed one is easy to use and will take advantage of the possibility to cut and paste complete sub-hierarchies, thereby eliminating loosing file connections and the need for making manual connections to existing files.

Solution: Open the system S, add a new block type symbol and name it BT. Open the block B and change the kernel heading from “block B” to “block type BT”. Note that the sub-hierarchy in the Organizer, previously below B, has now been moved to BT. In system S, change the text in the block symbol to “B:BT” to make it an instance of BT and cut the block type symbol to the clipboard. Create a package diagram named P and paste the block type into this diagram.

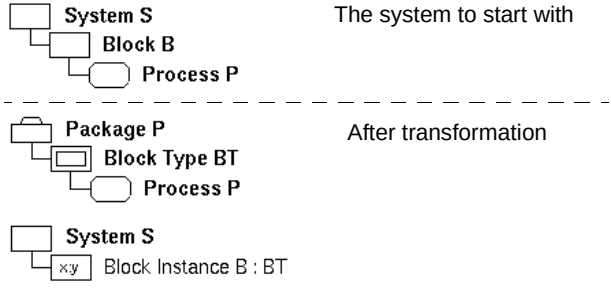


Figure 361: The Organizer view

To make the new system complete you need to add the text “use P;” in the package reference symbol in system S, define the gates inside the block type BT and use these gates to connect the existing channels for the block B. If needed, you can change the filename of the block type with the *Save As* command in the SDL Editor to use a filename that re-

sembles the block type name and also use the file name extension that conforms to the SDL suite conventions.

Example 2

This is a variant of example 1. Assume we want to keep the original diagram but want to have a new block type BT in package P that is a copy of the existing B.

Solution: In package P place a new block type symbol and name it BT. Double click on it and in the appearing Edit dialog tick the *Copy existing file* option and use the file connected to the existing block diagram B. Click *Continue* in the warning about recommended file name extension. In the opened diagram, change the kernel heading to "block type BT".

For all diagram references in BT (in the example there is only one), use the *Connect* command in the Organizer to connect them to the existing files. In the Connect dialog be sure to use the *Expand Substructure* option to get automatic connections for possible sub-hierarchies.

As an alternative to connecting the diagram references you can copy the reference symbols from block B and paste them into block type BT, then you will get automatically all the connections for sub-hierarchies below the pasted references.

Note that in either case you will use the same file connection for process P in our example for both the original process and also for the process P referenced in the block type, meaning that if you change the contents of this file you will change both the behavior of the block B and the block type BT, which might lead to unexpected results. If you want unique file connections you should take a copy of the existing file, for example copy `P.spr` to `P1.spr`, and reconnect one of the processes to this new file instead.

Example 3

The SDL system consists of system S, block B, process P and procedure subPr inside P. You want to transform most of the process including its procedure subPr into a new procedure Pr defined in the system S.

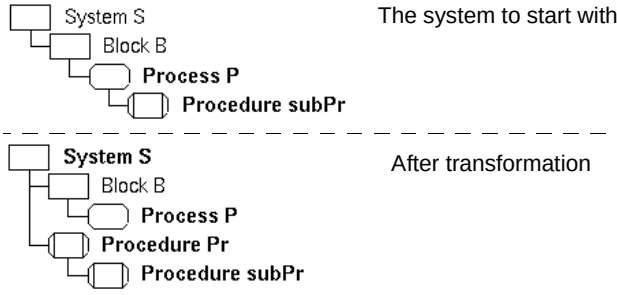


Figure 362: The Organizer view

Solution: This solution is similar to the solution to example 2. Open the system diagram S and place a procedure reference symbol named Pr. Double click it and open a copy of the existing file connected to process P. Edit the kernel heading to contain the text “procedure Pr”. Save the procedure as a new file. Connect all referenced procedures inside Pr or copy the procedure references from the process P to get all the sub-hierarchy file connections. Finally edit process P and procedure Pr to make them useful in their new context.

The symbol is dashed and the change is immediately reflected in the Diagram Structure (this is only true if the *Dashed Diagram* option is on in the Organizer’s *View Options*).

- You may at any time *undash* a dashed symbol. Select the symbol and select *Undash* from the *Edit* menu.
- If the symbol (prior to the dash operation) was connected in the Organizer the following dialog will appear:

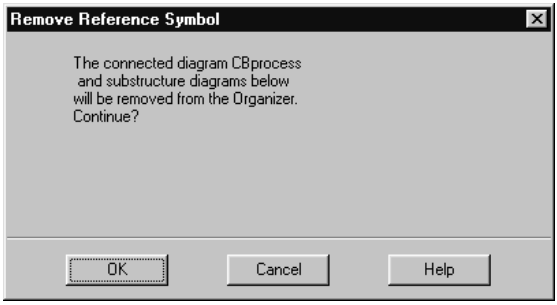


Figure 363: Removing substructure by dashing

- If you click *OK*, the substructure will be removed. However, you may undo this.

About Symbols and Lines

Following is a description of the components in the drawing area. See also appendix 45, *Symbols and Lines – Quick Reference, on page 1987*.

The Additional Heading Symbol

Symbol Appearance	Symbol Name	References to Z.100
	Additional heading	Z100: 2.2.5 Additional heading Z100: 6.1.1.3 Process type heading Z100: 2.4.4 Formal parameters, Signalset Z100: 6.3 Specialization Z100: 6.3.1 Inherits Z100: 6.3.2 Virtual, Redefined, Finalized Z100: 6.3.2 Virtuality constraint (atleast) Z100: 6.2 Actual context parameters Z100: 6.2 Formal context parameters Z100: 6.2.1 - 6.2.9 Formal context para

The additional heading symbol is not defined further according to Z.100. In the SDL Editor, it looks like a dashed text symbol or it is shown without any border; this is controlled by the Editor preference ScreenZ100Symbols. The symbol is editable and resizeable, but cannot be moved. Its intended use is to define:

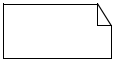
- Inheritance and specialization, using the following keywords:
  - `INHERITS`
  - `ADDING`
  - `ATLEAST`
- Formal parameters for a process / procedure, using the keyword `FPAR`

- Directives to the SDL to C Compiler. A directive uses the SDL comment notation and a hash character, such as:
  `/#INCLUDE file.pr */`
- `SIGNALSET` and instance information.

The additional heading symbol is either repeated through all pages contained in an SDL diagram or it is only shown on the first page. This is controlled by the editor preference AdditionalHeadingOnlyOnFirstPage and can also be specified with the SDL Editor command *Diagram Options*.

Note that the combined text in the Kernel Heading and the Additional Heading symbol must use the syntax as specified for a heading in SDL/PR to be accepted in the SDL Analyzer.

The Package Reference Symbol

Symbol Appearance	Symbol Name	References to Z.100
	Package reference	`Z100: 2.4.1.2 Reference in Package` `Z100: 2.4.2 Reference in System`

The package reference symbol is available on system and package diagrams only. It is located outside the frame, on top and to its left. It contains references to package(s) containing definitions that are to be included in the SDL system.

You can select and resize, but not move the package reference symbol.

Other SDL Symbols

These are the symbols that describe the structure or behavior of the SDL diagram. They must be placed inside the frame.

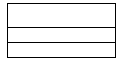
You can draw symbols in color, see "Color on Symbols" on page 1794.

Syntax Checking on Symbols

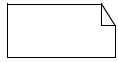
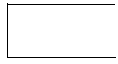
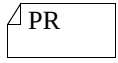
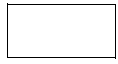
The SDL Editor checks that the symbols you add to a diagram are in accordance with the syntactic rules imposed by SDL. Symbols that are not allowed in a diagram / page of a specific type are dimmed in the symbol

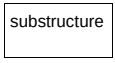
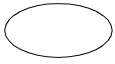
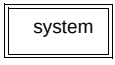
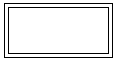
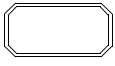
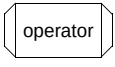
menu. Also the text entered is checked for syntax errors. See "Layout syntax check" on page 1951.

Symbols on both Interaction Pages and Flow Pages

Symbol Appearance	Symbol Name	References to Z.100
	Class	Not included (Z100 11/99)

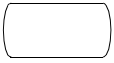
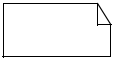
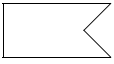
Symbols on Interaction Pages

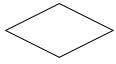
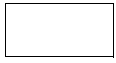
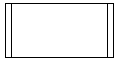
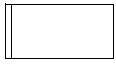
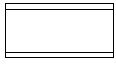
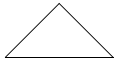
Symbol Appearance	Symbol Name	References to Z.100
	Text	Z100: 2.5.4 Signal Z100: 2.5.5 Signal list Z100: 5.2.1 Newtype Z100: 5.3.1.9 Syntype Z100: 5.3.1.13 Synonym Z100: 5.3.1.12.1 Generator Z100: 4.13 Remote variable Z100: 4.14 Remote procedure
	Comment	Z100: 2.2.6
	Text extension	Z100: 2.2.7 (depends on the symbol connected to)
	Text Reference	None
	Block reference	Z100: 2.4.2 Block definition Z100: 6.1.3.2 Block def based on block type Z100: 6.1.2 Type expression Z100: 6.2 Actual context parameters

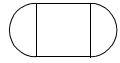
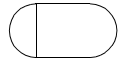
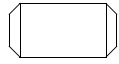
Symbol Appearance	Symbol Name	References to Z.100
	Process reference	Z100: 2.4.3 Process definition Z100: 2.4.4 Number of instances Z100: 6.1.3.3 Process def based on block type Z100: 6.1.2 Type expression Z100: 6.2 Actual context parameters
	Block substructure reference	Z100: 3.2.2
	Service reference	Z100: 2.4.4
	System type	Z100: 6.1.1.1
	Block type	Z100: 6.1.1.2
	Process type	Z100: 6.1.1.3
	Service type	Z100: 6.1.1.4
	Operator reference (an SDL/GR extension defined in the SDL suite)	Z100: 5.3.2 Referenced operator in SDL/PR

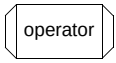
Symbol Appearance	Symbol Name	References to Z.100
	Gate	Z100: 6.1.4 Gate Z100: 2.5.5 Signal list

Symbols on Flow Pages

Symbol Appearance	Symbol Name	References to Z.100
	State or nextstate	Z100: 2.6.3 State Z100: 4.4 Asterisk state Z100: 4.5 Multiple appearence of state Z100: 2.6.8.2.1 Nextstate Z100: 4.9 Dash nextstate
	Text	Z100: 2.5.4 Signal Z100: 2.5.5 Signal list Z100: 5.2.1 Newtype Z100: 5.3.1.9 Syntype Z100: 5.3.1.13 Synonym Z100: 5.3.1.12.1 Generator Z100: 2.6.1.1 Variable Z100: 2.6.1.2 View Z100: 2.8 Timer Z100: 4.13 Remote variable Z100: 4.13 Imported variable Z100: 4.14 Remote procedure Z100: 4.14 Imported procedure
	Input	Z100: 2.6.4 Input Z100: 4.6 Asterisk input Z100: 4.14 Remote procedure input Z100: 6.3.3 Virtual transition z100: 5.4.3 Variable z100: 5.4.3.1 Indexed variable z100: 5.4.3.2 Field Variable
	Comment	Z100: 2.2.6

Symbol Appearance	Symbol Name	References to Z.100
	Save	Z100: 2.6.5 Save Z100: 4.7 Asterisk save Z100: 6.3.3 Virtual save
	Text extension	Z100: 2.2.7 (depends on the symbol connected to)
	Output	Z100: 2.7.4
	Decision	Z100: 2.7.5 Decision Z100: 5.3.1.9.1 Range condition Z100: 2.2.3 Informal text
	In-connector or out-connector	Z100: 2.6.7 In-connector Z100: 2.6.8.2.2 Out-connector
	Task, set, reset or export	Z100: 2.7.1 Task Z100: 5.4.3 Assignment Z100: 2.8 Set, Reset Z100: 4.13 Export
	Procedure call	Z100: 2.7.3 Call Z100: 2.7.2 Actual parameters
	Macro call	Z100: 4.2.3
	Create request	Z100: 2.7.2
	Transition option	Z100: 4.3.4 Transition option Z100: 5.3.1.9.1 Range condition

Symbol Appearance	Symbol Name	References to Z.100
	Continuous signal or enabling condition	Z100: 4.11 Continuous signal Z100: 4.12 Enabling condition
	Start	Z100: 2.6.2 Start Z100: 6.3.3 Virtual transition
	Priority input	Z100: 4.10 Priority input Z100: 2.6.4 Stimulus Z100: 6.3.3 Virtual transition z100: 5.4.3 Variable z100: 5.4.3.1 Indexed variable z100: 5.4.3.2 Field variable
	Stop	Z100: 2.5.8.2.3
	Procedure start	Z100: 2.4.6
	Inlet	Z100: 4.2.2
	Return	Z100: 2.6.8.2.8
	Outlet	Z100: 4.2.2
	Procedure reference	Z100: 2.4.6 Procedure definition

Symbol Appearance	Symbol Name	References to Z.100
	Operator reference (an SDL/GR extension defined in the SDL suite)	Z100: 5.3.2 Referenced operator in SDL/PR
	Gate	Z100: 6.1.4 Gate Z100: 2.5.5 Signal list

Note:

The operator reference symbol is not part of the current Z.100 recommendation. It has been added to the SDL Editor as a convenience for the user. It makes operators visible in the SDL structure which is handled by the Organizer and facilitates thus navigating.

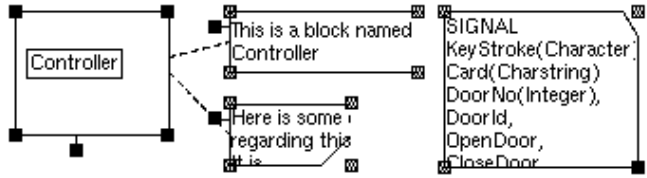
The operator reference symbol does not, however, refer to the operator diagram implicitly. References to operator diagrams must be explicitly entered in SDL/PR, as stated in Z.100.

Graphical Properties of Symbols

You can select and move all symbols that are available in the symbol menu, and you can assign the symbols arbitrary locations.

You can resize certain symbols; these are indicated by filled selection squares. Other symbols can only be partially resized or cannot be resized at all[1]; this is shown by grayed selection squares.

1. The size of non-resizeable symbols is computed by the SDL Editor to fit the text contained in the symbol.



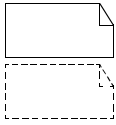
This symbol can be resized...

...while that one can be either full sized or clipped...

...and the third one can be resized by dragging the lower right corner.

Figure 364: Resizeable and non-resizeable symbol

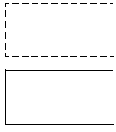
No symbol is allowed to overlap any other symbol except the text symbol and the additional heading symbol.



Text / Additional Heading / Package Reference Symbols

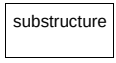
The text, additional heading and package reference symbols look like a piece of paper. When all of the text within a text symbol is in view, the upper right corner is “folded” down. When any portion of the text within a text symbol cannot be seen (because the text symbol is too small), the upper right corner is clipped or diagonally cut off. See Figure 364.

Clipped text symbols are printed in their whole on a separate sheet of paper.



Comment / Text Extension Symbols

The comment and text extension symbols can be shown either in full size enclosing the complete text or clipped (collapsed) where the symbol is in the minimum size and only the first part of the text is visible. The clipped symbol is visualized as if the lower right corner has been cut off. See Figure 364.



Block Substructure Symbol

The appearance of this symbol differ from the drawing rules in Z.100. The non editable text “substructure” is drawn inside the symbol. Without this extra text it is impossible to distinguish the block symbol and the block substructure symbol by their visual appearance.

The appearance can be set to be according to Z.100 by the Editor preferences ScreenZ100Symbols and PrintZ100Symbols. Setting these to on means that the extra text will not be drawn or printed.

Symbol Text Attributes

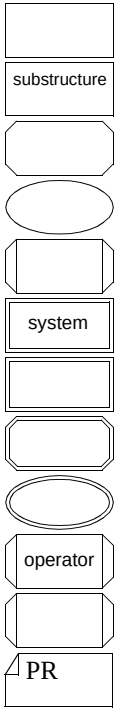
Most SDL symbols have one or multiple text attributes. A text attribute should be filled with an SDL/PR expression (textual expression) that is syntactically correct according to Z.100. Depending on the correctness of the text, the SDL suite tool set has the ability to perform the following operations:

- Syntactic Analysis
- Semantic Analysis
- Pretty-printing
- Generation of reports
- Generation of code.

Syntax Checking on Text Attributes

You are not forced by the SDL Editor to fill text attributes with text. However, a context sensitive syntax check is performed for each text attribute. The first located syntax error is indicated by a red bar underlining the text where the error occurs.

A global syntax check for the complete diagram is performed by the SDL Analyzer.



Reference Symbols

Diagram reference symbols allow to build an entire SDL system by referring to diagrams that are structurally related. Reference symbols are the following:

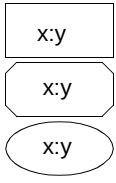
- Block reference
- Process reference
- Block substructure reference
- Service reference
- System type
- Block type
- Process type
- Service type
- Operator reference
- Procedure reference
- Text Reference

These symbols are handled by the SDL Editor in order to ensure consistency between the SDL diagrams and the SDL structure that is handled by the Organizer. Because of that, the SDL Editor imposes the restriction that a reference symbol must be unique within an SDL diagram (this restriction is not in conflict with Z.100).

You are permitted to have diagram reference symbols that are not assigned any name.

The SDL Editor performs a number of checks when you edit an SDL reference symbol. Furthermore, when you double-click a reference symbol, the SDL diagram that the symbol refers to will be opened.

See also "Working with Diagram Reference Symbols" on page 1835.



Instantiation Symbols

You may use any of the following reference symbols for the instantiation of a type:

- Block reference
- Process reference
- Service reference.

An instantiation symbol differs from a “normal” reference symbol in the sense that the syntax of the symbol’s text differs. The syntax is:

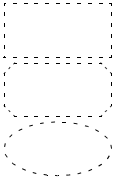
```
a:a_type
```

where *a_type* is the name of the SDL type diagram (block type, process type or service type diagram) that is instantiated into the diagram with the name *a*.

Instantiation symbols are visualized in the Organizer structure. The Organizer structure is updated to reflect the nature of the changes applied to the SDL diagram, in a similar fashion as for diagram reference symbols (see “Reference Symbols” on page 1827).

The names of instantiation symbols must be unique within an SDL diagram.

See also “Working with Diagram Instantiation Symbols” on page 1835.



Dashed Reference Symbols

Any of the following reference symbols may be dashed:

- Block reference
- Process reference
- Service reference.

Dashed symbols are visualized in the Organizer structure. The Organizer structure is updated to reflect the nature of the changes applied to the SDL diagram, in a similar fashion as for diagram reference symbols (see “Reference Symbols” on page 1827).

The names of dashed reference symbols must be unique within an SDL diagram.

See also “Working with Dashed Symbols” on page 1836.

Lines

Lines are the graphical objects that interconnect symbols. One line only is available in the symbol box, namely the gate, which is handled as a symbol. You can insert it by selecting it and placing it into the drawing area.

You insert the other lines by selecting a symbol and dragging the *handle* that appears on the source symbol and connecting it to the target symbol. (Some symbols have multiple handles).

Lines are always connected to symbols, they are not allowed to exist on their own.

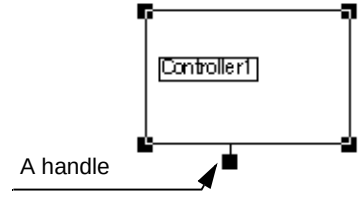


Figure 365: A handle

You can select, move and reshape lines. Some layout work is performed automatically by the SDL Editor.

A line is allowed to overlap any other object.

Lines in Interaction Diagrams

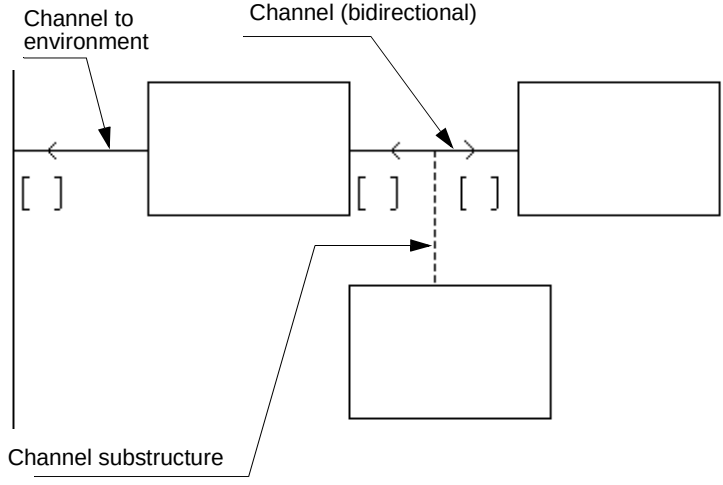


Figure 366: Lines (1)

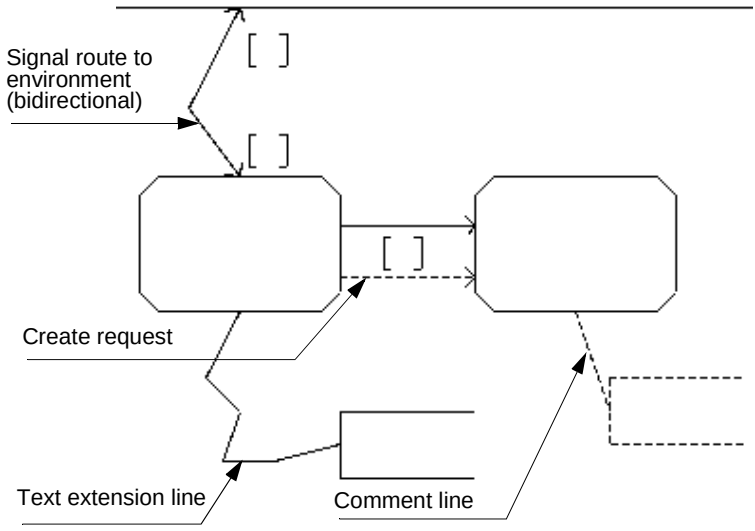


Figure 367: Lines (2)

The following lines are defined in interaction diagrams (see Figure 366 and Figure 367):

- channels
- channel substructures
- signal routes
- create requests
- text extension lines
- comment lines

You can reshape each of these lines and you can also move the connection points.

Lines in Flow Diagrams

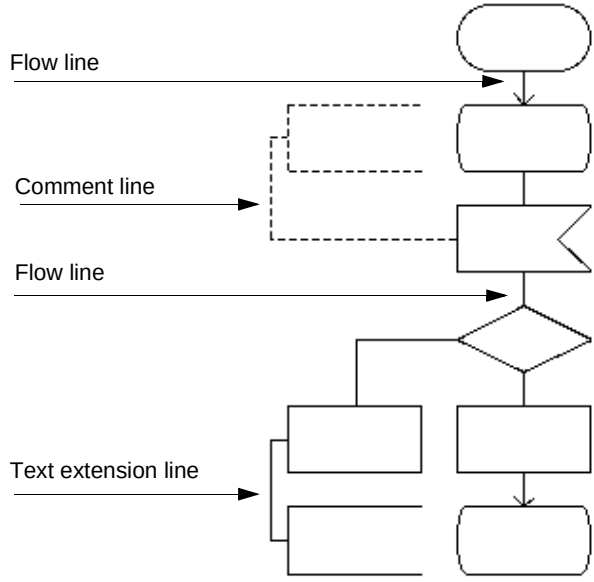


Figure 368: Lines (3)

On flow diagrams, the following lines are defined (see Figure 368):

- flow lines
- comment lines
- text extension lines

Line segments on flow pages always use a 90-degree angle. You can add segments on a line, but connection points are fixed.

Syntax Checking on Lines

The SDL Editor checks that the target symbol is in accordance with the syntactic rules imposed by SDL. If not, the SDL Editor will refuse to connect the symbol. When you edit the text attributes are instantly checked for syntactical errors.

Textual Objects

Textual objects are the textual attributes that are related to a symbol or a line. Each of these attributes is prepared by the SDL Editor – you need of course to fill in their textual contents.

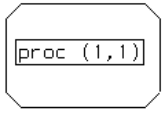
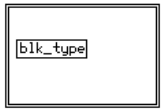
Textual attributes are indicated by a small rectangle which appears upon selection of the symbol or line the attribute belongs to.

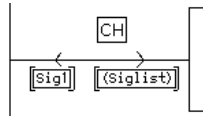
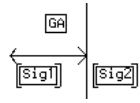
You can select and edit textual objects. You can move them freely, as long as their location does not violate the rules defined by Z.100.

Textual objects are allowed to overlap any other objects.

Textual Objects in Interaction Diagrams

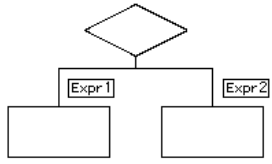
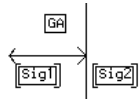
The following textual objects are defined in interaction diagrams:

Graphical Appearance	Textual objects
	• Name of a diagram reference symbol – Includes the *number of instances* for a process reference symbol
	• Name of a diagram type reference symbol
	• Name of a diagram instantiation symbol (e.g. `inst:blk_type`) – Names of graphical connection points in diagram instantiation symbol (e.g. `CON1`)

Graphical Appearance	Textual objects
	• Name of a channel or signal route (e.g. `CH`) – Signal list on a channel / signal route (there are 2 signal lists if the line is bidirectional, e.g. `Sig1` and `(Siglist)`) – Connection point for channel / signal route to / from environment, e.g. `CON`
	• Name of a gate, e.g. `GA` – Signal lists on a gate (there are 2 signal lists when the gate is bidirectional, e.g. `Sig1` and `Sig2`)

Textual Objects in Flow Diagrams

In flow pages, the following textual attributes are defined:

Graphical Appearance	Textual objects
	• Decision expressions connected to a decision symbol or transition option, e.g. `Expr1` and `Expr2`.
	• Name of a gate, e.g. `GA` – Signal lists on a gate (there are 2 signal lists when the gate is bidirectional, e.g. `Sig1` and `Sig2`

Graphical Connection Points

Connection points are text objects which are created automatically by the SDL Editor when you draw a channel or signal route to the frame or to an instantiation symbol.

They are handled in a similar way as other text attributes. You are free to fill connection points on the frame or not. The alternative is to define textual connection statements between channels and signal routes in a text symbol.

Change Bars

A change bar is a vertical line to the left of a text in an SDL diagram. It visually identifies changes in the SDL system.

A solid change bar indicates that the text has been changed. A dotted change bar indicates that a text or its associated symbol has been moved.

The change bars associated with the diagram name and the page name act as change bars for the complete page. As soon as a change has been made to a page, change bars for the diagram name and the page name are added.

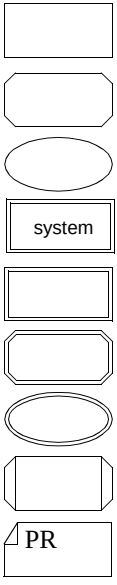
You can turn change bars on and off and also clear them from the Organizer. See "Change Bars" on page 145 in chapter 2, *The Organizer*.

Working with Symbols

The main operations provided by the SDL Editor on symbols is described in this section.

- *Working with Diagram Reference Symbols*
- *Working with Diagram Instantiation Symbols*
- *Working with Dashed Symbols*
- *Selecting Symbols*
- *Adding Symbols*
- *Inserting a Symbol into a Flow Branch*
- *Navigating in Flow Diagrams*
- *Navigating from Symbols*
- *Cutting, Copying and Pasting Symbols*
- *Moving Symbols*
- *Resizing Symbols*
- *Mirror Flipping Symbols*
- *Adjusting Symbols to the Grid*
- *Editing the Diagram Kernel Heading*.

In addition, text editing functions are provided for the text associated with the symbols.



Working with Diagram Reference Symbols

When you perform certain modifications to the diagram reference symbols listed in the left margin (*system type, block, block type, substructure, service, service type, process, process type, procedure and text reference symbols*), the SDL Editor checks with the Organizer to verify that the modifications are valid. This occurs in the following situations:

- *Adding a Diagram Reference Symbol*
- *Adding an Operator Diagram Reference Symbol*
- *Renaming a Diagram Reference Symbol*
- *Removing Symbols*

Working with Diagram Instantiation Symbols

Instantiating a Diagram Type

The implication of an instantiation symbol is that the description of the type diagram is actually instantiated, i.e. a physical copy is created. This

copy may then be given additional properties using SDL's inheritance and specialization mechanisms.

When a diagram type is to be *instantiated*, do as follows:

1. Make sure the type diagram is visible according to the SDL scope rules.
2. Insert a diagram reference symbol. When specifying the name of the symbol, use the syntax `<instance_name>:<type_name>`.
3. When the symbol is deselected, the diagram structure is immediately updated to show the instantiation (this is only true if the *Instance Diagram* option is on in the Organizer's *View Options*).

Working with Dashed Symbols

Dashing and Undashing a Reference Symbol

Dashed reference symbols are used to refer to an object that is defined elsewhere in one of the supertypes of the current subtype.

To *dash* a reference symbol:

1. Select the reference symbol.
2. Select *Dash* from the *Edit* menu.

Selecting Symbols

With the SDL Editor, you often perform actions that apply on a selection. This section discusses topics related to selection of symbols.

Selecting a Flow of Symbols

You often want to select a flow of subsequent symbols in a flow page, including all branches - the *tail* of the flow. To select a flow of symbols:

1. Click on the symbol where the flow starts.
2. Select *Select Tail* from the *Edit* menu. The tail of the flow is selected. See Figure 369, where the left picture shows only one selected object and its associated connections, whereas the right picture shows the selected symbol and its connections plus the selected rest of the flow diagram – the *Selected Tail*.

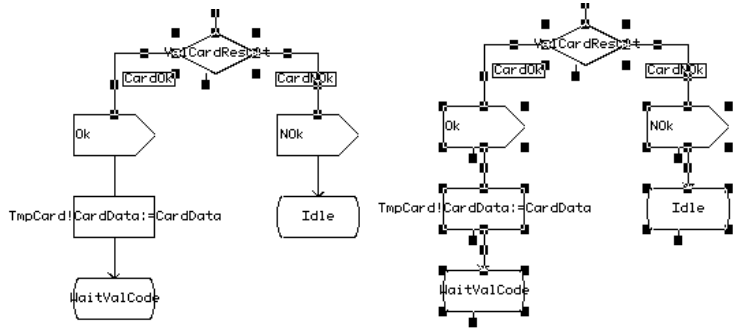


Figure 369: Selecting the tail of a subtree

Selecting a Symbol That Has Associated Objects

Consider the following when you select a symbol which has lines or text associated with it:

- When you select a symbol, you implicitly select the associated objects as well. A text cursor is inserted directly in the text associated with a symbol, and the *text window* will be updated to contain the text.
- If you point to a line associated with a symbol, you will select the line, **not** the symbol. The process of selecting lines is described in "Selecting Lines" on page 1866.
- If you point to the text associated with a symbol, the resulting selection depends upon the type of the symbol.

Displaying Selected Symbol Attributes

Selected symbol attributes are displayed as follows:

- When you select a symbol from which you can draw lines, it is displayed with a *handle* which you use when drawing lines from that symbol. Handles are shown in Figure 370.

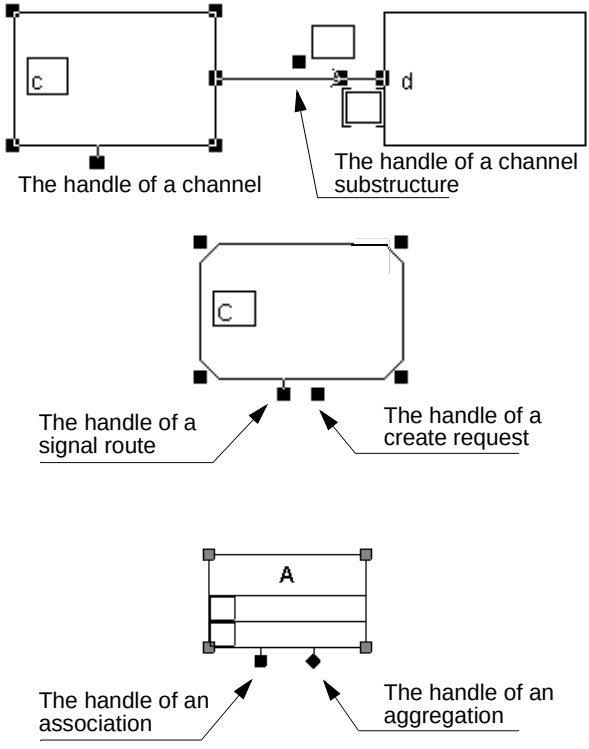


Figure 370: A selected symbol's handles

- *Text attributes* associated with a selected symbol are marked by a rectangle around the text, as shown in Figure 371. For class symbols, this only applies when text attributes contain no text.

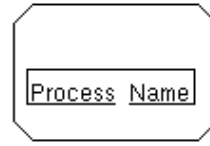


Figure 371: A selected symbol's text attributes

- In an interaction page, a line is marked by drawing filled *selection squares* at the line's starting point, ending point, and each breakpoint. In the middle of each line segment is a tiny selection square that provides an easy way to create new breakpoints. See Figure 372.

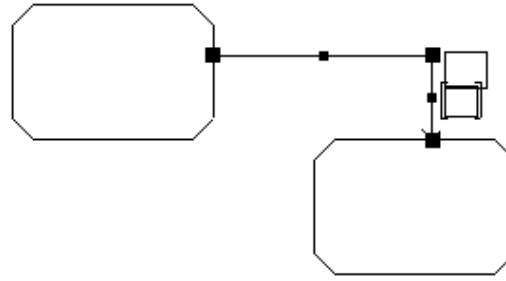


Figure 372: The selection squares of a line in interaction pages

- In a flow page, a selected flow line is marked by drawing filled *selection squares* in the middle of each line segment. This indicates that you can move the individual line segments (but not the flow line as a whole). See Figure 373.

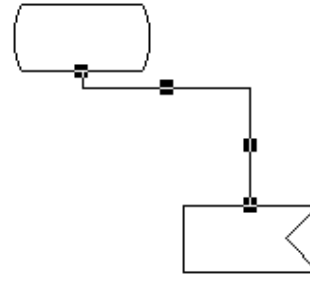


Figure 373: The selection squares of a line in flow pages

Adding Symbols

You may place symbols into the drawing area in either manual mode or in automatic mode.

Adding Symbols in Manual Mode

This section describes how to do this in manual mode. The automatic mode is described in "Adding Symbols in Automatic Mode" on page 1842.

Selecting and Placing a Symbol

1. Click the symbol in the symbol menu.
2. Click at the desired location in the drawing area. Overlap of symbols is not allowed.

   The symbol is drawn and adjusted to the symbol grid.

Entering Symbol Text Attributes

Once a symbol has been added, you can enter the text into the symbol's text attributes. The symbol remains selected. Enter the text required directly using the keyboard or edit the text inside the *text window*.

Connecting the Symbol to Another Symbol

Once the symbol is at the desired location it can be connected to another symbol as required.

To connect two symbols:

> **Note:**
>
> The line will be drawn only when the SDL syntax rules allow the symbols to be interconnected.

1. Add the *source symbol*, i.e. the symbol from where you want to draw the line.
2. Point the mouse onto the handle of the source symbol, and start dragging the handle. As soon as mouse motion has begun, you can release the mouse button. Point to the *target symbol* and click the mouse button. The line is drawn to the symbol and required text attributes are added to the connecting line. The box without brackets

is reserved for the name of the channel, while the box with brackets [ ] is for the signal list.

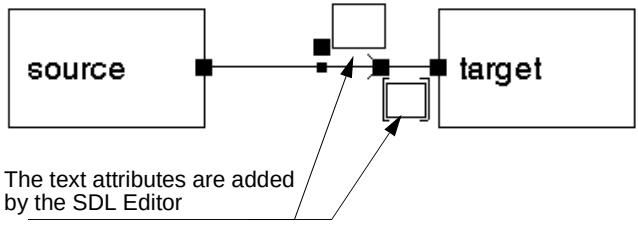


Figure 374: Having connected two symbols

3. Fill in the text attributes.

Conversely, once you have placed a symbol in the drawing area, you can follow the same procedure from a symbol resident in the drawing area to the new symbol.

Connecting the Symbol to the Environment

1. Hold the mouse pointer on the handle of the symbol, drag it to the frame (environment) and release the mouse button.
2. Enter text in the channel or signal route and connection point boxes.

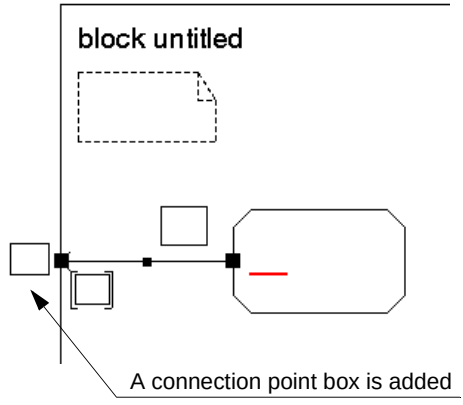


Figure 375: Having connected the symbol to the environment

Adding Symbols in Automatic Mode

The automatic mode is particularly useful when you want to add symbols sequentially or in parallel in a symbol flow. Both situations are described below.

Adding in Sequence

To add symbols sequentially within a flow page, proceed as follows:

1. Select the source symbol in the drawing area.
2. Double-click the *target symbol* in the *symbol menu.*

   The target symbol is added to the source symbol, and a flow line is drawn between the two symbols if the syntax rules allow it.
3. Fill in the text in the target symbol.

The target symbol is positioned directly beneath the source symbol if possible. Otherwise, the added symbol is placed in the first available location to the right.

After this operation, the target symbol is automatically selected and the source symbol is de-selected.

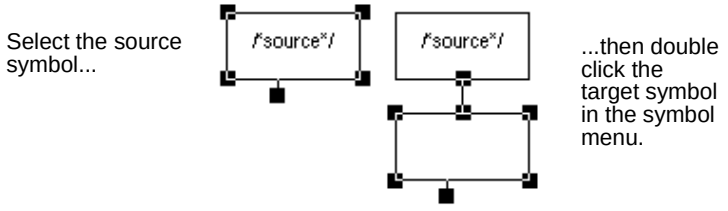


Figure 376: Adding in sequence

Adding in Parallel

To add symbols in parallel in a flow page:

1. Select the *source symbol* in the drawing area.
2. Press `<Shift>` and double-click the *target symbol* in the *symbol menu*.

   The target symbol is added to the selected symbol. The source symbol remains selected.

You can repeat this procedure in order to add additional symbols to the selected symbol. The target is positioned immediately below the source symbol, if possible. Otherwise, the target symbol is placed in the first available location to the right.

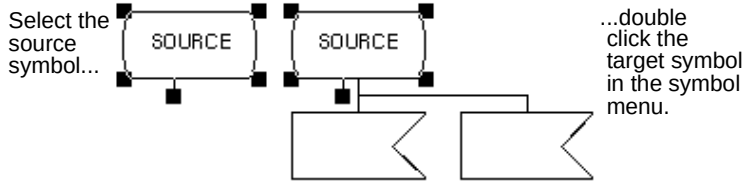


Figure 377: Adding in parallel

Using Default Placement

You can also take advantage of the auto-placement by double-clicking a symbol without having to connect it to other symbols.

Insufficient Space Left

When you insert a symbol using the double-click facility, there may not be any space left to perform the operation, The SDL Editor informs you of this with a message.

You have the following options:

- Click *OK* to increase the page. The *Drawing Size* dialog will be opened, where you can specify the new size of the SDL Page.

  If you select this method, the SDL page may not fit any longer into the paper that you have set up with the Print preferences. The result may be an SDL page that requires multiple sheets of paper when printing it (unless you rescale it).

- To *Cancel* the operation. Try to find a more suitable location for the new symbol and use manual placement (see "Adding Symbols in Manual Mode" on page 1840).

Adding a Gate Symbol

The gate symbol differs slightly from other symbols when you add it.

Adding a Gate in Manual Mode

1. Select the gate symbol.
2. Move the pointer into the drawing area. The gate is immediately displayed and connected in a 90-degree angle to the frame and the environment.
3. Move the mouse to a suitable location. The gate symbol follows the mouse motion and is automatically reconnected.
4. Click the mouse.
5. Enter the name of the gate and the signals it conveys.

Note: Coalesced Gates and Connection Points

See "Connecting a Gate Symbol with a Channel or Signal Route" on page 1853 if you get a message with the text "A Gate has coalesced with a connection point...".

Adding a Gate in Automatic Mode

You may add and connect a gate to a channel or signal route with a double-click. The channel or signal route must not be already connected to a gate or connection point.

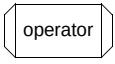
1. Select the channel or the signal route.
2. Double-click the gate symbol in the symbol menu.
3. If required, redirect or bidirect the gate so that it matches the direction of the channel or signal route.
4. Enter the name of the gate and the signals it conveys.

Adding a Diagram Reference Symbol

When you add a diagram reference symbol, the SDL Editor checks that the name and type are unique within the SDL diagram. The SDL Editor will refuse any duplicates. The newly added reference symbol is then inserted into the Organizer's diagram structure.

1. Select the symbol to be added from the symbol menu and add it to the diagram.
2. Type in the name of the symbol.
3. Deselect the symbol. The SDL Editor checks that the name and type are unique within the SDL diagram being edited.

   If the new reference symbol is accepted, it is inserted into the diagram structure, located beneath the referring diagram.



Adding an Operator Diagram Reference Symbol

In a data type construct it can be stated that an operator diagram is referenced elsewhere. This means that the operator will not be described in the current place, but in an operator diagram.

The SDL Editor does not know that the operator is referenced in a data type. The operator reference symbol is available for this situation. It is a convenience that has been added to the SDL suite tool set to facilitate navigating within an SDL system. Its presence in an SDL diagram does not modify the meaning of the diagram.

You add an operator reference symbol in a similar manner as when adding any other reference symbol. See "Adding a Diagram Reference Symbol" on page 1845.

Adding a Text Reference Symbol

The Text reference symbol is used to include SDL PR into the SDL system.

You add an operator reference symbol in a similar manner as when adding any other reference symbol. See "Adding a Diagram Reference Symbol" on page 1845.

Inserting a Symbol into a Flow Branch

The SDL Editor allows you to insert symbols into a flow branch. When inserting symbols, the SDL Editor will rearrange the page in order to prepare the required space for inserting the symbol. This can be done in different ways:

- By pushing the tail of the branch downwards. This is the preferred method.
- By increasing the size of the SDL page.

Inserting a Symbol into a Flow Branch

1. Select the symbol in the symbol menu.
2. Point to the line interconnecting the symbols where the new symbol is to be inserted.
3. Click the mouse.

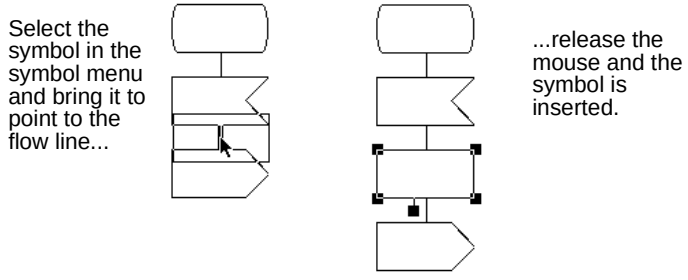


Figure 378: Inserting a symbol into a flow branch

Inserting a Symbol by Double-Click

To insert a symbol into a flow branch by double-clicking it:

1. Select the line that interconnects the two symbols. Alternatively you can select the symbol preceding the line, see "Insert Paste" on page 1941.
2. In the symbol menu, double-click the symbol to be inserted.

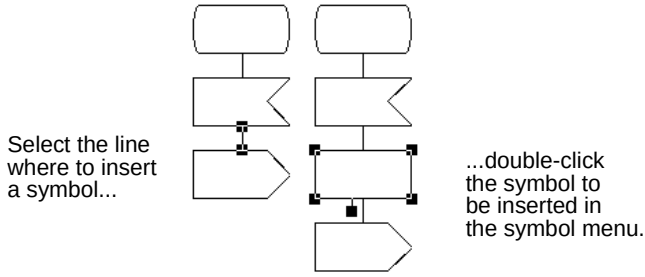


Figure 379: Inserting a symbol with double-click

Navigating in Flow Diagrams

For certain symbols you can use the *Navigate* command in the *Tools* menu or double click on the symbol to navigate around in the diagrams. These symbols are:

- Create request, procedure call and macro call
- State and nextstate
- Inconnector and outconnector
- Reference symbols; see further "Navigating from a Diagram Reference Symbol" on page 1848

> **Note: Navigating from a Class Symbol**
>
> Navigating from a class symbol is possible only by using *Tools>Navigate*. See further "Navigating from Class Symbols" on page 1849

Navigating from Create Request and Procedure/Macro Call

For a create request, procedure call or macro call the *Navigate* command will show the corresponding process, procedure or macro diagram. The diagram is found by searching in the Organizer structure for all relevant diagram types matching the name used in the symbol. If one and only one match is found this diagram will be shown in the editor. If there exists more than one diagram a dialog is issued.

Navigate to the desired diagram by selecting it in the dialog.

Navigating from State/Nextstate

For a state or nextstate the *Navigate* command will show other occurrences of the same name(s) in other state or nextstate symbols within the same diagram. If only one symbol matches the name search this symbol will be shown. If more than one symbol have the same name, a dialog is shown.

Navigating from Connectors

For an outconnector, the *Navigate* command will show the corresponding inconnector in the same diagram having the same name as the outconnector.

Similarly for an inconnector; you can navigate to all occurrences of outconnectors within the same diagram having the same name as the inconnector.

Navigating from Symbols

Navigating from a Diagram Reference Symbol

When you double-click or choose the Navigate command on a diagram reference symbol, the SDL diagram that the reference symbol refers to is opened, provided that the diagram is included in the Organizer structure.

This feature allows easy navigation down an SDL hierarchy.

See also "Navigating from an Instantiation Symbol" on page 1848 and "Navigating from a Dashed Symbol" on page 1849.

Navigating from an Instantiation Symbol

You may use instantiation symbols to navigate in the diagram structure.

When you choose *Navigate* in the *Tools* menu or double-click on an instantiation symbol, the SDL Editor matches the name of the type referred to in the instantiation symbol against the existing type diagrams in the Organizer structure. The following happens depending on how many type diagrams that match the given diagram name.

- **One and only one diagram is found**: This diagram will be shown.

- **No diagram matches the name**: A dialog will appear which offers the possibility to open the Type Viewer window which can be used to navigate among the inheritance and redefinition trees extracted from the SDL system:
- **More than one diagram is found**: A dialog will appear showing the file names of all the found diagrams. Additionally the possibility to show the Type Viewer window is available.

Navigating from a Dashed Symbol

When you select *Navigate* in the *Tools* menu or double-click a dashed symbol, a scenario similar to the action of navigating from an instantiation symbol takes place. See "Navigating from an Instantiation Symbol" on page 1848

Navigating from Class Symbols

For a class symbol, the *Navigate* command will show other occurrences of the same name(s) in other class symbols within the same diagram. If only one symbol matches the name search, this symbol will be shown. If several symbols have the same name, a select dialog is shown.

Cutting, Copying and Pasting Symbols

The clipboard of the SDL Editor allows you to cut or copy a selection of symbols. The selection may then be pasted to:

- The same page
- Pages of the same diagram, provided the page types match
- Other diagrams, provided the diagram types match

See also "Cutting, Copying and Pasting Reference Symbols" on page 1851.

Cutting and Pasting Symbol(s)

A dialog may appear when cutting or pasting an object with link endpoints. See "Deleting an Object" on page 461 in chapter 10, *Implinks and Endpoints* or "Pasting an Object" on page 461 in chapter 10, *Implinks and Endpoints*.

Limitations when Pasting

> **Note: Effect of Syntax Check when Pasting**
>
> Some symbols may not be pasted if the syntax check is on. The clipboard buffer content must comply with the SDL syntax rules. Any **symbols which do not comply will be omitted**. (The clipboard buffer content may have been copied from a page edited with syntax checking turned off.)

Pasting is not always possible. Note the following cases, and refer to "Relationship between Diagrams and Pages" on page 1782.

- The selection cannot be pasted if the page is too small. In this case, you need to enlarge the page. See "Resizing a Page" on page 1892.
- You are not allowed to paste symbols so that they overlap existing symbols in the drawing area. If you attempt to do this, the *Paste* operation is cancelled. Try another suitable location.
- You are not allowed to paste flow symbols into an interaction page and vice versa. In this situation, the *Paste* menu choice is dimmed.

Pasting on Flow Pages

On a flow page, symbols are pasted along with the flow lines that interconnect them.

Pasting on Interaction Pages

On an interaction page, symbols are pasted along with the lines that interconnect them. Any lines that connect the pasted symbols to the environment will also be drawn.

Pasting Symbols into a Flow Branch

On a flow page, symbols can be pasted into a flow branch in a similar manner as "Inserting a Symbol into a Flow Branch" on page 1846.

1. Copy or Cut one or more symbols in a flow page. The symbols must be connected and placed below each other.
2. Go to the page where you want to insert the symbols and select a symbol or a flowline.
3. Select *Insert Paste* from the *Edit* menu.

4. The pasted symbols will be placed below the selected symbol and inserted into the flow branch. Any previously following symbols will be automatically shifted downwards to make place for the inserted symbols.

Cutting, Copying and Pasting Reference Symbols

When you cut or copy a reference symbol, the possible sub-hierarchy shown in the Organizer below this reference symbol is also placed in the clipboard. When pasted in the same diagram or in another diagram this sub-hierarchy will be restored at the new position.

> **Note:**
>
> When a reference symbol is cut, the sub-hierarchy that is shown in the Organizer will be removed. However, the hierarchy will be restored if you select *Paste* or *Undo*.

When you copy or paste a diagram reference, it is assigned a new name that consists of the original name, preceded with the prefix *CopyOf*. The prefix is repeated as many times as necessary in order to build a name / type combination that is unique within the SDL diagram being edited. If the cut/copied symbol had an underlying structure this structure is restored after the paste.

Cutting, Copying and Pasting Class Symbols

When cutting, copying or pasting a class symbol, one or more operators with a defined body code may be completely removed, i.e. there is no longer any class symbol that refers to a specific operator. If so, the following dialog is shown:

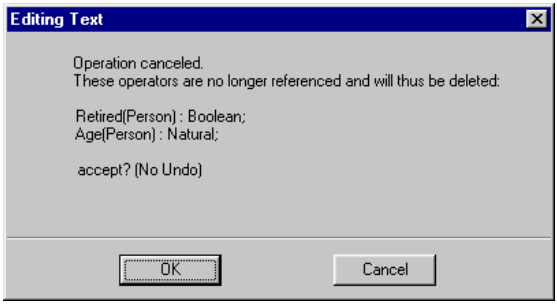


Figure 380: Checking with user if OK to remove operators

You have the following options:

- Accept the deletion of the listed operators and their defined body code by pressing *OK*.
- Cancel the issued operation by pressing *Cancel* and try to correct the error.

Caution!

By confirming the deletion of the operators, you accept that the listed operators, and possibly defined body codes, will be permanently removed. No *Undo* option is available.

Moving Symbols

You may move one or more symbols in a single operation. The lines and text objects that are related to the symbol will be moved accordingly in order to preserve as much as possible of the original appearance. When you move a symbol, its position is automatically adjusted to fit exactly into the symbol grid.

Symbols are not allowed to be moved to positions where they would overlap other symbols.

Moving Symbols and Text Attributes

You can move symbols and text attributes by pointing and dragging to the desired position. To grab the text attribute the pointer must be at the border of the text extent.

You cannot move text within flow symbols and class symbols.

Connecting a Gate Symbol with a Channel or Signal Route

When you move a gate symbol in order to reconnect it to a line or vice versa (channel or signal route that is connected to a connection point), the following takes place:

- If the connection point is not assigned any name, the SDL Editor connects the gate symbol and the line.
- If the connection point is assigned a name, a conflict occurs, since the line cannot be connected to the connection point and the gate simultaneously. The following dialog is issued:

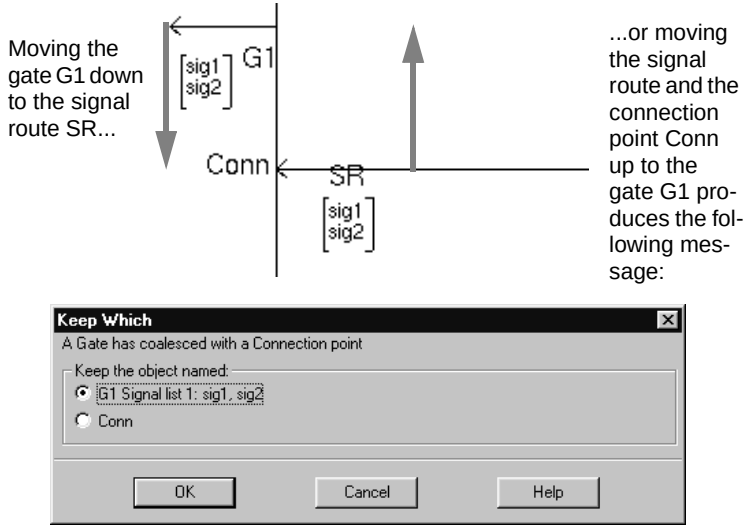


Figure 381: Prompting to solve connection conflict

You have the following options:

- To keep the gate G1 and to connect the line to it. The connection point Conn is removed.
- To keep the connection point Conn and to connect the line to it. The gate G1 is removed.

- Click *OK* to reconnect the line and symbol accordingly.

Note:

The situation above also occurs if you add a new gate symbol and place it on a connection point.

Resizing Symbols

The procedure for resizing a symbol differs slightly depending on whether the symbol is a flow page symbol or an interaction page symbol. When you resize a symbol, its size is automatically adjusted to fit exactly into the symbol grid. Any lines connected to the symbol are similarly adjusted to preserve as much as possible of the original appearance.

The text, the additional heading, the package reference, comment, class, and text extension symbols are handled differently. Their size is determined by the SDL Editor to fit the text they contain.

Specifying Default Size of Symbols

You define the default sizes for symbols in the Preference Manager:

- For interaction pages, the predefined size of a symbol is 30 * 20 mm.
- For flow pages, the predefined size of a symbol is 20 * 10 mm. The width / size relationship is always 2:1.

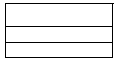
Resizing Interaction Page Symbols

1. Point to a corner of the rectangle that delimits the symbol. From this point on, the corner that you are pointing to is movable, while the opposite corner is fixed.
2. Drag the corner until the symbol has the desired size. An interaction symbol cannot be adjusted to a size less than 15*10 mm.

3. Release the mouse button. The symbol is redrawn with its size adjusted to fit exactly into the symbol grid. The lines connected to a resized interaction symbol are automatically adjusted to preserve a nice appearance.

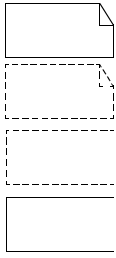
Resizing Flow Page Symbols.

1. Point to a corner of the rectangle that delimits the symbol. From this point on, if you pointed to any of the lower corners, the top of the symbol is fixed (and in a similar way, if you were pointing to any of the upper corners, the bottom of the symbol is fixed).
2. Drag the corner until the symbol has the desired size. The symbol shrinks or swells both horizontally and vertically, the proportions remaining 2:1. A flow page symbol cannot be adjusted to a size less than 20*10 mm.
3. Release the mouse button. The symbol is redrawn.



Resizing the Class Symbol

The class symbol contains three text boxes. Whenever the text in any of the boxes grows bigger than the symbol, the class symbol is automatically re-sized.



Resizing Text / Additional Heading / Package Reference / Comment / Text extension Symbols

The text, additional heading and package reference symbol look like a piece of paper. When all of the text within a text symbol is in view, the upper right corner is “folded” down. When any portion of the text within a text symbol cannot be seen (because the text symbol is too small), the upper right corner is clipped or diagonally cut off. See Figure 382 on page 1856.

Clipped text symbols are printed in their whole on a separate sheet of paper.

These symbols react in the same way for resizing. There are several ways you can resize these symbols:

Adjusting to a Specific Size (only for Text / Additional Heading / Package Reference)

1. Place the mouse pointer on the filled selection square at the bottom right corner of the symbol.
2. Holding the left mouse button down, make the symbol smaller by pushing to the left or larger by pulling the corner to the right.

Toggling Between Minimum and Maximum Size

You may rapidly toggle between a text symbol's maximum and minimum size. This can be accomplished in two ways:

- Select the symbol and choose the *Collapse/Expand* command in the *Edit* menu.
- Double-click on any part of the symbol.

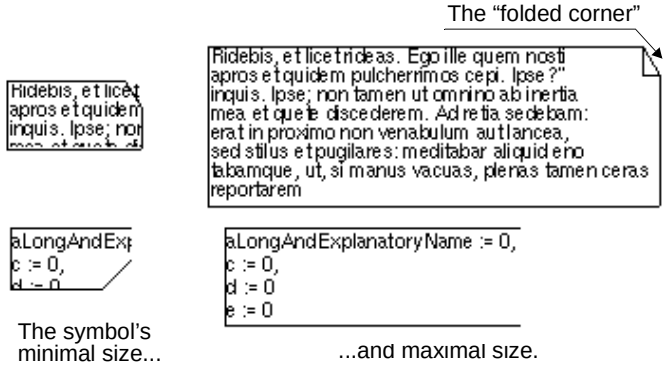


Figure 382: Minimizing and maximizing a symbol

Mirror Flipping Symbols

The input, priority input, output, comment and text extension symbols are drawn by default as displayed in the symbol menu. In some cases, the layout of the connecting lines would be nicer if the symbols were drawn flipped L/R.

1. Select the symbol.
2. Choose *Flip* from the *Edit* menu. The symbol is flipped L/R.

Flipping the text extension symbol mirrors it and redraws the connecting line

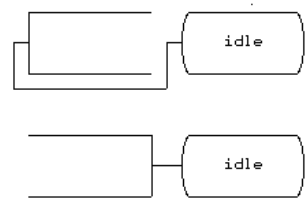


Figure 383: Flipping a symbol

Note:

By using the Preference Manager you can specify whether the output symbol in the symbol menu should point to the left or to the right, or you can have both. The same applies for the input and priority input symbols.

Adjusting Symbols to the Grid

The drawing area of the SDL Editor provides two grids for easier positioning of objects. These are the *symbol grid* and the *line grid*, which are invisible.

After you move or resize symbols, the position of the upper left corner of each symbol is positioned on the closest intersection in the symbol grid. In addition, resizing a symbol will adjust the size so that it matches the resolution of the grid, horizontally as well as vertically. The resolution of the symbol grid is variable and can be set with the Preference Manager. The default is 5 * 5 mm for interaction pages, and 25 mm * 15 mm for flow pages.

You can temporarily disable the grid by toggling the *Use Grid* to off in the *Diagrams Options* dialog.

Handling Symbols of Different Sizes

Since symbols adhere to the symbol grid, mixing symbols of different sizes may imply that symbols cannot be aligned symmetrically, and that enlarging symbols becomes impossible due to overlap of symbols already laid out.

You can align symbols of different sizes by disabling the symbol grid.

Editing the Diagram Kernel Heading

You can rename, retype and re-qualify a diagram with the SDL Editor, by modifying the diagram's kernel heading.

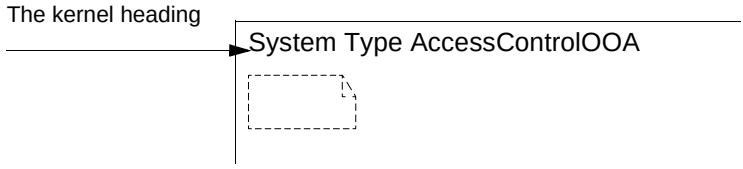


Figure 384: The kernel heading

See also "Transforming the Type of a Diagram" on page 1812 for more complicated transformation of diagram types.

The SDL suite accepts a number of syntaxes for kernel headings, 13 in all. See "The Kernel Heading" on page 1792.

Note:

Changing the kernel heading of a diagram may have a large impact on the diagram's meaning. Also, the SDL diagram structure where the diagram is included may be affected to a large extent. You may, however, want to change the kernel heading in order to create a new diagram that you will use in a different context than the SDL structure it originally is tied to, or in order to perform a simple rename of diagrams.

Changing the Name of a Diagram

This is the bottom-up method you would use when renaming diagrams in an SDL structure.

1. Select the kernel heading.
2. In the text window, edit the kernel heading text.

3. Deselect the kernel heading. The SDL Editor checks that the name is still valid according to SDL rules. Any syntax error will cause a dialog with the following alternatives to be issued:

   – Clicking *OK*, you are returned to the text window where you should correct the error.

   – Clicking *Revert*, the kernel heading is reverted to its original contents.

   – Clicking *Ignore*, the erroneous kernel heading is accepted. The error will probably lead to other kind of error reports, typically when verifying the diagram with the Analyzer tool.

   The Organizer detects that the name of the diagram no longer matches the name that is expected in the diagram structure, and marks the diagram icon as mismatch. (A warning is also added to the Organizer Log window.)

4. To correct the mismatch, you have to update the name of the reference symbol accordingly. Use the command *Edit Reference Page* to bring the page with the reference symbol into view. Locate the reference symbol and rename it accordingly.

For renaming diagrams in a top-down fashion., see "Renaming a Diagram Reference Symbol" on page 1863.

Changing the Type of a Diagram

It is possible to re-type a diagram, provided the old and the new types are "compatible" from the SDL Editor's point of view. The basic compatibility criteria is that the old and new type of diagram must both be interaction diagrams or both be flow diagrams. Also checks are made on the pages that are contained in a diagram (some diagram types are allowed to have different types of pages in the same diagram, see "Relationship between Diagrams and Pages" on page 1782. The table below summarizes the type changes that are allowed.

case	A diagram of type...	...can be changed to any of:
1.	SYSTEM	SYSTEM TYPE BLOCK SUBSTRUCTURE BLOCK TYPE PACKAGE

case	A diagram of type...	...can be changed to any of:
2.	BLOCK	BLOCK TYPE SYSTEM SUBSTRUCTURE SYSTEM TYPE PACKAGE
3.	PROCESS	PROCESS TYPE SERVICE PROCEDURE SERVICE TYPE MACRODEFINITION OPERATOR
4.	SERVICE	SERVICE TYPE PROCESS PROCEDURE PROCESS TYPE MACRODEFINITION OPERATOR
5.	PROCEDURE	PROCESS SERVICE PROCESS TYPE SERVICE TYPE MACRODEFINITION OPERATOR
6.	SUBSTRUCTURE	SYSTEM BLOCK SYSTEM TYPE BLOCK TYPE PACKAGE
7.	MACRODEFINITION	PROCESS SERVICE PROCEDURE PROCESS TYPE SERVICE TYPE OPERATOR
8.	SYSTEM TYPE	SYSTEM BLOCK SUBSTRUCTURE BLOCK TYPE PACKAGE
9.	BLOCK TYPE	SYSTEM BLOCK SUBSTRUCTURE SYSTEM TYPE PACKAGE

case	A diagram of type...	...can be changed to any of:
10.	PROCESS TYPE	PROCESS SERVICE PROCEDURE SERVICE TYPE MACRODEFINITION OPERATOR
11.	SERVICE TYPE	SERVICE PROCESS PROCEDURE SERVICE TYPE MACRODEFINITION
12.	OPERATOR	PROCESS SERVICE PROCEDURE PROCESS TYPE SERVICE TYPE MACRODEFINITION
13.	PACKAGE	SYSTEM BLOCK SUBSTRUCTURE SYSTEM TYPE BLOCK TYPE

To change the type of a diagram:

1. Select the kernel heading.
2. In the text window, edit the kernel heading text to apply a new type. See the table above for allowed transformations.
3. Deselect the kernel heading. The SDL Editor checks that the type is still valid according to the table above. Any violation of these rules will cause a dialog with the following alternatives to be issued:
   - Clicking *OK*, you are returned to the text window where you should correct the error.
   - Clicking *Revert*, the kernel heading is reverted to its original contents.
   - Clicking *Ignore*, the erroneous kernel heading is accepted. The error will probably lead to other kind of error reports, typically when verifying the diagram with the SDL Analyzer.

Changing the Qualifier of a Diagram

By changing the qualifier that appears in the kernel heading, you can change the context of the diagram.

The SDL suite accepts the two Z.100 notations for a qualifier, namely with and without the surrounding "<< >>".

Example 304: An SDL Qualifier ––––––––––––––––––––––––––––––

```
PROCESS TYPE <<SYSTEM MYSYS/BLOCK MYBLOCK>> Myproc
```

Alternatively:

```
PROCESS TYPE SYSTEM MYSYS/BLOCK MYBLOCK Myproc
```

––

To change the qualifier:

1. Select the kernel heading.
2. In the text window, edit the kernel heading text to apply the new qualifier.

3. Deselect the kernel heading. The SDL Editor checks that the new qualifier is syntactically correct. Any syntax error will cause a dialog with the following alternatives to be issued.

   – Clicking *OK*, you are returned to the text window where you should correct the error.

   – Clicking *Revert*, the kernel heading is reverted to its original contents.

   – Clicking *Ignore*, the erroneous kernel heading is accepted. The error will probably lead to other kind of error reports, typically when verifying the diagram with the SDL Analyzer.

Renaming a Diagram Reference Symbol

When you change the name of a diagram reference symbol, the SDL Editor checks that the new name is not in conflict with existing symbols. If the reference symbol passes this check, the Organizer structure is updated accordingly.

When renaming diagrams in a top-down fashion, you start by changing the name of a reference symbol. After the reference symbol is renamed, you should also update the referenced diagram's kernel heading accordingly to maintain consistency in the diagram structure.

To rename a diagram reference symbol and the referred diagrams:

1. Select the symbol to be renamed.
2. Change the name of the reference symbol.
3. Deselect the symbol. The SDL Editor checks that the name and type are still unique within the diagram.

   If the new name of the reference symbol is accepted, the diagram structure is updated to reflect the nature of the change.

4. To ensure that the change also applies to the referred diagram, you should use the Organizer command *Update Headings* (see "Update Headings" on page 99 in chapter 2, *The Organizer*). This command verifies and possibly changes the name in the diagram kernel heading against the expected name (which is the one that now is shown in the reference symbol). The mismatch between the name of the diagram is reported in a dialog:

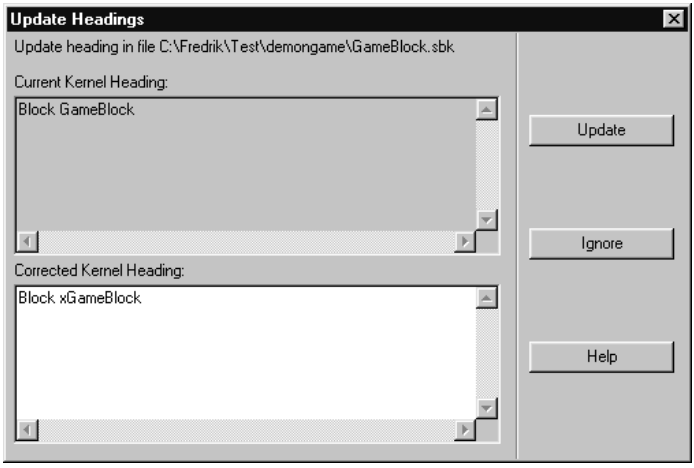


Figure 385: The mismatch in headings is reported

– Correct the mismatch by clicking *Update*.

Note that if you select not to update the kernel heading, you simply postpone the problem.

For renaming in a bottom-up way, see "Changing the Name of a Diagram" on page 1858.

Removing Symbols

When you remove a diagram reference symbol, the underlying structure is also removed from the Organizer structure. However, the hierarchy will be restored if the remove operation is restored by doing Undo.

1. Select the symbol(s) to be removed.
2. Select *Clear* from the *Edit* menu. The symbol(s) are removed, as well as the lines that were connected to these symbols.

Printing Symbols

A selection of symbols can be printed. This allows for instance to print only the portion of a diagram which is of interest.

1. Select the symbols to be printed.
2. Select *Print* from the *File* menu.
3. In the *Print Dialog*, click the *Setup* button. The Print Setup dialog will be issued. In the dialog, select the *Print only selected symbols* option.
4. Adjust, if necessary, other *Print options* and click the *OK* and *Print* buttons. See "Printing a Diagram" on page 1812 for more information.

Working with Lines

The main operations provided by the SDL Editor on lines, and their associated text attributes are:

- *Selecting Lines*
- *Drawing Lines*
- *Re-Routing and Reshaping Lines*
- *Redirecting and Bidirecting Lines*
- *Adjusting Lines to the Grid*

In addition, text editing functions are provided for the text associated with the lines.

Operations on lines are performed slightly different depending if you are editing

- *flow pages*
- *interaction pages*

The main difference between the two is that flow pages may only allow vertical and horizontal lines.

Some restrictions also apply when connecting symbols in flow pages, depending upon whether or not the syntax checking is enabled

Selecting Lines

You select lines and extend and reduce selections of lines with standardized selection commands.

Selecting a Line That Has Associated Objects

When you select a line, you implicitly select the associated attributes as well. To be able to edit a text attribute you must click on the text attribute and a text cursor will be inserted directly in the text, at the same time the text window will be updated to contain the text.

Displaying Selected Line Attributes

Selected line attributes are displayed as follows:

- Each text attribute is displayed inside a text selection rectangle.

- The following attributes are displayed with a filled selection square:
  - Each arrow (on interaction pages only)
  - Each line breakpoint (on interaction pages only)
  - The middle of each line segment (on interaction pages only a tiny selection square marks the handle to create a new break-point)
  - The middle of each line segment (on flow pages only)

Selecting Overlapping Objects

On a page, objects may overlap each other. A *layer order* is defined so that it is possible to determine how overlapping objects are stacked.

Drawing Lines

This section describes how to interconnect symbols with lines. Two modes are supported in the SDL Editor.

Drawing a Line with Layout Syntax Checking Enabled

The way that you draw a line differs somewhat depending upon whether you are working on an interaction page or a flow page. The differences will be pointed out later in this section.

When syntax checking is enabled, you are only allowed to draw lines that are correct according to the SDL rules. When syntax checking is disabled, the rules are less strict.

Designating the Start Point

1. Select the symbol from which the line will originate.
2. Point to the handle and drag it to begin forming the line. The line that is formed ends where the mouse pointer is located. Note that the mouse button can be released as soon as you have moved the mouse pointer.

Designating the End Point

1. Designate the start point. See previous section.
2. You can route the line and specify where it will end either automatically or manually.

- To route the line automatically:
  - Click any valid target symbol to specify where the line should end.

    A symbol is valid if the syntax rules enforced by the SDL Editor at edit-time allows the drawing of a line from the first symbol to the second symbol.
- To route the line manually, indicating where *line breakpoints* are to be inserted, follow the steps below.

  1. To enter manual mode, click any location in the drawing area other than a valid destination symbol.
  2. Click each location where you want to insert a line breakpoint.
  3. Click on a valid destination symbol.

     As long as you do not point to a valid destination symbol, the line will be routed from the selected symbol to the current mouse position. The automatic line drawing facility will determine the layout of the line until you have inserted the first breakpoint.

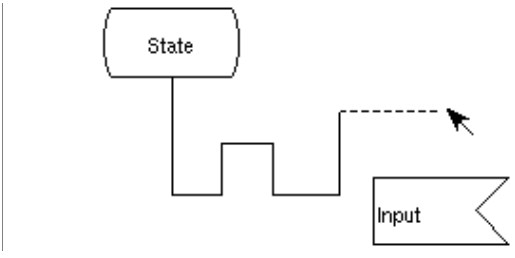


Figure 386: Drawing a line in manual mode

Drawing Multiple Lines

When you draw lines between symbols and keep the <Control> key pressed, the start point for the next line to draw is automatically transferred to the target symbol.

This feature provides the easiest way to draw multiple lines in rapid succession.

Differences between Flow and Interaction in Line Drawing

The way that you draw a line differs somewhat depending upon whether you are working on an interaction page or a flow page:

- On an interaction page, you can draw lines in all possible directions.
- On a flow page, you can only draw lines horizontally or vertically. The last line breakpoint, even if determined manually, will be adjusted so that the destination symbol is connected correctly, normally in the middle of the upper edge of the symbol. Exceptions are made for comment and text extension symbols, which are connected to the center of the right or left edge of the symbol.

Cancelling the Line Drawing Operation

You can cancel a line drawing operation in either of the following ways:

- Press <ESC>.
- Double-click any part of the drawing area that is not a valid destination symbol.

Drawing a Line with Layout Syntax Checking Disabled

The process of drawing a line is the same regardless of whether or not syntax checking is enabled. To enable and disable syntax checking, see "Turning Syntax Checking On and Off" on page 1786.

When drawing lines with syntax checking disabled, you may interconnect symbols in any way you want.

Re-Routing and Reshaping Lines

Reshaping a line refers to changing the path that the line follows. This operation does not affect the endpoints of the line. The flow lines only move in either a horizontal or vertical way.

Re-routing a line denotes changing the ending point of the line from one symbol to another symbol.

Reshaping a Flow Line

1. Point to one of the segments of the line.
2. Press the mouse and, holding down the mouse button, drag the segment to the required position.
3. Release the mouse button.

Re-Routing a Flow Line

You can move the ending point of a flow line, and reconnect it to any valid destination symbol. You can redraw the line either manually or automatically.

To move a line's ending point:

1. Point at the ending point that you want to reconnect.
2. Press the mouse. Start dragging the ending point. As soon as you move the mouse, the line's last segment follows the mouse motion and you can release the mouse button.
3. You can click additional *line breakpoints* where you like.
4. Click the destination symbol to complete the operation. If it is a valid symbol, the line will be re-routed automatically.

Re-Routing an Interaction Line

You can move any starting or endpoint to and from a symbol and the environment.

1. Point to the starting or ending point of the line.
2. Press the mouse and, holding the mouse down, drag it to the desired symbol.
3. If required, click to add line breakpoints at desired positions.
4. Terminate by clicking on the new symbol. The line is redrawn accordingly.

Moving an Interaction Line Breakpoint

To move a line breakpoint in an interaction page:

1. Point to the breakpoint that you want to move.
2. Press the mouse and drag the breakpoint to the desired location.
3. Release the mouse button to redraw the line.

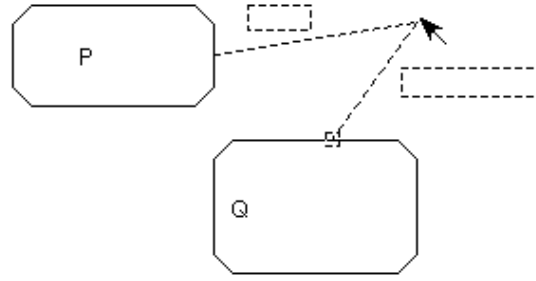


Figure 387: Moving an interaction line breakpoint

Inserting Breakpoints in an Interaction Line

1. Point at the middle of a line segment. When the line is selected a tiny selection square will be displayed at this point.
2. Press the mouse and drag it to shape the breakpoint.

An alternative way to insert breakpoints on an interaction line:

1. Press `<Ctrl>`
2. Point in a line segment where you want to insert a new breakpoint.
3. Press the mouse and drag it to shape the breakpoint.
4. Release the `<Ctrl>` key.

Moving an Interaction Line Segment

You can move a line segment in an interaction page as long as it is not a starting segment or ending segment.

1. Point to the line segment to be moved.
2. Press the mouse and drag the line segment to the desired location.

3. Release the mouse button. The line segment and the adjacent line segments are redrawn accordingly.

Moving Arrows

Arrows on channels, gates and signal routes denote the orientation of flow on that line. They are assigned default locations when creating the line.

You can manually move arrows **on channels** along the line.

1. Point to the arrow.
2. Press the mouse. Still holding the mouse down, drag it until the arrow reaches the new place.
3. Release the mouse. The arrow and the signal list are redrawn accordingly.

To create a non-delay channel you simply drag the arrow to the starting or ending breakpoint.

Moving a Line and Its Connection Point

A channel or a signal route may be connected to a connection point (usually located close to the line, immediately outside the frame).

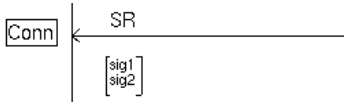


Figure 388: A signal route connected to a connection point.

Moving the line's ending point with the connection point unselected will move the line only and create a new connection point. The old connection point is removed.

To move the line **and** the connection point:

1. Click on the ending point of the line. Make sure the connection point is selected (i.e. the selection rectangle appears).
2. Drag the line's ending point to the new location. The connection point follows the mouse motion as well.
   – If the line's new ending point coincides with a gate symbol and the connection point has a name, a dialog is issued. The meaning of this message is described in "Connecting a Gate Symbol with a Channel or Signal Route" on page 1853.

Moving a Line Text Attribute

A line text attribute is a textual element that is related to a line. The following lines have associated text attributes:

- Gates
- Channels
- Signal routes
- Flow lines after decision, transition option and macro call symbols.

When drawing a line, the SDL Editor positions the text attribute close to the source symbol and originating line, so that it is easy to identify what line the text attribute is related to. You are however free to move the text attribute to any location in the drawing area within the frame.

To move a line text attribute:

1. Point to the border of the text attribute.
2. Press the mouse, and, still holding the mouse down, drag the text attribute to its new location.
3. Release the mouse button.

Redirecting and Bidirecting Lines

When you add a channel or signal route to an interaction diagram, the line is initially drawn in a uni-directional manner, originating from the source symbol and pointing to the target symbol.

The only way to draw a line from the environment to a symbol is to first draw it from the symbol and then redirect it.

Redirecting a Line

1. Select the line
2. Choose *Redirect* from the *Edit* menu. The line is redirected.

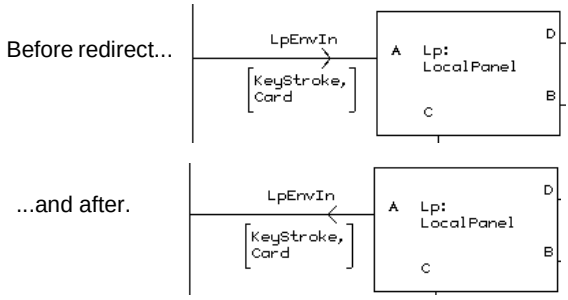


Figure 389: Redirecting a line

Bidirecting a Line

1. Select the line.
2. Choose *Bidirect* from the *Edit* menu. The line is bidirected.
3. Fill in the second signal list.

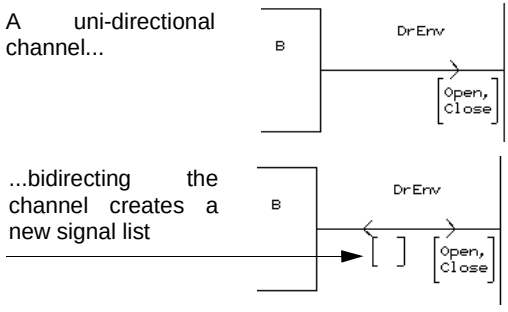


Figure 390: Bidirecting a line

Uni-Directing a Bidirected Line

1. Select the bidirected line.
2. Choose *Unidirect* from the *Edit* menu. A dialog will appear asking what arrow to keep.

Adjusting Lines to the Grid

When you move or reshape interaction lines, the line breakpoints are adjusted to the line grid, which cannot be disabled and has a fixed resolution of 2.5 mm.

Flow line segments and line text attributes are always placed so that they match the resolution of the line grid.

Working with Classes

This section describes how Classes are handled graphically.

The class symbol is a graphical syntax for defining SDL data structures, instead of writing textual newtype definitions. It is allowed in every diagram where textual data definitions are allowed.

When analyzing the SDL specification, class symbols are translated to standard textual newtype definitions, that are used as input to the analyzer.

It is important to notice that, in a diagram, all class symbols with the same class name are treated as if they were a single, merged class. Thus two classes, both named “a_class” but defining two separate operators, would be the same as a single class named “a_class”, defining both operators.

While attributes and operators may be defined and edited in the SDL Editor directly, they may also be edited in the *Browse&Edit Class* dialog, see “Browse & Edit Class Dialog” on page 1877.

The complete definition of all classes may be viewed in the *Browse&Edit Class* dialog, but also, in SDL/PR form, in the *Class information* dialog. The *Class Information* dialog may also be used for traversing among the classes that define the SDL/PR code, see “Class Information” on page 1876.

> **Note:**
>
> Both the *Browse&Edit Class* dialog and the *Class Information* dialog present the aggregate of all graphical class symbols with the same class name.

Class Information

The complete class may be presented in textual (SDL/PR) form. This is done in the read-only *Class information* dialog.

The dialog can be started from the *Windows* menu, and will also be started automatically in response to a *Show symbol* request concerning a class, e.g *Show Error* from the *Organizer Log*. Since the sum of all class symbols with the same name is a single class, some information may be duplicated. The PR-representation of an error is shown in the

class information dialog, and it will be possible to navigate to the symbol(s) that generated the PR-code.

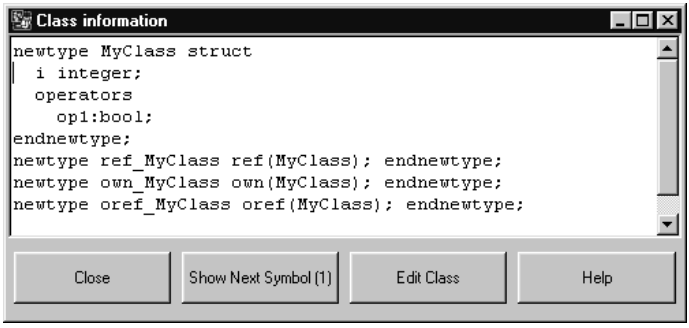


Figure 391 Class Information window

Buttons:

- *Show Next Symbol* shows the next symbol described by the PR code line on which the cursor is placed. The number within brackets shows the total number symbols in the current SDL diagram that that are described by this line. If a line does not refer to any particular symbol, *Show Next Symbol* shows the next symbol belonging to the current class. Each time *Show Next Symbol* is clicked, the next symbol that is described by the current line is shown.
- *Edit Class* will show the *Browse & Edit Class* dialog for the current class.

If the cursor is moved to another line in the text area, the *Show Next Symbol* button is updated.

Browse & Edit Class Dialog

The *Browse & Edit Class* dialog is opened when you select *Class...* from the *Edit* menu. The dialog allows inspection of classes in the SDL diagram.

A class is defined as the union of attributes and operators in the class symbols, and the purpose of the *Browse & Edit Class* dialog is to display this union and ensure consistency when editing this combined information.

The *Browse & Edit Class* dialog is a modal dialog, which is divided into two parts. The browsing functionality is placed at the top part of the dialog, where all classes in the diagram, and all occurrences of each class, are listed in two drop-down menus. Below, in the editing part, the name of the class, its attributes and all its operators are listed.

The *Browse & Edit Class* dialog is available when a single class symbol is selected, unless the class name contains incorrect syntax.

Browse

In the topmost part of the *Browse & Edit Class* dialog, you can browse all class symbols within the current diagram.

Edit

In the lower part of the *Browse & Edit Class* dialog, you can edit the name, attributes and operators of a class. Any changes will propagate into all class symbols of this class within the SDL diagram. The attributes and operators will be presented in a list. This list contains parsed information, derived from the texts in the class symbols. If a particular text contains syntax errors, attributes and operators contained in the same text might be missing from this list.

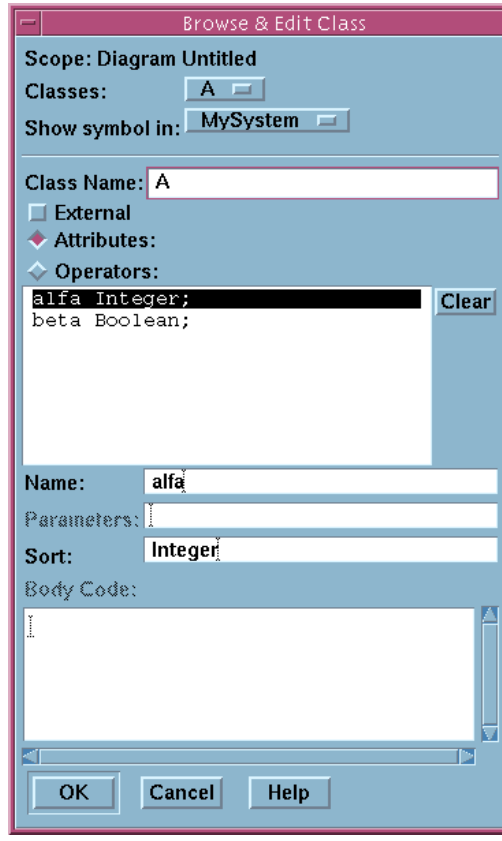


Figure 392: The Browse & Edit Class dialog

The *Scope* Label

The *Scope* label is a non-editable text field, indicating which SDL diagram the *Browse & Edit Class* dialog operates on.

The *Classes* Drop-Down Menu

The *Classes* drop-down menu lists all classes within the current SDL diagram. When a class is selected from this menu, its attributes and operators are displayed in the editing part of the dialog.

The *Show Symbol in* Option Menu

A class can be represented by several class symbols. The *Show Symbol in* drop-down menu lists the name of all pages with occurrences of the current class (the current class is displayed in the *Classes* drop-down menu). By selecting a page name in this menu, the corresponding page will be shown in the work area.

The *Class Name* Field

The *Class Name* field is an editable text field, containing the name of the class being edited.

When the *OK* button is clicked, changes to the *Class Name* field update all occurrences of the class name in all class symbols, within the current SDL diagram.

The *External* Check Box

The *External* check box is checked when a class is externally defined. This means that no PR code will be generated for this class.

When the *OK* button is clicked, changes to the *External* check box update all occurrences of the external flag, in all relevant class symbols using the current class name, within the current SDL diagram.

The *Attributes* and *Operators* Buttons

The *Attributes* and *Operators* buttons are used to select the contents of the attributes/operators list. Clicking Attributes will list all attributes defined for the selected class, while clicking *Operators* will list all operators.

The *Attributes/Operators* List

The *Attributes/Operators* list lists all attributes or all operators defined for the selected class.

The *Clear* Button

The *Clear* button is used to remove the attribute or operator currently selected in the *Attributes/Operators* list.

When the ‘OK’ button is clicked, the selected attribute or operator is removed from all relevant class symbols with the current class name, within the current SDL diagram.

The *Name* Field

The *Name* text field contains the name of the attribute or operator currently selected in the *Attributes/Operators* list, if any.

When the *OK* button is clicked, changes to the definition of the attribute or operator are applied to all relevant class symbols with the current class name, within the current SDL diagram

The *Parameters* Field

The *Parameters* field is an editable text field, containing the parameters of a class operator. It is only editable when an operator is selected in the *Attributes/Operators* list.

When the *OK* button is clicked, changes to the parameter list are applied to the definition of that operator in all relevant class symbols with the current class name, within the current SDL diagram.

The *Sort/Returnsort* Field

The *Sort/Returnsort* field is an editable text field that is editable only when an attribute or an operator is selected in the *Attributes/Operators* list.

Depending on the selection, the field contains:

- the sort of the selected attribute,
- or the return sort of the selected operator.

When the *OK* button is clicked, the sort of the selected attribute, or the return sort of the selected operator, is updated in all relevant class symbols with the current class name, within the current SDL diagram.

The *Body Code* Text

The *Body Code* field is a text field, containing the body code of a class operator. It is only editable when an operator is selected in the *Attributes/Operators* list.

The declaration part of the body code is automatically generated, and all code should be added between the two comments indicating start and

end of the user defined part, respectively. The comments themselves should never be edited. For example, the operator `myOp:Boolean;` will have the following default declaration:

```
operator myOp returns Boolean {
/* Start of user defined body code */
/* End of user defined body code */
}
```

When the *OK* button is clicked, changes to the body code definition are applied to the operator in the class with the current class name, within the current SDL diagram.

The *OK* Button

The *OK* button closes the *Browse & Edit Class* dialog, and updates all appropriate class symbols in the current SDL diagram.

No changes are made in the diagram until the *OK* button is clicked. This makes it possible to specify several changes in the dialog and disregard them later, by clicking the *Cancel* button.

All values are syntactically checked, so it is not possible to add syntax errors to your classes by using the *Browse & Edit Class* dialog. This might make it impossible to delete syntactically incorrect text from class symbols using the *Browse & Edit Class* dialog, since the dialog relies on information obtained by parsing the relevant symbol text compartments. A syntactically incorrect text must then be corrected by editing individual symbols.

Once the *OK* button is clicked, changes cannot be undone.

The *Cancel* Button

The *Cancel* button will close the *Browse & Edit Class* dialog and discard any changes specified in the dialog.

Syntax and Definition of Class Symbols

Name

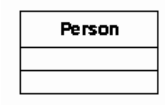


Figure 393 Class symbol

The syntax for the name field is `<SDL_name>`, where `<SDL_name>` is a name in accordance with the SDL lexical rules.

The above class symbol will generate the following newtype definition.

```
newtype Person
endnewtype;
newtype Ref_Person Ref(Person); endnewtype;
newtype Own_Person Own(Person); endnewtype;
newtype Oref_Person Oref(Person); endnewtype;
```

The three pointertypes, Ref_Person, Own_Person and Oref_Person, are generated for use when creating associations and aggregations.

Attributes

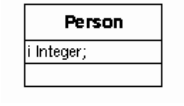


Figure 394 Class symbol with attribute

The syntax for the attribute field is `<name><sort><end>`, where `<name>` and `<sort>` are names in accordance with the SDL lexical rules.

The above class symbol will generate the following newtype definition.

```
newtype Person struct
  i Integer;
endnewtype;
newtype Ref_Person Ref(Person); endnewtype;
newtype Own_Person Own(Person); endnewtype;
newtype Oref_Person Oref(Person); endnewtype;
```

Operators

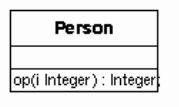


Figure 395 Class symbol with operator

The syntax for the operator field is
`<name>['('<par>{','<par>*')'][<returns><sort>]<end>`

where `<par>` is defined as

`[<kind>]<name>[<colon>]<sort>`

and `<kind>` is defined as

`'in'|'in/out'`

The above class symbol will generate the following newtype definition.

```
newtype Person
  operators
    op: Integer -> Integer;
endnewtype;
newtype Ref_Person Ref(Person); endnewtype;
newtype Own_Person Own(Person); endnewtype;
newtype Oref_Person Oref(Person); endnewtype;
```

Syntax and Definition of Association Lines

Figure 396 Association line

The syntax for the role name of an association is `[<name>][<constraint>]`, where `<constraint>` is in the format `'{'name'}'`.

Associations can be of a number of types, depending on the constraint:

- if there is no constraint, the type is `Ref_<opposite_class_name>`
- if the constraint is “own”, the type is `Own_<opposite_class_name>`

- if the constraint is “oref”, the type is `Oref_<opposite_class_name>`
- if there is a constraint, but not “own” or “oref”, the type is `<constraint_name>`.

The above association will generate the following newtype definition.

```
newtype Company struct
  e Ref_Employee;
endnewtype;
newtype Ref_Company Ref(Company); endnewtype;
newtype Own_Company Own(Company); endnewtype;
newtype Oref_Company Oref(Company); endnewtype;
newtype Employee
endnewtype;
newtype Ref_Employee Ref(Employee); endnewtype;
newtype Own_Employee Own(Employee); endnewtype;
newtype Oref_Employee Oref(Employee); endnewtype;
```

Syntax and Definition of Aggregation Lines

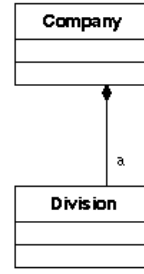


Figure 397 Aggregation line

The above aggregation will generate the following PR code.

```
newtype Company
  a Ref_Division;
endnewtype;
newtype Ref_Company Ref(Company); endnewtype;
newtype Own_Company Own(Company); endnewtype;
newtype Oref_Company Oref(Company); endnewtype;
newtype Division
endnewtype;
newtype Ref_Division Ref(Division); endnewtype;
newtype Own_Division Own(Division); endnewtype;
newtype Oref_Division Oref(Division); endnewtype;
```

Working with Pages

This section describes the more important aspects of page handling.

Each diagram page must have a unique name for identification purposes. This name must be correct in accordance with the naming convention for pages in SDL. Each diagram must contain at least one page, and there is no maximum amount of pages that a diagram may contain.

The *autonumbering* facility available for pages relates to automatically updating the sequential numbering when pages are added or deleted.

When diagrams are opened, the default way of presentation is that the first page of the requested diagram is shown (first in accordance to the order in which pages have been added). This default can be overruled, and you can go directly to a specific page that has been predetermined.

Most of the page managing functions are available through the menu choices on the *Pages* menu.

A number of functions are available through the *Edit* menu choice in this menu:

- *Ordering Pages*
- *Naming Pages*
- *Adding a Page*
- *Renaming a Page*
- *Clearing (Deleting) a Page*
- *Pasting a Page*
- *Transferring to a Page*
- *Printing a Page*
- *Resizing a Page*

Other page functions are also available in the SDL Editor:

- *Applying Autonumbering on Page Names*
- *Removing Autonumbering on Page Names*
- *Designating the Page to Open*
- *Transferring to a Specific Page*
- *Transferring to the Referring Page*
- *Transferring to the Next or the Previous Page*
- *Transferring to the First Page*
- *Transferring to the Last Page*

Ordering Pages

Within an SDL diagram, page locations are set according to the order in which they are added (see "Adding a Page" on page 1888). This order is reflected in:

- The Organizer structure
- The listing order for pages in the *Edit Pages* dialog

To re-order pages, you use the *Cut* and *Paste* button.

Naming Pages

Whenever you name a page, the names that you use must strictly adhere to SDL naming conventions. If you use unacceptable notation (e.g. blank spaces or a semi-colon), you receive a message.

Applying Autonumbering on Page Names

If autonumbering is required, a feature can assign numeric names to the pages in the diagram. The page names will be assigned 1, 2, 3 and so forth.

1. Select *Edit*.
2. Select the page to autonumber in the page list.
3. Toggle the *Autonumbered* toggle button to on in the *Edit Pages* dialog. You are then prompted if you want to autonumber the selected page only or all pages in the dialog which is issued.

   – If you click *Yes*, the SDL Editor assigns the page name the highest assigned numeric value + 1. You are then returned to the *Edit Pages* dialog.

   – If you click *All Pages*, all pages will be assigned numeric names, starting with 1. The numbering will follow the listing order in the page list. You are then returned to the *Edit Pages* dialog.

Removing Autonumbering on Page Names

The autonumbering feature can be turned off. This must for instance be done if you want to assign a specific name to a page.

1. Select *Edit*.
2. In the page list, select the page to remove autonumbering on.
3. Toggle the *Autonumbered* toggle button to off in the *Edit Pages* dialog. You are then prompted to confirm the operation in the dialog which is issued.
   - Clicking *Yes* transfers you to a *Rename Page* dialog where you assign the page a new name. Autonumbering is removed.

Adding a Page

To add a page to an existing diagram:

1. Select *Edit*.
2. In the list of existing pages, select the page to precede or succeed the new page to be added.
3. Click the *Add* button. The *Add Page* dialog is issued.

A faster way to add a page before or after the current page is to select the *Add* menu choice directly. In this case, the *Add Page* dialog is issued directly.

In the *Add Page* dialog:

1. Enter the required page name in keeping with SDL conventions (the name must be unique within the diagram). Otherwise, select *Autonumbered* if you want the pages to be automatically renumbered to incorporate the new page.
2. Select the position of the new page – before or after the current page.
3. Select the type of page required. The dialog box automatically shows the page options that are available to you under SDL rules.
4. Click *OK*. Control is returned to the *Edit Pages* dialog.
5. Click *Done*.

Designating the Page to Open

When a diagram is opened without specifying a particular page, the default is that it is opened at the first page that has been added to the diagram, showing the upper left part of the page.

You can however open a diagram at a specific page in the SDL Editor.

To specify the page to be opened first:

1. Bring up the *Edit Pages* dialog.

   The *Open this page first* field, below the toggle button, reflects the page to be opened first (page 1 in Figure 398).

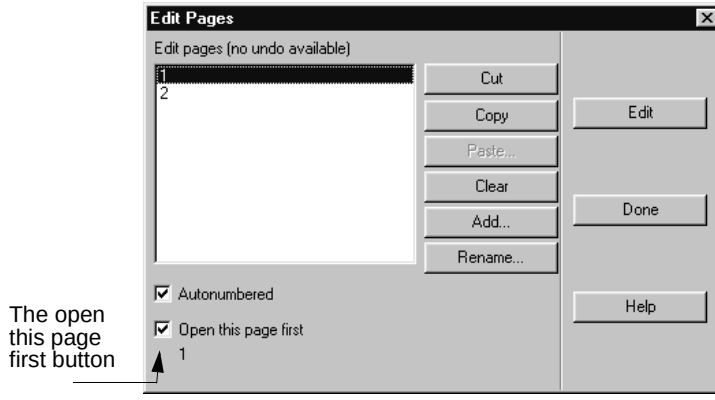


Figure 398: Specifying what page to open first

2. To specify another page, select the page to be opened first in the *Edit pages* dialog.
3. Turn the *Open this page first* button on. The identification of the page to be opened is then placed under the *Open this page first* button.
4. Click *Done*.

Renaming a Page

1. Select *Edit.*
2. Select the page to rename.
   - To rename an auto-numbered page, first click the *Autonumbered* button off. Autonumbering is removed on the selected page and the *Rename Page* dialog is issued. Go to step 4.
3. Click the *Rename* button. The *Rename Page* dialog is issued.
4. Fill in the required new name in keeping with SDL naming conventions. Page names must be unique within the diagram. For block and process diagrams, an option is available to also change the page type.
5. Click *OK*. Control is returned to the *Edit Pages* dialog.
6. Click *Done*.

Clearing (Deleting) a Page

If there is only page in a diagram, you cannot remove it. A message is then shown.

Caution!

Clear pages with caution as there is no *Undo* option available.

1. Select *Edit*.
2. Select the page to be deleted in the page list.
3. Click on the *Clear* option and then on *Clear* to delete the page.
4. Click *Done*.

Pasting a Page

1. Open the SDL diagram where to paste the page.
2. Select the *Edit* menu choice.

3. In the list of existing pages, select the page to precede or succeed the new page to be pasted. Click the *Paste* button. The *Paste Page* dialog is issued.
4. Enter the required page name in keeping with SDL conventions (the name must be unique within the diagram). Otherwise, select *Autonumbered* if you want the pages to be automatically renumbered to incorporate the page to be pasted.
5. Select where to paste the page - before or after the current page.
6. Click *Paste*. Control is returned to the *Edit Pages* dialog.

   If there are reference symbols on the pasted page all the underlying substructure is also pasted. See "Cutting, Copying and Pasting Reference Symbols" on page 1851.
7. Click *Done*.

Transferring to a Page

There are several ways to transfer to another page of the current diagram. When you transfer to a page, it will be placed on the top of the stack in the window if you are using one window only. Otherwise, that page will be displayed in a window of its own.

Transferring to a Specific Page

- Transfer to a page, located not further than two pages from the current page, by selecting that page in the *Pages* menu.
- In the *Edit Pages* dialog, transfer to a page by clicking on the required page and then click on the *Edit* button.
  - Alternatively, double-click the page icon in the Organizer.

Transferring to the Referring Page

To transfer to the page where the current page is referenced from (i.e. where the SDL reference symbol referring to the current diagram is located):

- Select *Edit Reference Page* from the *Pages* menu or the *Reference Page* quick-button.

Transferring to the Next or the Previous Page

You may transfer to the next or previous page in the diagram. The order is specified according to the listing order in the *Edit Pages* dialog.

To transfer to the next or previous page:

- Use the quick-buttons for *Previous Page* or *Next Page*.

Transferring to the First Page

Select *First* from the *Pages* menu. You are transferred to the first page of the diagram.

Transferring to the Last Page

Select *Last* from the *Pages* menu. You are transferred to the last page of the diagram.

Printing a Page

The SDL suite allows to print individual pages from the SDL Editor. You can also restrict the scope of printing to a part of the page. A number of options which affect the resulting printout are possible to specify.

Resizing a Page

1. Transfer to the page to resize.
2. Select *Drawing Size* from the *Edit* menu. A dialog is issued, showing the current width and height.
3. Type in the new values. Click *OK*.

> **Note:**
>
> If you enlarge a page so that the page size becomes wider or larger than the physical page size defined in the *Print Options*, your printouts will require more pages than before, if you use a fixed printout scale.

Working with Windows

Each SDL Editor window shows one SDL page. The SDL Editor is a multi-window tool, allowing you to open new windows on a page and close windows that are no longer needed.

A newly opened window is a mirror image of the window from which the selection to open a window was made. Multiple windows may be opened (one at a time) and they are all identical with the first one. Any subsequent modifications made will reflect the same in either the original window, or any of the newly opened window(s). This option affords the possibility of viewing in more than one window, and can ideally be used in conjunction with the scaling factor when looking at a detailed page.

For information on how you open:

- The Grammar Help Window, see "Opening the Grammar Help Window" on page 1906.
- The Signal Dictionary Window, see "Opening the Signal Dictionary Window" on page 1918.

Opening and Closing Windows

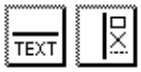
Standardized functions for opening and closing windows are provided in the *Window* menu: *New Window* and *Close Window*. If you close the last window opened by the SDL Editor, the editor will exit.

Hiding and Showing Parts of the SDL Editor Window

You can show and hide the various component parts of the SDL Editor window. **On UNIX**, if they are hidden, it increases the drawing area available to be used in the creation or modification of pages of a diagram.

You find all the available options for hiding and showing parts of the window in the dialog box accessed via the *Window Options* command on the *View* menu. You can set all of these options as preferences.

All of these options are turned on and off with a toggle button. Some of these options are available with quick-buttons:



- The quick-button for *text window* on / off or *symbol menu* on / off.

  – If you click *All Windows* it applies the values to all instances of the SDL Editor windows.

Editing Text

General

There are several ways of editing text in the SDL Editor, i.e.

- In the drawing area
- In the text window
- In an external text editor

Two support functions are provided, namely:

- Grammar Help, see "Using Grammar Help" on page 1906.
- Signal Dictionary, see "Using the Signal Dictionary" on page 1917.

Editing in the Drawing Area and Text Window

The SDL Editor allows you to edit textual objects directly in the drawing area.

If you select a text attribute to an object in the SDL Editor window, a text cursor is inserted directly in the text associated with the object, and the *text window* will be updated to contain the text. This is explained in "Selecting a Symbol That Has Associated Objects" on page 1837 and "Selecting a Line That Has Associated Objects" on page 1866.

When you type text in the drawing area, all accelerator keys are interpreted as input to the text. As an example, the `<Delete>` key will in text editing mode delete a character, but in non-text editing mode it will remove (*Clear*) an entire symbol. This means that you can only use text editing keyboard accelerators like `<Home>`, `<End>`, `<Ctrl+A>` and `<Ctrl+E>` in text editing mode.

When editing text directly in the drawing area it is possible to select text by dragging the mouse over the text or select words by double-clicking.

You can extend or reduced the selection by clicking while pressing the <shift> key.

To edit large texts it might be more convenient to use the *Text Window* to edit the text. If you select more than one text object, the text window is not updated. Each line (except for the last line) in the text window is terminated by a carriage return, and may consist of any number of legal characters.

Regardless of whether you edit directly in the drawing area or in the text window, the SDL Editor makes sure that the contents of both displayed texts are consistent.

A context sensitive syntax check is performed on all texts being edited. If a syntax error is found a red bar underlining the text will appear where the first error occurs in the text. See also "Syntax Rules when Editing" on page 1786.

You can also edit text in an external text editor by using the *Connect to Text Editor* command in the *Tools* menu. This allows you to use the Text Editor, the Emacs text editor **(on UNIX)** or MS Word **(in Windows)** to edit large pieces of text.

The SDL Editor also supports editing text outside its context, by transferring text to / from a file.

Text Window

On UNIX, the text window is a pane of the SDL Editor window.

In Windows, the text window is a floating, resizeable and moveable window that can be placed anywhere on the screen. A single text window is shared by all instances of the SDL Editor currently running.

The text window contains two menus described in:

- "The File Menu of the Text Window" on page 1962
- "Edit Menu" on page 1938

Hiding and Showing the Text Window

You can hide or show the text window with the *Window Options* command from the *View* menu, or by using a quick-button.

In Windows: When visible, the text window will always be placed on top of the SDL Editor window.

Basic Text Editing Functions

Delayed Updating of the Drawing Area

When you edit text in the text window, both the text window and corresponding text in the drawing area will be updated. The the drawing area will be updated after a slight time delay and this can be set by an editor preference.

If you try to type an illegal character, you will be warned by a beep.

Standardized Text Editing Functions

When you edit text, you can:

- Position the cursor by clicking or by using the arrow keys.
- Insert characters after the current position of the text pointer.
- Delete one character backward by pressing `<backspace>`.
- Delete one character forward by pressing `<delete>`.
- Replace selected text by typing.
- Delete selected text by pressing `<backspace>` or `<delete>`.
- Select text by dragging or by clicking and `<Shift>`-clicking.
- Select a word by double-clicking it.
- Select a line by triple-clicking it.

Searching and Replacing Text

In the SDL editor, you can search for and replace text in a page or in a diagram. In the Organizer, you can search for and replace text in a diagram or in a diagram structure that is managed by the Organizer. In both the SDL editor and the Organizer, there are two search variants:

- Dialog search
- Fast search

Searching for Text in a Diagram using Dialog Search

To search for text in the current page from the SDL editor:

1. Select *Search* from the SDL editor's *Tools* menu or click the SDL editor quick-button for *Search*. The *Search dialog* is issued.

2. Specify the text to be found in the *Search for* text field.
   - Click the *Consider case* button on to search case sensitive.
   - Click the *Wildcard search* button on to use wildcard matching, where an asterisk ('*') matches a sequence of zero or more characters of any kind.
   - Turn the *Search all pages* button on to specify that the scope of search should comprise all pages in the diagram. If you turn the button off, the scope of search will only comprise the current page.
3. Click the *Search* button. The object where the next occurrence of text is found will be selected and the text window updated accordingly.
   - The diagram that the tool is currently processing is displayed to the right of the *Search In* field.
4. Click *Close*.

Searching and Replacing Text using Dialog Search

1. Open the Search dialog as described in "Searching for Text in a Diagram using Dialog Search" on page 1896.
2. Specify the text to be found in the *Search for* text field.
3. Specify the text to be replaced with in the *Replace with* text field.
4. Click the *Search* button to find the first occurrence.

5. If the text is found, the object where the next occurrence of text is found will be selected and the text window updated accordingly. You may use any of the following options:
   - Search further by clicking *Search* repeatedly
   - Replace the text by clicking *Replace*
   - Replace and continue search with the *Replace & Search* button.
   - Replace all occurrences with the *Replace All* button.
6. Close the Search dialog by clicking *Close*.

Fast Search

Fast search is like dialog search but without the dialog. The setting from the last dialog search is used for *Wildcard search*. Fast search do never consider case, and does always search all objects and all pages. To invoke Fast search, type ctrl+F when the SDL editor is not in text editing mode. If you are in text editing mode, you can click in the diagram background first to get out of text editing mode.

When Fast search is invoked, the message area displays the text that will be searched for. Initially, the text is *Search for:*. When you type on the keyboard with the mouse pointer over the drawing area, the characters will turn up in the message area. When you have typed the text pattern to search for, press enter or return to start the search operation.

The same search operation as for Dialog search is used. If the search operation finds a match, you can search for another match by pressing enter or return once more.

To finish the Fast search operation, click in the drawing area or select another operation.

The next time Fast search is invoked with ctrl+F, the search text that was used the last time is proposed as a search text once more. To use it, press enter or return. To use another search text, press ctrl+F once more or the delete key several times, to erase the search text.

Editing Text by Using an External Text Editor

Whenever a text object is selected and the text is visible in the text window, the command *Connect to Text Editor* in the *Tools* menu is avail-

able. This command opens an external text editor containing the text of the selected object. Which external text editor to start is defined by the preference SDT*TextEditor, which can be set either to the Text Editor or to the Emacs text editor **(on UNIX)**.

From this point on, you can only edit the text by using the external text editor.

The text in the SDL Editor is updated every time the external text editor saves the text. When the text is no longer edited in the external text editor, the editing control returns to the SDL Editor.

Importing / Exporting Text

The SDL suite allows you to import and export text from / to a file. You can, for instance, import text from files that contain SDL/PR into text symbols and export the contents of text symbols to files.

Importing Text from a File

Caution!

Importing text from a file replaces existing text as well. Use the *Undo* facility to revert to the original text, as long as the symbol is selected – deselecting the symbol will disable *Undo*.

You can import the contents of a text file into an object managed by the SDL Editor in the following way:

1. Select the object where you want to import the contents of a file.
2. Select *Import* from the text window's *File* menu.
3. Specify the file to import.
4. Click *OK*. This **replaces** the contents of the text window with the contents of the file you have specified in the file selection dialog.

Exporting Text from a File

To export the contents of the text window into a file:

1. Select the object which text you want to export to a file.
2. Select *Export* from the text window's *File* menu.

3. Specify the file to export the text to.
4. Clicking *OK* possibly creates a new file and replaces the contents of the file with the contents of the text window.

Copying, Cutting and Pasting Text

In the text window, you can cut, copy and paste text between different symbols, lines and text attributes. You can also transfer text between different applications.

Programmable Function Keys (UNIX only)

On UNIX, the SDL suite allows to tie a function key to a defined text string. When you type that defined function key, the programmed text string will be inserted at the current cursor location. You can customize your own programming of function keys.

Global SDL Suite Resources

The function keys are set up as X resources. It is possible to set up both system default and user-defined X resources, allowing you to customize your environment. The X resources are defined in a file that is common for all SDL suite users, namely

```
/usr/lib/X11/app-defaults/SDT
```

To program the function keys, insert following lines anywhere into the `SDT` file:

```
/* Any suitable comment */
SDT*XmText.translations: #override \n\
<Key>F1: insert-string("F1Text") \n\
<Key>F2: insert-string("F2Text") \n\
<Key>F3: insert-string("F3Text") \n\
<Key>F4: insert-string("F4Text") \n\
<Key>F5: insert-string("F5Text") \n\
<Key>F6: insert-string("F6Text") \n\
<Key>F7: insert-string("F7Text") \n\
<Key>F8: insert-string("F8Text") \n\
<Key>F9: insert-string("F9Text")
/* Note the absence of \n\ on line 9 */
```

Note:

Omitting to define some of the function keys is permissible.

User-Defined Resources

You can define your own function keys. You do this by defining the X resources described above in a personal copy of the definition file and to store that file it into your home directory:

```
~username/SDT
```

Alternatively, any directory designated by the environment variable `XAPPLRESDIR` can be used.

Restrictions

- You can only define one line for each function key. If you attempt to define more than one line into one function key may cause an unpredictable result when you press that key.

  For instance, it is not certain that the following line will produce the expected result:

  ```
  <Key>F1: insert-string("F1Line1\nLine2") \n\
  ```

- You can define the keys F1–F9.

Changing Fonts on Text Objects

You may change the font faces and font sizes used in the textual objects displayed by the SDL Editor. All textual objects use the same font faces and font sizes, meaning that you can neither change them individually nor change them during an SDL Editor session.

There is one exception though – it is possible to use a separate font for text symbols. This is useful in process diagrams where you may want a proportional font in most symbols to save space, while you may want a non-proportional font in text symbols to be able to align words on different lines.

The font faces available depend on the target system on which you are running the SDL suite.

Defining What Font to Use

To modify the desired font size and font face, you must use the Preference Manager. See chapter 4, *Managing Preferences*.

Textual Objects Preferences

When the setting is in effect, the SDL suite will use the font face names given by the preference settings

Editor*FontText*ScreenFontFamily
Editor*FontText*PrintFontFamily

to select font face names. Note that in this way you can select different font names for screen and for print.

On UNIX, if you leave the Editor*FontText*ScreenFontFamily preference setting empty, you will edit your documents using the *SDT Draft* font, but print them using the font you specified with the Editor*FontText*PrintFontFamily setting.

Text Symbol Preference

For text symbols, the preference Editor*FontText*TextSymbolFontFamily is used both for on-screen viewing and printing.

Supported Font Faces

On UNIX, the availability of font faces is determined by the version of the X Windows server which is running. With a revision 5 or higher (X11 R5), scalable fonts are supported. In that case, the available list of predefined font faces would be:

- Times
- Helvetica
- Courier
- SDT Draft (mapped to the Schumacher font)
- Other (mapped to a user-defined font)

In Windows, the availability of font faces is determined by the TrueType fonts that are currently installed on the computer (use for instance the Windows Control Panel to determine what is available).

Default Font Face

The default font face is Helvetica. **On UNIX**, if scalable fonts are not supported, the font face will be replaced by a *Schumacher* font (which can be used in all circumstances).

Default Font Sizes

The default font sizes are as follows:

Page Font Sizes (except overview pages)	
Text Object:	**Font Size**
Kernel heading, texts in reference symbols, channel names, signal route names, gate names	12 points
Other text objects	9 points

Overview Page Font Sizes	
Text Object:	**Font Size:**
All texts except signal lists	10 points
Signal list symbol	8 points

Setting Alignment

Normally the alignment of the text inside symbols are automatically set. However, you can use the preference setting Editor*Font-Text*TaskSymbolLeftAligned to set the alignment for the task symbol, procedure call symbol, macro call symbol, create request symbol and the save symbol. The default value is “off”, meaning that the text will be centered, but if you set it to “on” the text in these symbols will be left aligned.

Determining Which Scalable Fonts Your Server Can Access (UNIX only)

On UNIX, use the `xlsfonts` command to list installed fonts. Font names containing 0 for width and height are scalable.

How to Determine what Fonts are Available

From the OS prompt, typing:

```
hostname% xlsfonts | grep "\-0\-0\-" | more
```

will return a list of accessible scalable fonts.

Scalable Fonts Under R5 Servers

To use scalable fonts under X11R5 you must normally first connect to a font server.

How to Start the Font Server

1. Start the font server on any local host:

   ```
   hostname% fs
   ```

2. Connect the server to `fs` indicating which host the font server is running on (which can be the same host that the X server is running on):

   ```
   hostname2% xset +fp tcp/<hostname>:7000
   ```

For further information see the X11R5 documentation or use `man fs` to read the manual page describing the font server you are running.

Disabling Font Scaling (UNIX only)

On UNIX, if the fonts look poor on the screen, a possible work-around is to disable the scaling option.

> **Note:**
> Disabling font scaling effectively disables WYSIWYG!

To do this, you should edit the `SDT` resource file.

1. Open the file `SDT` in a text editor.
2. Locate the line with the text: `SDT*sdtUseScalableFonts`
3. Change the line to `SDT*sdtUseScalableFonts: false`
4. Save the file and restart the SDL suite environment.

Grammar Help and Signal Dictionary

General

Besides the context sensitive syntax check performed on the texts, there is a *grammar help* support function. It assists you in entering the correct text, according to the SDL grammar, in the selected text attribute to an object.

Designing using SDL implies to large extent defining, sending and receiving signals. A *signal dictionary* assists you in reducing the time it takes to find out names and parameters for a signal that you already have used somewhere in a diagram. The signal dictionary also incorporates timers.

Keyboard Accelerators (UNIX only)

In addition to the standard keyboard accelerators, described in "Keyboard Accelerators" on page 35 in chapter 1, *User Interface and Basic Operations*, the Grammar Help and Signal Dictionary window features the following:

Accelerator	Command / functionality
`g`	Select the *Grammar* section if the option is enabled and bring the separator into view
`u`	Similar to `g` but applies on *Up*.
`t`	Similar to `g` but applies on *This*.
`d`	Similar to `g` but applies on *Down*.
`a`	Similar to `g` but applies on *All*.
`m`	Similar to `g` but applies on *MSC*.
`e`	Similar to `g` but applies on *External*.
`Ctrl+g`	Toggles the *Grammar* option on / off (see "Grammar" on page 1965). When the option is enabled, brings the Grammar separator into view and selects it.
`Ctrl+u`	Same as `Ctrl+g`, but applies on the *Up* option.

Accelerator	Command / functionality
Ctrl+t	Same as Ctrl+g, but applies on the *This* option).
Ctrl+d	Same as Ctrl+g, but applies on the *Down* option.
Ctrl+a	Same as Ctrl+g, but applies on the *All* option.
Ctrl+m	Same as Ctrl+g, but applies on the *MSC* option.
Ctrl+e	Same as Ctrl+g, but applies on the *External* option.
Ctrl+f	Finds the **last** occurrence of the first word in the currently selected object in the SDL Editor drawing area and selects it in the *This* section. The first word is defined as the text string from the start of the object's text and until one of the following characters: ' ' (space) ',' (comma) '(' (left parenthesis) <TAB> '\n' (newline)

Using Grammar Help

The Grammar Help window is a multi-functionality window; it can also provide signal dictionary capabilities. These functions are further described in "Using the Signal Dictionary" on page 1917. What functionality is provided depends on the options defined in the *Select* menu.

Each SDL Editor window has its associated Grammar Help window.

Opening the Grammar Help Window

- Select the *Grammar Help* menu choice from the *Windows* menu.

  The *grammar help window* is issued (see Figure 399 on page 1908).

Requesting Grammar Help

If you select the object you need assistance on, you will see the keywords and options available for use in given situations, and the corresponding reference to sections of the ITU Z.100 SDL Definition, followed by the grammar syntax of the meta language.

To request grammar help:

- Select the object of interest. The grammar help window contents are automatically updated to list the following:
  - The nature of the selected object
  - The grammar for that object
  - The various “use cases”.

Inserting Text

This operation inserts the contents of the grammar help window into the SDL Editor text window.

To insert the text related to a given “use case”:

1. Locate the “use case” of interest. The left part of the window provides a list of situations, named using some abbreviation that associates to the situation.
   - The first item(s) in the list reads *GRAMMAR*, possibly with a suffix that informs you about the grammar context. You can take advantage of the GRAMMAR items either as using the references to Z.100 if you need more detailed information about the selected sort of object, or if you need to read more about the concrete grammar.

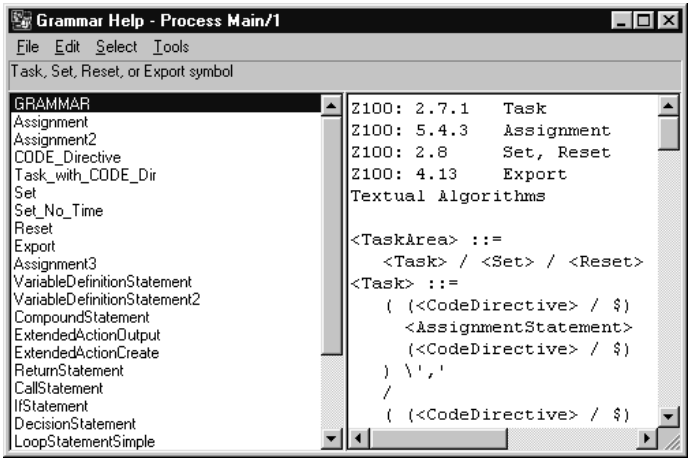


Figure 399: The grammar for a task symbol

– The next items contains a number of “use cases”. As an example, if you select a task symbol, the first “use case” reads *Assignment*, while the second reads *Assignment2*. This is to be interpreted as:
An assignment statement,
Multiple assignment statements in a task symbol.

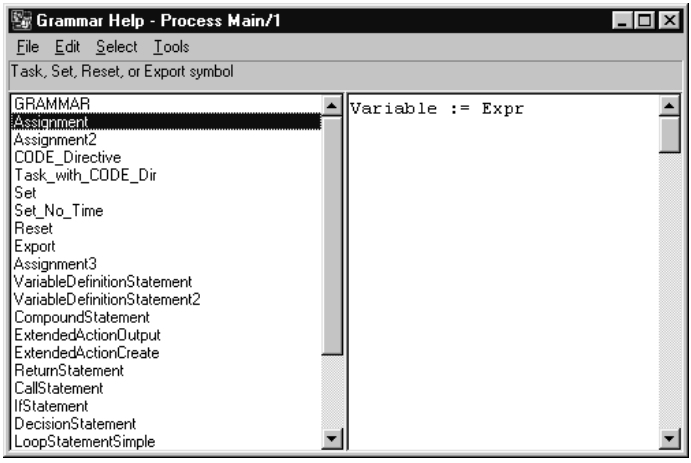


Figure 400: The Assignment "use case"

2. Click on the item of interest to select it.
3. Select *Insert* from the *Edit* menu or double-click the item in the list. The contents of the right part of the window are inserted into the SDL Editor text window, at the current I-beam cursor position.
4. Finally, replace the formal notation with the appropriate values, variables, signals, etc. used in your diagram.

Replacing Text

This operation signifies replacing the contents of the SDL Editor text window with the contents of the grammar help window.

- Proceed as described in "Inserting Text" on page 1907, but select *Replace* from the *Edit* menu.

Customizing the Grammar Help

With the SDL suite, a standard grammar help template file is provided. In addition to this, the SDL Editor allows you to use your own templates and to merge these with the standard templates.

To create grammar help definitions:

1. Copy an existing grammar help file.
2. Edit the file using any text editor (the grammar help file is an ASCII file). See "The Grammar Help File" on page 1912 for a description of the contents and syntax of the file.

To use another grammar template definitions:

- Select *Load* from the *File* menu. In a File Selection Dialog you then specify what template definitions to load.

To merge the current grammar template definitions with another one:

- Select *Load* from the *File* menu. In a File Selection Dialog, you then specify what template definitions to merge with the current.

The *Symbol* Label

The *symbol label* field is a non-editable text field that displays the type of symbol that is currently selected in the SDL Editor.

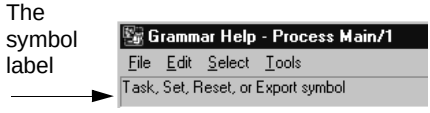


Figure 401: The symbol label

The symbol label reads "No single symbol selected" in the following circumstances:

- No symbol or line is currently selected in the SDL Editor
- More than one symbol is selected
- The symbol type is not defined in the *Grammar Help* file.

The *Name* Field

The *name field* consists of a scrollable list that contains a list of names of templates associated with the currently selected symbol.

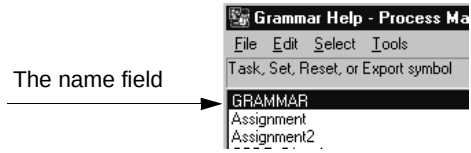


Figure 402: The name field

The list is updated automatically each time an object in the SDL Editor's drawing area is selected, to reflect the templates that are currently available for that object. By default, the first item in the list is selected.

The name field is empty in the following circumstances:

- The symbol label reads "No single symbol selected"
- The symbol type information is missing in the *Grammar Help* file.

When selecting an item in the list, the corresponding template definition appears in the grammar field, allowing to check its contents before inserting it into the text window (see "Insert" on page 1964 and "Replace" on page 1964).

The *Grammar* Field

The *grammar field* is a scrollable field from which text can be copied. Its contents are updated to reflect the definition for the currently selected item in the name field.

The grammar field is empty if no item is selected in the name field (or if the name field is empty).

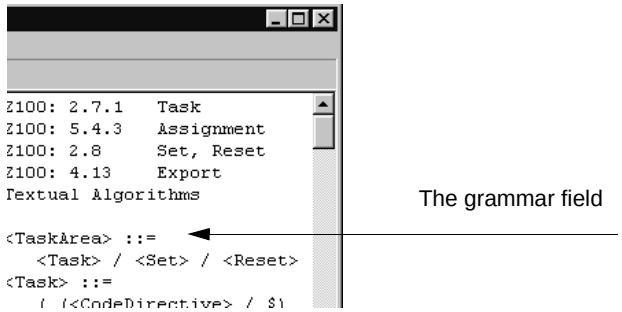


Figure 403: The grammar field

To copy and paste text from the template definition field to the text window, you can:

- Use the *Insert* command, described in "Insert" on page 1964.
- Use the *Replace* command, described in "Replace" on page 1964.
- Copy and paste by using the graphical environment clipboard functions.

The *Grammar Help File*

This section is a reference for the format of the files that contain SDL Grammar Help definitions.

The template definition file format is line-oriented, and uses two separator characters chosen so as to not interfere with the possible contents of templates appearing in the file.

> **Note:**
>
> The characters `'@'` and `'$'` are separator characters. They are **reserved for that purpose.**

The template definition file is divided into **sections** defining templates for a particular symbol type. A section begins with a *Grammar Help Declaration* part and ends with a *Grammar Help Definition* part.

The **grammar help declaration** of a new section has the following syntax:

```
@<editortype>@<symboltype>[@[<symbolcomment>]]
```

<editortype>, <symboltype> and <symbolcomment> are to be replaced with the adequate text strings.

Comments in Grammar Help files are also permissible.

Example 305: The Declaration of a Channel Template ––––––––––––

```
@SDL@CHANNEL@Channel symbol
```

The declaration is to be interpreted as:

- `SDL`
  Valid for the SDL Editor only
- `CHANNEL`
  A channel line
- `Channel Symbol`
  An informative text that will be displayed to you (see "The Symbol Label" on page 1910).

––

editortype

editortype may currently be of one of the keywords:

- - The section contains templates relevant for the SDL Editor.
- - The section contains templates for the Message Sequence Chart Editor.

Note:

Grammar Help is currently not supported by the MSC Editor. The MSC keyword is reserved for future functionality extensions.

symboltype

symboltype is defined per editor and the allowed names are the following:

```
ADDITIONAL_HEADING_BLOCK
ADDITIONAL_HEADING_BLOCK_TYPE
ADDITIONAL_HEADING_PROCEDURE
ADDITIONAL_HEADING_PROCESS
ADDITIONAL_HEADING_PROCESS_TYPE
ADDITIONAL_HEADING_P_AS_SERVICE
ADDITIONAL_HEADING_SERVICE
ADDITIONAL_HEADING_SERVICE_TYPE
ADDITIONAL_HEADING_SYSTEM
ADDITIONAL_HEADING_SYSTEM_TYPE
BLOCKSUBSTRUCTURE
BLOCK_REF
BLOCK_TYPE_REF
CALL
CHANNEL
CHANNELSUBSTRUCTURE
CONNECTION_POINT
CONNECTOR
CONTINUOUS_SIGNAL
CREATE
DECISION
ENABLING_CONDITION
GATE
INPUT
KERNEL_HEADING
KERNEL_HEADING_BLOCK
KERNEL_HEADING_BLOCK_TYPE
KERNEL_HEADING_PROCEDURE
KERNEL_HEADING_PROCESS
KERNEL_HEADING_PROCESS_TYPE
KERNEL_HEADING_SERVICE
KERNEL_HEADING_SERVICE_TYPE
KERNEL_HEADING_SYSTEM
KERNEL_HEADING_SYSTEM_TYPE
MACROCALL
OPERATOR_REF
OUTPUT
CLASS
PACKAGE_REF
PAGENUMBER
PRIORITY_INPUT
PRIORITY_OUTPUT
PROCEDURE_REF
PROCEDURE_START
PROCESS_REF
PROCESS_TYPE_REF
RETURN
SAVE
SERVICE_REF
SERVICE_TYPE_REF
```

```
SIGNALROUTE
START
STATE
STOP
TASK
TEXTSYMBOL_BLOCK_SUBSTR
TEXTSYMBOL_MACRO
TEXTSYMBOL_PROCEDURE
TEXTSYMBOL_PROCESS
TEXTSYMBOL_P_AS_SERVICE
TEXTSYMBOL_SERVICE
TEXTSYMBOL_SYSTEM
TRANSITION_OPTION
```

Note:

If the same section appears more than once, all template definitions under those sections will be available.

symbolcomment

symbolcomment is an optional feature, which allows specification of a text to be associated with each *symboltype*. The text for a specific *symboltype* appears in the symbol type field of the Grammar Help window whenever a symbol of that type is selected in the editor.

Comments

Also, comments may appear in the file using the following syntax:

```
@COMMENT@<commenttext>
```

Note:

Such a comment signals the end of the previous section and should only be used before another section.

Grammar Help Definition

A SDL grammar template has significance only if it is located after a valid declaration. The template will be added to the list of templates for that section.

A template definition is started with a line beginning with a '$' sign and continues until either:

- A line beginning with a '$' sign is read.
- A line beginning with a '@' sign is read.
- End of file is encountered.

The **syntax** of a template definition is simply:

```
$<templatename>'Newline'
<multiple lines constituting the template
definition>
```

Example 306: The definition of a channel example ––––––––––––––––

```
$GRAMMAR
Z100: 2.5.1 (p.45)
Z100: 2.5.5 (p.50) Signal list

<ChannelName> ::= <Name>
<SignalList> ::=
  ( <SignalName> / '(' <SignalListName> ')' )",'
$SignalList
SignalName, SignalName
$SignalList2
SignalName, SignalName, (SignalListName)
```

––

Note:

Neither leading nor trailing newlines will be added to the template definition. These can be added to the template simply by adding a leading empty line and/or a trailing empty line in the template definition.

File Handling

When a Grammar Help window is opened, the SDL Editor will try to locate a SDL grammar help file and to load it, if found. The search order is as follows:

1. The SDL Editor starts by fetching the preference value `Editor*TemplateFile`.
2. If no value is specified for the preference parameter, it will use the **default file name** `sdt.tpl`.
3. The directories where the SDL Editor will search for the file are:
   - The current directory
   - Your home directory (`$HOME` **on UNIX** and `%HOME%` **in Windows**)
   - The installation directory (`$sdtrelease` **on UNIX**)
4. If no file can be located, either with a specified name or the default name, you get the option of searching manually (i.e. from a File Selection Dialog) for files with the standard file name extension `.tpl`.

5. If you decline by clicking *Cancel*, the Grammar Help window will appear without contents. You can now locate a grammar help file by choosing *Load* (or *Merge*, which will have the same effect in this case).

Using the Signal Dictionary

Messages from a Message Sequence Chart and signals from an external signal dictionary may be included into the SDL Editor's signal dictionary.

All functionality is provided in the *signal dictionary window*. Each SDL Editor window has its associated Signal Dictionary window.

> **Note:**
>
> The signal dictionary function requires a file with grammar help templates (`sdt.tpl`) in order to function properly. See "The Grammar Help File" on page 1912.

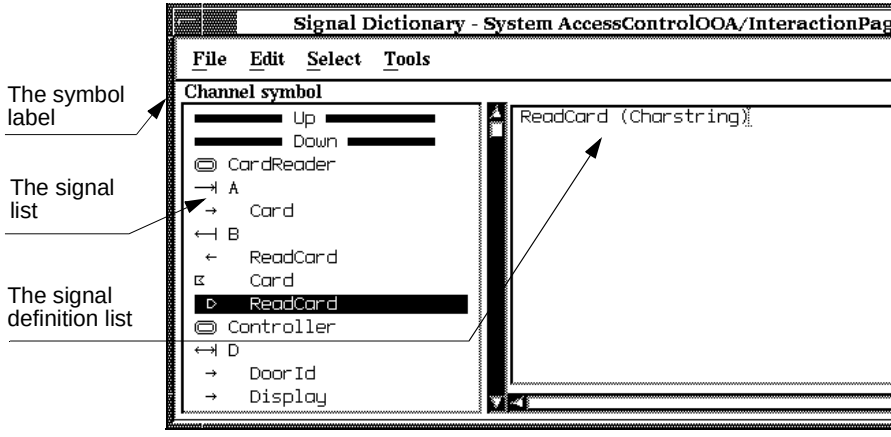


Figure 404: The Signal Dictionary window **(on UNIX)**

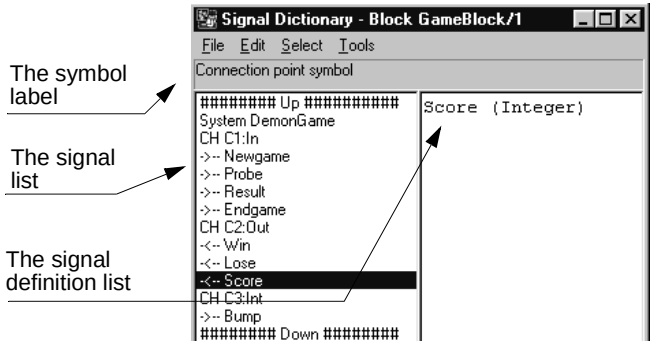


Figure 405: The Signal Dictionary window **(in Windows)**

Signal Dictionary Update

The signal dictionary function stores information about signals in an *information server*. The information server updates its contents **each time an SDL diagram is saved**. When the diagram is saved by the SDL Editor, the SDL Analyzer is invoked and produces a mirrored image which is an SDL/PR description of the SDL/GR diagram. That information is then loaded into the information server which serves the SDL Editor with information upon request. All of this is done automatically.

> **Note:**
>
> In order to extract signal information from a diagram, the SDL suite requires the diagram to be syntactically correct in the sense that SDL/PR code can be generated from the SDL/GR diagram without encountering errors.

Opening the Signal Dictionary Window

- Select *Signal Dictionary* from the SDL Editor *Windows* menu. The signal dictionary window is issued (see Figure 404 on page 1917 and Figure 405 on page 1918).

Setting the Mode to Graphical or Textual (UNIX only)

On UNIX, the signal dictionary window can take advantage of your computer's font capabilities to present the information in a graphical way, using SDL-like notation. This is the default mode.

If the information is presented using strange characters, your terminal does not support the font family used by the signal dictionary window. You should use the textual mode.

To turn the mode to textual:

1. Set the environment variable `SDLENOGRAPHICS`
   - If you are to run several Signal Dictionary sessions in this mode, you may want to set the environment variable in a file that you will automatically source (e.g. `.cshrc`)
2. Exit all SDL suite tools.
3. Restart the SDL suite to have the environment variable affect the behavior.

Requesting Signal Dictionary Services

When you select an object where a reference or definition of a signal makes sense, the signal dictionary window is automatically updated to reflect the signals and *signal conveyors*[1] that are available according to what options you have selected.

The signal dictionary function responds when selecting the following objects (see "Object" on page 1929 for more details):

- channel
- signal route
- connection point
- gate
- signal input
- signal output
- priority input
- save of signal
- text symbol

Updating the Signal Dictionary

The signal dictionary is automatically updated each time you save an SDL diagram or an MSC that is referred to in the Organizer's SDL System Structure chapter.

1. The term *signal conveyor* denotes a communication path that conveys signals; a channel, signal route, a gate or connection point.

When modifying an SDL diagram, you may have unintentionally introduced SDL constructs that are incomplete and thus cause the signal dictionary to fail in providing signal information for that diagram.

Where the signal dictionary fails on a diagram, a bug symbol **(on UNIX)**, or an error message **(in Windows)** will appear after the diagram name in the signal list. Correct the source of error and save the diagram again. For a reference on signal dictionary errors, see "Error Notification" on page 1935.

Specifying the Signal Dictionary Options

You can customize the signal dictionary to fit your needs, by restricting or extending the amount of information that is presented through a set of options that you can activate or deactivate.

Depending on the method you are following when designing with SDL, you should activate the required options.

Next follows a guide for when to use the available options.

To enable an option:

1. Activate the *Select* menu.
2. Toggle the appropriate option on by selecting it (an option which is enabled is indicated with an asterisk preceding the option name). Selecting the option once again disables it.
   - Alternatively, select the *Options* menu choice and toggle the buttons on or off. When satisfied, click *OK*.

> **Note:**
>
> The more options are enabled, the more information is computed by the SDL Editor and the more time it takes.

Bellow follows a few general guidelines for enabling the various options, depending on the approach you are following when designing with SDL.

Working Top-Down

This expression means starting by designing the root diagram (e.g. the system diagram) and continuing with the block diagrams, next the process diagrams and so forth.

If you follow this approach, selecting the *Up* option will enable access to signals used one level up in the SDL hierarchy. This option is enabled by default.

– You may also type `<Ctrl+u>` to toggle the option. Typing **`u`** brings the *Up* option into view (if enabled).

Working Bottom-Up

Following this method means starting with the diagrams at a deeper level (e.g. procedures) and working upwardly in the SDL hierarchy.

If you work bottom-up, select the *Down* option to enable access to signals used one level down in the SDL hierarchy. This option is enabled by default.

– You may also type `<Ctrl+d>` to toggle the option. Typing `d` brings the *Down* option into view (if enabled).

Working with Local Signals

If you look for entities that are used locally in a diagram, select the *This* option. Remember, a diagram needs to be saved in order to update the signal dictionary.

– You may also type `<Ctrl+t>` to toggle the option. Typing `t` brings the *This* option into view (if enabled).

– To find the definition of the signal in the *This* section, you may select an object in the drawing area that uses the signal and type `<Ctrl+f>`.

Listing all Signals

You can list all signals that are visible according to the SDL scope rules by turning the *All* option on. For instance, a signal that you declare on the system level will be available in all diagrams in the SDL hierarchy.

– You may also type `<Ctrl+a>` to toggle the option. Typing `a` brings the *All* option into view (if enabled).

Using Packages

If you want to gain access to signals that you have declared in a package, you should enable the *All* option.

Using Diagram Types

If you use the diagram type concept, you should enable the *All* option to gain access to the signals that are defined in the diagram types.

Listing MSC Messages

If you describe the dynamic behavior of an SDL diagram using the MSC Editor, you may take advantage of this by turning the *MSC* option on.

> **Note:**
>
> Messages used in an MSC will be available in the signal dictionary only if it is associated with an SDL diagram, i.e. linked into the SDL diagram structure.

Importing an External Signal Dictionary

You may import an external signal dictionary into the SDL suite through the Postmaster interface. This is described in *"Load External Signal Definitions into the Information Server" on page 680 in chapter 13, Using the Telelogic Tau Public Interface*.

Locating the Source Diagram

Note: Signal Dictionary and OO

If you are looking for a source diagram that is an SDL-92 type (system type, block type, process type or service type) and that you access through an instance of the type, the following conditions must be respected in order to have the signal dictionary window list the diagram and list all of the signals and signal conveyors that are used in the diagram:

- The type diagram **must be referred in the same diagram as where it is instantiated**. In the signal dictionary window, only the type will be listed, not the instance. Figure 406 shows an example of this.
  - As a consequence of this, you will not be able to look into the instance of a type that is defined in a package diagram and used in the current SDL system.
- In the case the type is inherited from a supertype, the entities that you are looking for **must be defined in the child type**.

The main cause of this limitation is that channels and signal routes are connected to the instance, not to the type.

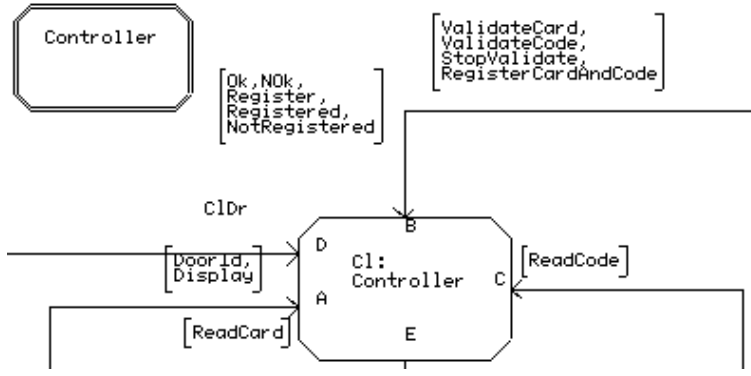


Figure 406: The type is referred where instantiated

The process type Controller is instantiated as Cl. The process instance Cl will be visualized as the process type Controller, and you can list signals conveyed on the gates A, B, C, D and E.

To locate the source diagram (i.e. the diagram where the signal is used or where the signal conveyor is available):

1. Locate the section (*Up*, *This*, *Down*, *All*, *MSC* and *External*) by scrolling up or down. The sections are listed in that order and the start of each section is identified with a separator with the corresponding name.

   – You may also use the keyboard accelerators `<Ctrl+u>`, `<Ctrl+t>`, `<Ctrl+d>`, `<Ctrl+a>`, `<Ctrl+m>` and `<Ctrl+e>` to toggle the respective option on and off.

   – Use the keyboard accelerators `u`, `t`, `d`, `a`, `m` and `e` to bring the respective section into view.

2. Identify the source diagram by its type and name. Diagrams are listed alphabetically by their name. The diagram type is identified either by a graphical SDL-like symbol (see Figure 414 on page 1931) in the left margin **(UNIX only)**, or by a text string that consists of the diagram type.

Updating the Text in the Target Diagram

Once you have located the source diagram your next task is to insert the text belonging to the object you have selected in the target diagram (the diagram where the change is to be done).

Updating a Signal / Timer Output, Input, Priority Input or Save

This operation makes sense only on pages where the implementation is described, i.e. the flow pages.

1. Locate the matching signal or timer in the signal list.

   – Signals are identified graphically with the SDL signal input and/or signal output symbol **(UNIX only)**, or with the characters ‘I’, ‘O’ or ‘?’ (see “Signal and Signal List” on page 1933.

   – Timers are identified with the MSC timer symbol **(UNIX only)**, or the character ‘T’ (see “Timer” on page 1935).

2. Make sure the signal output, input, priority input or save symbol to be updated is selected in the target diagram.

3. Select the signal or timer item in the signal list. Use *Insert* or *Replace* from the *Edit* menu to update the symbol contents. The target symbol is updated with the signal name and its formal parameters.

   – Alternatively, double-click the signal or timer.

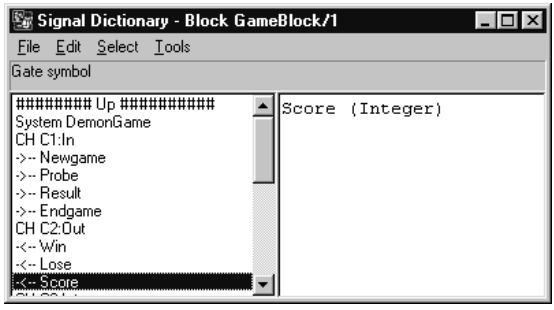


Figure 407: Selecting the signal Display

4. Update the signal's formal parameter type with a current parameter.

Updating a Signal Conveyor (Channel, Signal Route, Connection Point or Gate)

1. Locate the signal conveyor that constitutes the corresponding item that interfaces to the channel, signal route or gate that you are to update in the current diagram.

   – In graphical notation **on UNIX**, these entities are identified graphically by a symbol which has the appearance of the corresponding SDL concept (see Figure 415 on page 1932, Figure 416 on page 1932 and Figure 417 on page 1933).

   – Channels, signal routes and gates are identified textually with the abbreviations CH, SR and GA, respectively.

2. Make sure the **signal list** is selected in the target diagram.

3. Select the signal conveyor. Use the *Insert* or *Replace* from the *Edit* menu to update the signal list contents.

   – Alternatively, double-click the signal or timer.

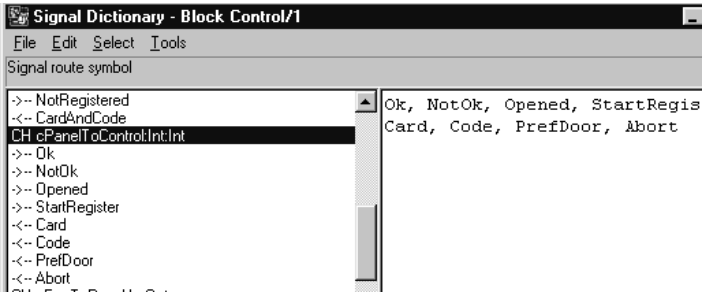


Figure 408: Selecting the Channel cPanelToControl:int:int

4. In the case of a bidirectional conveyor, you will see lines of text in the right window, each one corresponding to a signal list (as in Figure 408, above). You should copy each line of text to the corresponding signal list. Select the signal list in the SDL Editor and a line, then use *Insert* or *Replace* to update the signal list.

Updating a Text Symbol

When updating a text symbol, you are likely to declare signals and timers that you have referred to in the current diagram or referred to in the SDL hierarchy descending from the current diagram.

1. You should make sure the *Down* and *This* options are selected.
2. Remember to add a SIGNAL or TIMER keyword in the text symbol, before the enumeration of signals.
3. Locate all diagrams of interest. Look for signals / timers that you want to declare. Double-click each signal, and insert the necessary commas to create a syntactically correct declaration. Terminate the operation by inserting a semicolon.

Finding a Definition

When you select an object in the SDL Editor's drawing area and then type `<Ctrl+f>` the first occurrence of the first word in the object is looked for and selected in the This section.

Closing the Signal Dictionary Window

You close the signal dictionary window by selecting *Close* from the signal dictionary *File* menu.

The Symbol Label

The *symbol label* field is a non-editable text field that displays the type of object that is currently selected in the SDL Editor.

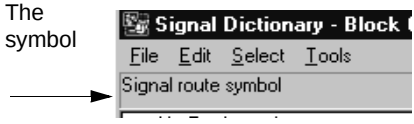


Figure 409: The symbol label

The symbol label reads “No single symbol selected” in the following circumstances:

- No object is currently selected in the SDL Editor
- More than one object is selected.
- The symbol type is not defined in the Grammar Help file.

The Signal List

The *signal list* is a selectable list where all occurrences of signals that are found according to the current selection criteria are listed.

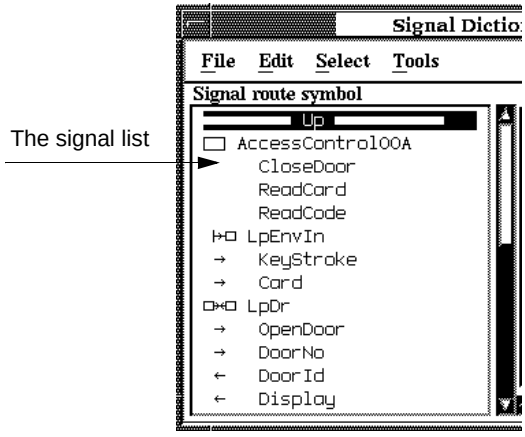


Figure 410: The signal list **(on UNIX)**

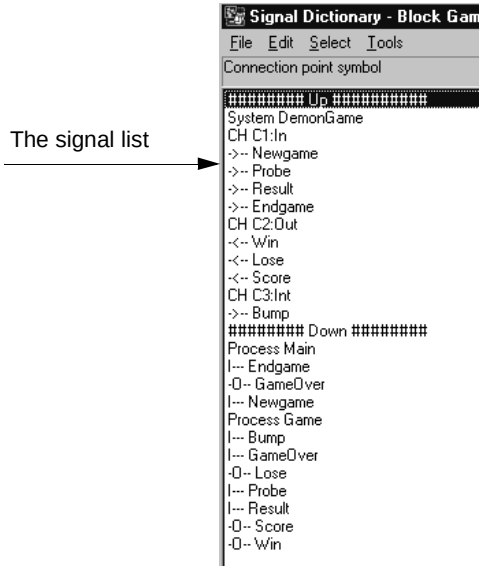


Figure 411: The signal list **(in Windows)**

The signal list is empty when:

- The symbol label reads “No single symbol selected”
- The signal dictionary function does not make any sense on the selected object. i.e. the selected object does not refer/define a signal.

The following objects are recognized as carrying signal information:

Object	Usage
channel line channel name signal list in channel	Reference to signal
connection point	Reference to signal
gate gate name signal list in gate	Reference to signal
input	Reference to signal
output	Reference to signal
priority input	Reference to signal
save	Reference to signal
signal route line signal route name signal list in signal route	Reference to signal
text	Declaration of signal

Signal List Notation and Syntax

Each line in the signal list follows a notation that informs about the various diagrams and symbols where signals are used and also about the context.

In Windows, this notation is purely textual.

On UNIX, the notation that is adopted is an SDL-like graphical notation. It is also possible to select a textual notation if the font face used in the graphical notation is not supported by the computer. You select the textual notation by setting the environment variable `SDLENOGRAPHICS` before you start the SDL suite.

Each item in the signal list is one of the following:

- *Separator*
- *Diagram Identifier*
- *Gate*
- *Channel*
- *Signal Route*
- *Signal and Signal List*
- *Timer*

Separator

The signal list is divided into several sections. The top of each section is clearly marked with a separator, having the appearance of a thick line (graphical notation **on UNIX**) or a number of hashes ("###...###") preceding and following a text that identifies the section.

Figure 412: A signal list separator (graphical notation **on UNIX***)*

Figure 413: A signal list separator (textual notation)

Each of these sections corresponds to an option that is enabled in the *Select* menu (see "Select Menu" on page 1964). Their meaning and order of appearance in the signal list is as follows:

- *Grammar*

  SDL Grammar Help section. See "Using Grammar Help" on page 1906 for more information about this topic.

- *Up*

  A list of all SDL signals that are available[1] by looking one level up in the SDL hierarchy. The *Up* section is empty when working on the top diagram (e.g. a system diagram).

- *This*

  All SDL signals that are available by looking at the current diagram, i.e. *this* diagram.

1. The term "available" means defined and / or referenced.

- *Down*

  All SDL signals that are available by looking at all diagrams one level down in the SDL hierarchy. The Down section is empty if the current diagram has no diagrams below it in the hierarchy.

- *All*

  All defined SDL signals that are visible for the current diagram according to the *SDL scope rules*.

- *MSC*

  Signals that have been exported from a Message Sequence Chart into the SDL Editor signal dictionary facility. An MSC must be *Associated* with an SDL diagram in the Organizer's diagram structure in order to include its signals into the Signal Dictionary (see "Associate" on page 97 in chapter 2, *The Organizer*).

- *External*

  Signals that have been imported from an external signal dictionary. An external signal dictionary is imported through the public PostMaster interface of the SDL suite. See *"Load Definition File" on page 625 in chapter 12, The Telelogic Tau Public Interface*.

Diagram Identifier

In the graphical notation **on UNIX**, diagrams are identified with an SDL Reference symbol in the left margin which identifies the diagram type, followed by the diagram name:

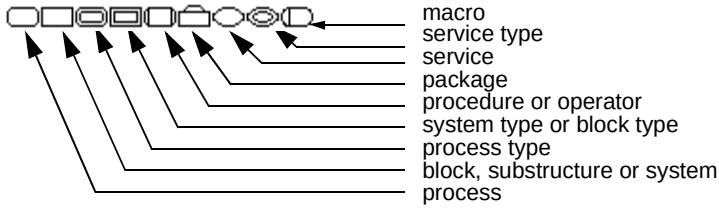


Figure 414: Diagram type symbols (graphical notation **on UNIX***)*

The textual notation reads `<diagramtype diagramname>`

All information that is listed after a diagram identifier is related to that diagram, until the next diagram identifier or separator.

Gate

In the graphical notation **on UNIX**, gates are identified by an arrow pointing into or out from the frame, followed by the name of the gate:

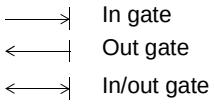


Figure 415: Gate symbols (graphical notation **on UNIX***)*

The textual notation reads:

```
GA <Gatename>:<In>|<Out>|<In:Out>
```

Signals on a gate are the lines that appear after the gate and before the next gate, channel, signal route or diagram identifier. The lines consist of all signals or signal lists that are conveyed on the gate, the signals sent to the diagram appearing first and the signals sent from the diagram appearing last.

Signals are listed according to the order of appearance in the respective signal lists.

If you select a gate in the list, one or two lines is displayed in the signal definition list. In the case of a bidirectional gate, the first line holds the signals sent to the diagram while the second line holds the signals sent from the diagram.

Channel

In the graphical notation **on UNIX**, channels are identified by an arrow between the frame and a block symbol (in the case of a channel to/from the environment), alternatively with an arrow between two block symbols (an internal channel):

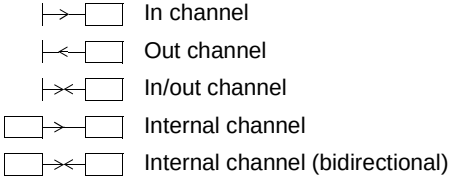


Figure 416: Channel symbols (graphical notation **on UNIX***)*

The textual notation of a channel is:

```
CH <Channelname>:<In>|<Out>|<In:Out>|<Int>|<Int:Int>
```

Channels are present on package, system, block, block substructure and system type diagrams only.

The lines of information that follow a channel consist of the signals that are conveyed on it. The appearance is similar to listing signals on a gate. See "Gate" on page 1932 for information about listing order.

Signal Route

In the graphical notation **on UNIX**, signal routes are identified by an arrow between the frame and a process symbol (a channel to/from the environment), alternatively with an arrow between two process symbols (an internal signal route).

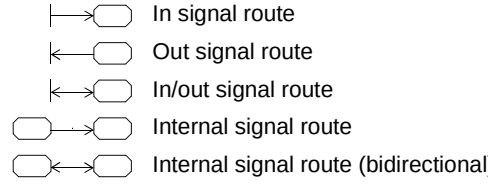


Figure 417: Signal route symbols (graphical notation **on UNIX***)*

The textual notation of a signal route looks like this:

```
SR <Signalroutename>:\
<In>|<Out]>|<In:Out>|<Int>|<Int:Int>
```

Signal routes are present on block and block type diagrams only.

The lines of information that follow a signal route consist of the signals that are conveyed on it. The appearance is similar to listing signals on a gate. See "Gate" on page 1932 for information about listing order.

Signal and Signal List

Signals and signal lists can appear in the following contexts:

- Signal Definitions (i.e. declarations in text symbols)
- Signal References on Lines (i.e. references in gates, channels and signal routes)

- Signal References in Flow Diagrams (i.e. references in flow diagrams; processes, process types and procedures).

The notation that is adopted differs slightly depending on the context.

A **signal definition** appears directly after the diagram identifier where the signal is declared. The signal is identified by its name and its location to distinguish it from other items.

The indentation is filled with blank space (see Figure 418).

```
System AccessControlOOA
     CloseDoor
     ReadCard
     ReadCode
```

Figure 418: Signal definitions

Note:

Each signal appears only once within a diagram. If a signal is referenced at least once within the diagram, it is not listed in the signal definitions section for that diagram, only references are listed.

A **signal reference** on a line uses a similar notation as for signal definitions. The indentation consist of an arrow that identifies the direction of the signals (see Figure 419).

```
lprc:Int:Int
-->- ValidateCard
-->- ValidateCode
-<-- StopValidate
```

Figure 419: Signal references on a line

Signal references in flow diagrams are listed along with the context the signal is used. This is indicated by the following attributes (multiple attributes can be set):

- Input[1]
- Output
- Reference to an undefined signal.

1. Save of signal is also considered as input by the Signal Dictionary function.

In the graphical notation **on UNIX**, these attributes are shown using SDL input and SDL output symbols, and a question mark for showing that the signal is not declared:

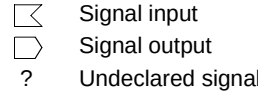


Figure 420: Signal symbols (graphical notation **on UNIX***).*

The textual notation uses the characters:

```
[I][O][?]
```

These characters appear at fixed positions. Where an attribute is not set, this is indicated with a hyphen ('-'). See Figure 421.

```
IO?- Undefined
```

Figure 421: An undeclared signal which is input and output

Timer

A timer is handled identically as described in "Signal and Signal List" on page 1933. To indicate that the item is a timer, the graphical notation **on UNIX** uses the MSC symbol for timer.

Figure 422: Timer symbol (graphical notation **on UNIX***).*

In the textual notation, a 'T' appears as the rightmost attribute.

Timers can appear in the following contexts:

- Definitions of timers
- Set statements
- Reset statements
- Active statements.

Error Notification

If the SDL suite fails in extracting signal information from some SDL diagrams, the information contained in that diagram is not available to the signal dictionary function.

In the graphical notation **on UNIX**, this is indicated in the signal list by a “bug” symbol located to the right of the diagram names.

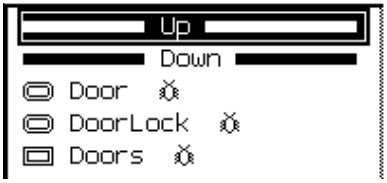


Figure 423: Errors were detected when extracting signals (graphical notation **on UNIX***)*

The SDL suite failed in extracting signal information from the process type diagrams Door, DoorLock and the block type diagram Doors (one level down in the SDL hierarchy)

In the textual notation, this is indicated by an error message following the diagram name:

```
<diagramtype diagramname> :GrPrError
```

```
######## This ########
Process Type Door :GrPrErr
```

Figure 424: Errors were detected when extracting signals

The current diagram (This diagram) is process type Door. The SDL suite could not convert this diagram to SDL/PR.

These error situations are either caused by:

- Syntactically incorrect SDL which causes the SDL suite to fail when converting the SDL/GR diagram to SDL/PR.
- Syntactically incorrect SDL which causes the information server to fail when parsing the contents of the SDL/PR file. When possible, the signal definition field will contain a graphical reference to the source of error. Double-clicking that graphical reference will display the source of error in an SDL Editor window.

The error reports are listed in the Organizer low window. See also “Signal Dictionary Update” on page 1918.

The Signal Definition

The signal definition contains the following information:

- If a signal or timer is selected in the signal list, the signal or timer is listed with its parameters (see Figure 425).

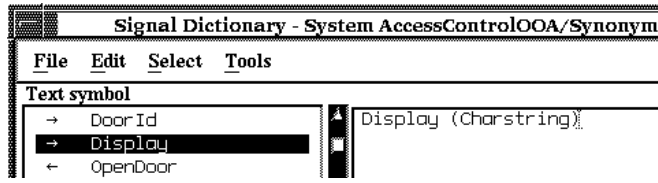


Figure 425: Selection of a signal

The signal Display is selected. The signal definition shows the signal along with its parameters - one parameter of type Charstring in the example.

- If a channel, signal route or gate is selected, the signal definition holds one or two lines of information. Each line contains a list of all signals that are conveyed (see Figure 426).
  - The first line contains the signal list to the parent diagram
  - The second line contains the signal list from the parent diagram.

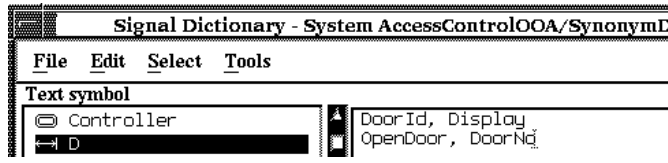


Figure 426: Selection of a gate

The gate D, which is an in/out gate is selected. The signal definition shows one line with all in-signals (Door, Display) and one line with all out-signals (OpenDoor, DoorNo) in relation to the parent diagram (Controller)

- If a signal list[1] is selected, the signal definition holds one line of information containing a list of all signals contained in the signal list. The syntax is

```
SignalListName = Signal1, Signal2,...,SignalN
```

1. A signal list is identified with its name within parentheses, such as `(SignalListName)`

Menu Bars

The SDL Editor's menu bar provides commands available in menus and menu choices for editing diagrams and pages of diagrams. Most of the functionality that the SDL Editor offers is contained within the commands from the menu bar. The menu bar provides the following menus:

- *File Menu*
- *Edit Menu*
- *View Menu*
- *Pages Menu*
- *Diagrams Menu*
- *Window Menu*
- *Tools Menu*
- *Help Menu*

The *File* menu is described in "File Menu" on page 8 in chapter 1, *User Interface and Basic Operations* and the *Help* menu in "Help Menu" on page 15 in chapter 1, *User Interface and Basic Operations*.

In addition, other menus may appear; one is the *Simulator* menu which appears when you set breakpoints in the SDL Editor (see *"Connect-To-Editor" on page 2082 in chapter 50, The SDL Simulator*).

Edit Menu

The *Edit* menu provides editing functions that you can perform on the objects in the drawing area. The *Edit* menu functions that are available differ depending on the type of diagram you edit, either:

- *Interaction Diagrams*
- *Flow Diagrams*
- *Overview Diagrams*

The *Edit* functions are as follows:

- *Undo*
- *Cut*
- *Copy*
- *Paste*
- *Insert Paste* (available on flow diagrams only)
- *Paste As*
- *Clear*
- *Collapse/Expand*

- *Redirect*
- *Bidirect/Unidirect*
- *Class...*
- *Flip*
- *Split Text*
- *Dash/Undash*
- *Mark as Important/Mark as Detail*
- *Mark Types...*
- *Drawing Size*
- *Select All*
- *Insert signal*
- *Complete Word*
- *Select Tail* (available on flow diagrams only)
- *Symbol Border Color > Set Color...*
- *Symbol Border Color > Set Default*
- *Symbol Border Color > <Color>*
- *Symbol Border Color > List All Colors...*
- *Symbol Fill Color > Set Color...*
- *Symbol Fill Color > Set Default*
- *Symbol Fill Color > <Color>*
- *Symbol Fill Color > List All Colors...*
- *Symbol Visibility > Hide*
- *Symbol Visibility > Show*
- *Include Expression*

Undo

With this command you restore the content of the drawing area to its state prior to the operation you performed most recently. It is only available for operations on symbols, lines and text attributes. You can undo the following operations:

- *Cut*, *Paste* and *Clear*
- *Flip*, *Redirect*, *Bidirect/Unidirect*
- *Adding Symbols*
- *Moving Symbols*
- *Resizing Symbols*
- *Drawing Lines*
- *Re-Routing and Reshaping Lines*
- *Resizing a Page*
- *Tidy Up*.

Note: Undo for text editing

You can undo text editing operations in the text window, such as *Import*, *Cut*, *Paste*, and *Clear*. There is also an *Undo* function available for text entered via the Grammar Help or Signal Dictionary functions. See "Undo" on page 1964.

Note: Undo for Class Symbols

If a class symbol is cut and thus also cleared, *Undo* will be enabled. If the Undo command is used, operators will be restored, including their body code.

Cut

This command removes the current selection from the drawing area or text window, and saves it in the clipboard buffer. Also see "Deleting an Object" on page 461.

In text editing mode the command will work on the current text selection.

Copy

This command makes a copy of the current selection, and saves it in the clipboard buffer.

In text editing mode the command will work on the current text selection.

Paste

This command inserts the contents of the clipboard buffer into the drawing area or text window.

No lines will be drawn between the pasted symbols and the symbols already present in the drawing area.

You can interrupt the *Paste* operation by pressing `<Esc>`.

Also see "Pasting an Object" on page 461.

In text editing mode the text contents from the clipboard buffer will be inserted at the current text insertion point.

Note: Cut, Copy and Paste for Class Symbols

When a class symbol is cut or copied, defined body codes are saved on the clipboard, along with their operators. When a class symbol is pasted, previously copied body codes are inserted as well, unless an operator already has defined another body code.

If any operator will be deleted as a result of the operation, a warning is shown, the command is canceled and a dialog similar to the one in "Cutting, Copying and Pasting Class Symbols" on page 1851 is shown.

Insert Paste

This function has the similar functionality as the *Paste* function, but is only available for **flow pages**. It inserts necessary space after the selected symbol before pasting the clipboard buffer and connecting the two flows.

The function requires that the clipboard is not empty and that you have selected exactly one symbol or line. Otherwise, the menu choice is dimmed. The menu choice is also dimmed if connecting the two symbol flows would violate the Z.100 syntax rules.

The functionality differs depending on if you have selected a symbol or a line. If you have selected a symbol the pasted symbols are inserted following the selected symbol. If for example you have selected a decision, the Insert Paste command will create a new branch. Also if a line following the selected symbol joins another line and the selected symbol only can have one follower, the inserted symbols will be placed following both the lines.

If you have selected only a line the pasted symbols will be inserted onto that line. For example if a line following a decision is selected, the inserted symbols will be inserted into the selected decision branch.

Before inserting the selection, the SDL Editor allocates the necessary space by pushing the required objects downwards. If this is not possible, a dialog is displayed.

- Clicking *Increase Page* in the dialog transfers control to the *Drawing Size* dialog, where you can specify the new width and height of the SDL page. The SDL Editor then carries on the operation and inserts the selection.

Paste As

Pastes the currently copied object (from the OM or Text Editor) as an SDL object in the drawing area. The object is transformed and a link is optionally created between the copied and pasted objects.

The *Paste As* dialog is opened. See "The Paste As Command" on page 448 in chapter 10, *Implinks and Endpoints* for more information.

Clear

This command removes the current selection from the drawing area or text window. The content of the clipboard buffer is not affected by this menu choice.

Also see "Deleting an Object" on page 461 in chapter 10, *Implinks and Endpoints*.

In text editing mode the current text selection will be removed.

> **Note: Deleting Class Symbols**
>
> When deleting a class symbol, a warning dialog is shown if an operator will be deleted as a result of this operation. See "Cut, Copy and Paste for Class Symbols" on page 1941.

Collapse/Expand

This command is available for symbols that can be clipped. These are the package reference symbol, additional heading symbol, text symbol, comment symbol and text extension symbol. A collapsed symbol has the minimum size and an expanded symbol will automatically adjust its size to the text.

Redirect

This command changes the direction of a selected channel, signal route or gate. The new direction is denoted by changing the orientation and position of the arrow, leaving the arrowhead pointing in the opposite direction. Subsequent redirections result in the signal list toggling between two directions.

Note:

The **only** way to draw a channel **from the environment to a symbol** is to:

1. Draw a channel from that symbol to the environment
2. *Redirect* the channel.

Bidirect/Unidirect

This command changes the selected channel, signal route or gate to be bidirectional/unidirectional. The Bidirect command will generate a new arrowhead and an empty signal list. The Unidirect command will show a dialog asking what signal list that should be kept.

Class...

This command opens up the *Browse & Edit Class Dialog*. The command is available only when a single class symbol has been selected. The command is not available if the class name contains incorrect syntax.

Flip

This command creates a vertically symmetrical copy of the symbol(s) you have selected. It is valid on the following symbols only:

- Input symbol
- Output symbol
- Priority input symbol
- Text extension symbol
- Comment symbol

The flow lines connected to a mirrored text extension symbol or comment symbol are redrawn in order to reconnect them to the opposite side.

Split Text

This command splits the text from a symbol and its associated text extension symbol (if any) into two parts, where the first part is placed in the symbol itself, and the second part in an associated text extension symbol. The split is done where the text insertion marker is, in the selected symbol.

To place all text in an associated text extension symbol, place the text insertion marker first in the text for a normal symbol and invoke this operation.

To remove an existing text extension symbol associated with a normal symbol, place the text insertion marker last in the text extension symbol and invoke this operation.

Dash/Undash

A dashed reference symbol is used in an SDL subtype when referring to an object that is defined elsewhere in one of the supertypes of the current subtype.

In a similar fashion, a dashed gate indicates that the gate is already defined in one of the supertypes in the inheritance chain.

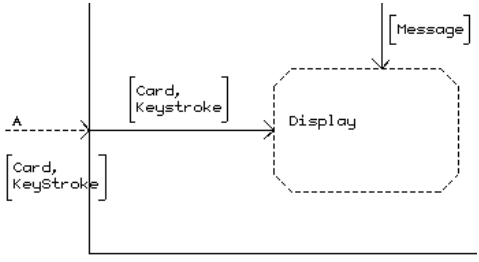


Figure 427: A dashed process reference symbol and a dashed gate

The *Dash/Undash* command is available on the following diagrams:

- *System Types*
- *Block Types*
- *Process Types*
- *Service Types*

The Undash is a toggle command.

Mark as Important/Mark as Detail

In the high-level view, only symbols which are marked as “important” will be shown. For more information, see “Show High-Level View/Show Detailed View” on page 1949.

Symbols that are marked as “important” will be shown with border and fill colors different from ordinary symbols. These colors can be changed through the Preference Manager.

The Mark as Detail command sets the view status of a symbol to “normal”, meaning that it will only be shown in the detailed view.

Mark Types...

This command brings up a dialog with a list of symbol types. The selected symbol types will be visible in the high-level view, while unselected symbol types will only be visible in the detailed view.

Drawing Size

This menu choice sets the size of the *drawing area* for the current SDL page. A dialog is issued where you can type in the width and height. The values are saved on file when you save the diagram.

> **Note:**
>
> Enlarging the drawing, the current SDL page may not fit any longer into the paper that is defined with the Print preferences; the result may be an SDL page that requires multiple sheets of paper when printed.

Select All

This command selects all of the text contained in the text window or all objects within the drawing area.

Insert signal

When an input or output symbol is selected this menu-choice will show a sub-menu with a list of signals available to the system. If an input symbol is selected it will list input signals and if an output symbol is selected output signals will be listed. Selecting a signal will place it in the symbol.

Complete Word

> **Note:**
>
> You can also complete a word by having the mouse pointer in the drawing area and pressing the tab key, provided that the preference Editor*_TabEqualsCompletion_ is true.

Word completion helps you complete long words that have already been used in the same context. Type the beginning of the already used word and invoke word completion to get the rest of the word. If there are several ways to complete the word, the word completion operation will initially give you the common part of the possible completions. If you invoke word completion again, you will get one of the alternatives. To get another alternative, invoke word completion once more.

Word completion takes the word closest to the current cursor position and tries to complete that word by adding characters at the end of the word.

The word completion operation uses the following sources to look for already used words:

- The current diagram page.
- Pages in the beginning of the diagram. The number of pages that the word completion operation examines is determined by the preference Editor*_CompletionMaxPages_.
- The names of entities listed in any running index viewer. This makes it possible to reuse names from other diagrams than the current diagram.
- All words listed in the file .../sdt/include/keywords/sdl.txt in the installation. As default, this file contains SDL keywords.

Select Tail

This command selects the currently selected symbol and all objects beneath it within the tree (following the flow lines).

Symbol Border Color > Set Color...

This command calls the color dialog where it is possible to set a custom color for the selected objects.

Symbol Border Color > Set Default

This command restores the default color for the selected objects. Default color for Symbol borders are set by the Preferences.

Symbol Border Color > <Color>

This command sets <Color> for the selected objects. Default available choices are Black, Blue, Green, Cyan, Red, Magenta, Yellow, and White. Custom colors are added to the menu when they are first used. The most recently used color is moved to the top of the list.

Symbol Border Color > List All Colors...

This menu choice becomes available when a maximum of 20 used colors has been surpassed. When you select *List All Colors* a dialog is issued containing all colors that has been used in the current SDL Editor session and you have the possibility to set them for the selected objects.

Symbol Fill Color > Set Color...

This command calls the color dialog where it is possible to set a custom color for the selected symbols.

Symbol Fill Color > Set Default

This command restores the default color for the selected symbols. Default fill color is set by the Preferences.

Symbol Fill Color > <Color>

This command sets <Color> for the selected symbols. Default available choices are Black, Blue, Green, Cyan, Red, Magenta, Yellow, and White. Custom colors are added to the menu when they are first used. The most recently used color is moved to the top of the list.

Symbol Fill Color > List All Colors...

This menu choice becomes available when a maximum of 20 used colors has been surpassed. When you select *List All Colors* a dialog is issued containing all colors that has been used in the current SDL Editor session and you have the possibility to set them for the selected symbols.

Symbol Visibility > Hide

This command hides the selected symbols. The hidden symbols will be replaced with a thick vertical line. The flow line structure will not be altered. The [Screen, Print]Z100Symbol preferences makes the replacement line as this as the flow line. See also *Show hidden symbols* and *Include Expression*.

Symbol Visibility > Show

This command restores the selected symbols.

Include Expression

This command displays a dialog, where it is possible to specify the include expression that is presented above the top left corner of a symbol, for the selected symbols.

The include expression is a name of a boolean variable (or external synonym). The value of the include expression is checked before analyzing an SDL system, when SDL/GR is converted to SDL/PR. Only those symbols that...

- have no include expression
- have an include expression that evaluates to true

...are included in SDL/PR and are visible for the analyzer and the code generator.

As default, all boolean variables have a value of false. To change a boolean value to true, an external synonym file (a plain text file with extension *.syn) must be created and placed close to the SDL system in the Organizer. The external synonym file closest to the SDL system will be used. An external synonym file contains lines of name value pairs. For instance:

```
DEBUG true
VERSION1 false
VERSION2 true
```

Note that all lines going to a symbol that is removed, will also be removed. An exception to this rule is that flow lines going to a symbol that will be removed, are reconnected to the first following symbol that will not be removed, if there is one and only one such symbol.

It is also possible to decide if SDL/GR symbols should be included or excluded in SDL/PR by hiding symbols. See “Analyze” on page 111 in

chapter 2, *The Organizer*, Symbol Visibility > Hide and Symbol Visibility > Show.

View Menu

The *View* menu provides rescaling functions and access to various options that affect the behavior of the SDL Editor.

The following menu choices are available in the *View* menu:

- *Set Scale*
- *Show High-Level View/Show Detailed View*
- *Temporary Colors > Remove*
- *Temporary Colors > Make Permanent*
- *Window Options*
- *Diagram Options*
- *Editor Options*

Set Scale

Issues a dialog where you can adjust the scale.

Show High-Level View/Show Detailed View

The high-level view shows only symbols which are marked as “important”, whereas the detailed view shows all symbols (normal view). This makes it possible to work with a less detailed version of the SDL diagrams.

By selecting the high-level view, only symbols which are either marked as “important”, or belong to one of the types selected in the Mark Types dialog, will be visible. Also, a tidy up operation will be performed, and the diagram will be read-only - all editing has to be done in the detailed view.

By selecting the detailed view, the diagram is reverted to the original, detailed view, where all symbols can be seen and edited.

Temporary Colors > Remove

Temporary colors are colors that are not saved in the SDL diagram. These colors are used by certain operations, such as SDL trace from SDL simulators and *Compare Diagrams* in the SDL editor, to highlight symbols of interest. Before doing a new operation that relies on color,

it might be useful to remove already existing temporary colors. This can either be done by using this menu choice or by closing down the SDL editor.

Temporary Colors > Make Permanent

Temporary colors are explained in *Temporary Colors > Remove*. This menu choice is used to convert temporary colors into permanent colors that can be saved in diagram files for future use.

Window Options

This menu choice issues a dialog where you set the options that affect the window properties.

- *OK* applies the options as defined in the dialog to the current window only.
- *All Windows* applies the options as defined in the dialog to all windows opened by the SDL Editor.
- *Tool Bar* determines whether the tool bar should be displayed or not.
- *Status Bar* determines whether the status bar should be displayed or not.
- *Symbol Menu* determines whether the symbol menu should be displayed or not.
- *Text Window* determines whether the text window should be visible or not.
- *Page Breaks* determines whether *physical page breaks*, with the appearance of dashed horizontal and vertical lines, should be displayed or not in the drawing area. These page breaks are defined by the print preferences; they show where the print utility will break the diagram into multiple printout pages.
- *Show Grid* determines whether the current symbol grid points are shown as small markers.

Diagram Options

This menu choice issues a dialog where you set the options that affect the editing properties applied on the current diagram.

Interaction and Flow Diagram Options

A dialog with the following options is displayed on interaction and flow diagrams.

- *Use Grid*

  This option determines whether or not the symbols that are added, resized or moved should adhere to the symbol grid or not. By default the symbol grid is active.

  By disabling the symbol grid, the SDL Editor will use the minimum grid of 5 * 5 mm.

- *Layout syntax check*

  This option determines whether the SDL Editor should adhere to the formal SDL syntax rules regarding symbols and the connection of symbols. Ideally the layout syntax check should always be on.

  – Turning syntax checking *off* allows you to insert any symbol displayed in the *symbol menu* into the drawing area. In this mode, symbols can be interconnected in any manner. Furthermore, the connector symbol will always be treated as an inconnector in this mode. Therefore the Analyzer will detect errors when trying to analyze the diagram. To avoid problems turn off the syntax checking only if you want to draw diagrams that never should be analyzed.

Note:

Turning the switch from off to on will **not** perform a retroactive syntax check for not allowed or interconnection of symbols. Only the objects that you add while syntax checking is enabled are checked.

- *Textual syntax check*

  This option determines whether the SDL Editor should perform a syntax check on all the texts in the diagram. Ideally the textual syntax check should always be on.

  – Turning syntax checking *off* disables the textual syntax check except for the kernel heading and the reference symbols which are always checked.

- *Additional Heading only on first page*

This option determines whether the SDL Editor should present the Additional Heading symbol on each page. When the option is on, only the first page in the diagram will show the Additional Heading symbol.

The diagrams options are also saved on file when the diagram is saved.

Overview Diagram Options

A dialog with the following options is displayed on *overview* diagrams.

- *Show line names*

  This option governs whether the names of lines (such as channels and signal routes) should be displayed or not.

- *Show signal lists*

  This option governs whether the signal lists associated to channels and signal routes should be displayed or not.

The diagrams options are also saved on file when the diagram is saved.

Editor Options

This menu choice issues a dialog where you can customize the behavior of the SDL Editor.

The options are controlled by toggle buttons. They are:

- *Always new Window*

  This option indicates whether or not a new window should be opened whenever you select the *New* or *Open* command or any command from the *Pages* menu.

  The default behavior is not to open a new window.

- *Sound*

  This option indicates whether or not improper actions in the SDL Editor, such as attempting to overlap symbols, should be brought to your attention by producing an alert sound.

  The default value for this option is on.

- *Show link endpoints*

This option indicates whether endpoint markers should be displayed for symbols having link endpoints. Regardless of the setting of this option, link endpoints are never shown when printed.

- *Use tab key for word completion*

  See "Complete Word" on page 1946.

- *Show hidden symbols*

  This option indicates whether hidden symbols should be displayed as a vertical line. If this option is selected hidden symbols will be displayed normally but filled with a grid pattern.

- *Use informal view*

  This option sets the editor in Informal mode. In informal mode the content of the symbols is substituted with the content of the attached comment symbol, if any. The comment symbols are hidden. Symbols without an attached comment symbol are displayed normally.

- *Show signal declaration*

  This option decides if the signal declaration including parameters should be shown in the message area when a signal name is selected by clicking on it in the drawing area. Turn this option off if you think it takes too long for the editor to find the signal declaration. There is a preference deciding the default value of this option: Editor*ShowSignalDeclaration.

Pages Menu

The *Pages* menu contains the following menu choices:

- *First*
- *<Page Name>*
- *Last*
- *Add*
- *Edit*
- *Edit Reference Page*

First

This menu choice opens the first page contained in the diagram. The first page is defined according to the order of appearance in the *Edit Pages* dialog. If the page is already opened, its window is displayed.

<Page Name>

Activating the *Page* menu presents up to four menu choices that consist of the names of the two pages that are sequentially right before and after the page you edit. If you edit the first page of the diagram, the next four sequential pages are shown. If you edit the last page of the diagram, the previous four pages are shown.

When you select one of these page names, that page is opened or restored in the SDL Editor. Use the *Edit* menu choice if you want open other pages.

Last

This menu choice opens the last page of a diagram. See "First" on page 1953 for more information.

Add

This menu choice is a shortcut for adding one page to the current diagram. The *Add Page* dialog is issued and when you click *OK* the new page is shown.

Edit

This menu choice opens a dialog where you can *Add*, *Rename*, *Move up*, *Move down*, *Clear*, *Cut*, *Copy* and *Paste* an SDL Page. See "Page Editing Functions" on page 1967.

Edit Reference Page

This menu choice opens (or, if already active, simply restores) the page in the parent diagram where the name of the current page is referred.

This menu choice is dimmed when the current page is one of the pages contained in the Organizer *Root Document* (in which case there is no parent diagram) or the diagram being edited is not known by the Organizer.

Diagrams Menu

The *Diagrams* menu records all diagrams and pages that are opened by the SDL Editor.

- *Back*

- *Forward*
- *<Diagram Name>*
- *List All*

Back

This menu choice opens the diagram that was previously visible in the same window.

Forward

This menu choice opens the diagram that was displayed in the window before you pressed the back button.

<Diagram Name>

The last edited page always goes to the top of the list, and subsequently moves the other diagrams and pages down a position. A maximum of 9 open pages can be shown. A tenth one will be put at the top of the list, but any subsequent opening of a diagram or page will only show the last 9 that have been opened. Another option – *List All* (at the bottom of the list) – is available to list all the open diagrams in the SDL Editor.

Each item in the menu provides information about the diagram type, its name, a slash (‘/’) followed by a page name, a hyphen and, possibly, the file it is stored on. A diagram that is preceded by an asterisk (‘*’) denotes that it has been modified during the SDL Editor session.

List All

This menu choice becomes available when a maximum of 9 open pages has been surpassed. When you select *List All* a dialog is issued containing all diagrams and pages that are currently open in the SDL Editor and you have the possibility to display them.

Window Menu

The *Window* menu contains the following menu choices:

- *New Window*
- *Close Window*
- *Grammar Help*
- *Signal Dictionary*
- *Class Information*

- *Entity Dictionary*
- *<Window Name>*
- *List All*

New Window

This command opens a new window containing a new view on the SDL page contained in the source window from where you selected this menu choice. You can edit the SDL page in any window.

Close Window

This option closes the open window, but, does **not** close the diagram. All but the last open window can be closed, the last one you must close from the *File* menu, possibly in conjunction with saving information (see "Close Diagram" on page 14 in chapter 1, *User Interface and Basic Operations*).

Grammar Help

This menu choice opens the *Grammar Help* window, see "Grammar Help and Signal Dictionary" on page 1905. Its purpose is to assist you in editing SDL textual elements that are correct according to the SDL grammar.

Each SDL Editor window has its associated Grammar Help window.

The SDL Editor tries to open a file with grammar help templates when opening the Grammar Help window (which file to open by default may be specified as an SDL Editor preference). If the file cannot be located, a dialog is issued where you can choose to select it manually. The filter in the file selection dialog will be set to `*.tpl`.

Signal Dictionary

This menu choice opens the *Signal Dictionary* window, see "Using the Signal Dictionary" on page 1917. It gives access to information about what signals are defined in an SDL hierarchy. Each SDL Editor window has its associated Signal Dictionary window.

Class Information

Opens the *Class Information* window. See "Class Information" on page 1876 for more information.

Entity Dictionary

Opens the Entity Dictionary window. See "The Entity Dictionary" on page 434 for more information.

<Window Name>

If more than one editor window is open the other windows are listed here. The behavior of this list will be similar with the diagrams list in the *Diagrams* menu. The only difference is that the menu items will not provide the diagram file information.

List All

This menu choice is available only if more than 9 editor windows are open, and has the same functionality as the *List All* menu command in the *Diagrams* menu.

Tools Menu

The *Tools* menu contains the following menu choices:

- *Show Organizer*
- *Link > Create*
- *Link > Create Endpoint*
- *Link > Traverse*
- *Link > Link Manager*
- *Link > Clear*
- *Link > Clear Endpoint*
- *Search*
- *Spelling > Comments*
- *Spelling > All Text*
- *Tidy Up*
- *Connect to Text Editor*
- *Navigate*
- *Show > Definition of X*
- *Show > Use of X*
- *Show > Definition or Use of X*
- *Show > Next (Definition or Use of) X*
- *Show GR Reference*
- *Create Bookmark*
- *Compare Diagrams*
- *Merge Diagrams*
- *Add Differences*

- *Generate PR*

Show Organizer is described in "Show Organizer" on page 15 in chapter 1, *User Interface and Basic Operations* and the Link commands are described in "Link Commands in the Tools Menus" on page 442 in chapter 10, *Implinks and Endpoints*.

Search

This menu choice opens a dialog that allows you to search for a text in the current diagram or document in any editor. You can search many diagrams and documents at the same time with this menu choice.

For more information about searching, see "Search" on page 140 in chapter 2, *The Organizer* and "Searching and Replacing Text" on page 1896.

Spelling > Comments

Check the spelling of comments in the current diagram. For more information about the spelling operation, see "Spelling > Comments" on page 144 in chapter 2, *The Organizer*.

Spelling > All Text

Check the spelling of all text in the current diagram. For more information about the spelling operation, see "Spelling > Comments" on page 144 in chapter 2, *The Organizer*.

Tidy Up

This menu choice rearranges the graphical layout of the entire SDL diagram, using default layouting algorithms when calculating the location and size of graphical objects. Before the tidy up operation is started, a dialog appears where you can adjust parameters to the operation:

- *Start transition on new page.* Decides if a state symbol representing the beginning of a new transition should be placed on the same page as already placed symbols, or on a new page.
- *Use extra space after comment and text extension.* Decides if a symbol in a flow, after a symbol with an associated comment or text extension, should be moved below the associated symbol or not.

- *Format input and output text.* Decides if text in input and output symbols should be reformatted or not. If the text is reformatted, the signal name is placed in the symbol, while any parameters are placed in a text extension symbol.
- *Replace textual comments with comment symbols.*
- *Place comment symbols to the left.*
- *No text extension for state.* Decides if text extensions should be used or not if the state name does not fit in the state symbol.
- *No text extension for decision.* Decides if text extensions should be used or not if the decision text does not fit in the decision symbol.
- *Sort states and transitions.* Decides the order state symbols (More precise: state and input symbol combinations) will be presented in the resulting diagram.

Connect to Text Editor

This command issues an external text editor and creates a temporary file from the currently selected text. After this you can only edit the text from the external editor. The SDL Editor is updated every time the external text editor saves the temporary file. When you no longer edit the temporary file the editing control returns to the SDL Editor.

Which external text editor you are to use is defined by the preference SDT*TextEditor.

Navigate

This command navigates to another diagram or to another symbol within the current diagram, depending on the selected symbol. It is available for the following symbols:

- For a reference symbol it will show the diagram being referenced.
- For an instantiation symbol it will show the corresponding type diagram. If there is more than one diagram that matches the type name, a dialog will appear where you can select one of the diagrams.
- For a create request symbol the process diagram for the created process will be shown. If the diagram is ambiguous a selection dialog appears.

- For a procedure call symbol the procedure diagram will be shown. If the diagram is ambiguous a selection dialog appears.
- For a macro call symbol the macro definition will be shown.
- For a state symbol another state or nextstate symbol using the same state name in the same diagram will be shown. A selection dialog will appear if there are more than one occurrence of the name.
- For an inconnector symbol the corresponding outconnector symbol will be shown. If more than one outconnector exists a selection dialog appears.
- For an outconnector symbol the corresponding inconnector symbol will be shown.
- For a class symbol, another class symbol, using the same class name in the same diagram, will be shown. If there is more than one occurrence of the name, a select dialog is shown.

Double-clicking a symbol will have the same effect as using the *Navigate* command. This applies to all navigable symbols, except class symbols.

The *Show* Sub-menu

All menu choices in the *Show* sub-menu require a running index viewer loaded with an up-to-date cross reference file. You can get this for all SDL systems that passes through the SDL Analyzer, by pressing the *Generate Index* quick-button in the Organizer.

Show > Definition of X

This command takes the name of the word closest to the current cursor position as input. If the name represents an SDL entity listed in the index viewer, the SDL Editor will navigate to one of the definitions of that entity. Usually there is only one definition to show, but for states, each state symbol is considered to be a part of the definition and one of them is shown. If you invoke this command once more for the same state name, another state symbol is shown (if there are several).

Show > Use of X

This command takes the name of the word closest to the current cursor position as input. If the name represents an SDL entity listed in the in-

dex viewer, the SDL Editor will navigate to one of the uses of that entity. If you invoke this command once more for the same word, another use of that entity is shown (if there are several uses),

Show > Definition or Use of X

This command takes the name of the word closest to the current cursor position as input. If the name represents an SDL entity listed in the index viewer, a dialog is opened, allowing you to navigate to one of the definitions or uses of that SDL entity.

Show > Next (Definition or Use of) X

This command repeats the last *Show > Definition of X*, *Show > Use of X* or *Show > Definition or Use of X* command using the same word as last time as input.

Show GR Reference

This command issues a message where the graphical SDT reference for the object you have currently selected is displayed.

The syntax of the graphical references used in the SDL suite environments described in chapter 19, *SDT References*.

Create Bookmark

This command creates a bookmark in the Organizer for the object you have currently selected.

Compare Diagrams

With this menu choice you can compare the contents of two SDL diagrams. See "Comparing and Merging Diagrams" on page 1970.

Merge Diagrams

With this menu choice you can compare the contents of two SDL diagrams and create a new diagram by merging the two SDL diagrams. See "Comparing and Merging Diagrams" on page 1970.

Add Differences

With this menu choice you can merge two SDL diagram versions into one. The SDL diagram that is visible in the editor when the operation is

invoked will be updated with information from the other version, for all differences. Symbols from the other diagram might be placed slightly to the left of the original place, to avoid overlapping symbols.

Differences are marked with temporary color, except for complete page differences when a complete page is missing in the other diagram. The reason for this is to support merging of a complete diagram (the SDL diagram in the editor) with diagram parts (containing a subset of all pages in the complete diagram) that might be slightly different.

In the *Add Differences* dialog, it is possible to specify the other diagram to compare with, either by specifying a file name or by specifying another editor buffer. The *Add Differences* dialog also makes it possible to specify the temporary colors to use for highlighting differences.

Generate PR

This command will generate SDL/PR/CIF for the current diagram. A dialog with the following options appears:

- *PR file*

  The name of the PR/CIF file to create.

- *Generate GR references*

  This option will generate GR references that can be used to trace the PR code back to the GR diagram.

- *Generate CIF*

  CIF is the standard defined in ITU recommendation Z.106 to be used when transferring SDL/GR diagrams preserving the graphical layout. With this option set, CIF comments will be generated together with the PR.

The *File* Menu of the Text Window

The *File* menu provides functions that transfer text from a file to the text window and vice-versa. The basic intention is to provide you the possibility to edit larger portions of text with a more suitable text editor (for instance signal definitions in text symbols). Another possibility to edit text externally is to use the *Connect to Text Editor* command in the *Tools* menu.

Import

Import imports the contents of a file into the text window and inserts the contents of the file at the text cursor position, possibly replacing selected text in the text window. In the file selection dialog, the filter is set to `*.txt` by default.

Export

Export exports the selected text to a file. If no text is selected in the text window, the entire text window contents will be exported to the file. In the file selection dialog, the filter is set to `*.txt` by default.

Menu Bar in Grammar Help Window and Signal Dictionary Window

These menu bars contains the following menus:

- *File Menu*
- *Edit Menu*
- *Select Menu*
- *Tools Menu*

File Menu

The *File* menu contains menu choices that access files that contain *SDL Grammar Help templates*.

These commands are useful only when requesting Grammar Help support.

Load

This command opens a file selection dialog, where you can select and load the file with grammar help definitions.

Grammar template files normally have the file extension `.tpl`. With the SDL suite, a standard template file, `sdt.tpl`, is enclosed. The SDL Editor attempts to locate a `.tpl` file each time you invoke the *Load* command.

Any template definitions that are already loaded will be replaced by the contents of the selected grammar help definition file.

Merge

Merge extends the current loaded grammar definitions with the contents of a grammar help definition file.

This command issues a file selection dialog, in which you can select and add a set of custom templates to the basic set provided in the SDL suite.

Any template definitions that are already loaded will be extended with the contents of the merged template definition file.

Edit Menu

The *Edit* menu provides two utility functions that you can use when copying text from the *Grammar* window or the *Signal Dictionary* window to the SDL Editor's text window.

Undo

You can revert changes with the *Undo* command.

Insert

This menu choice copies the entire contents of the *Grammar* field or the signal definition and copies it into the text window at the current I-beam cursor location. Selected text will thus be overwritten.

A shortcut for *Insert* is to double-click on the template name in the name field.

Replace

This command replaces the contents of the text window with the contents of the *grammar field*. This function is a shortcut for the following sequence:

- "*Select All, Clear* and *Insert*" **(Grammar Help)**
- "*Select All* and *Insert*" **(Signal Dictionary)**

Select Menu

The commands on the *Select* menu define what functionality the *Grammar Help* window and the *Signal Dictionary* window should provide.

Each of the options can be individually turned on and off. In addition, an *Options* menu choice (see "Options" on page 1966) allows you to enable and disable multiple options in one command only.

The options that are currently enabled are indicated by an asterisk ('*') preceding the corresponding menu choice.

When you invoke the Grammar Help, the *Grammar* option is enabled, while the remaining options are disabled.

These functions restrict or extend the scope of search for signals. The Signal Dictionary functionality can be extended to provide access to MSC Messages and an external signal library.

Grammar

This toggle option, when set to on, will enable the SDL Grammar Help function.

Up

This toggle option enables the *Up* option provided in the Signal Dictionary window. This option displays the SDL signals that are available when looking one level up in the SDL hierarchy.

This

Turning *This* on lists all signals that are used in the current diagram.

Down

This option displays the SDL signals that are available when looking one level down in the SDL hierarchy.

All

This option displays all signals that are defined and are visible according to the SDL scope rules, starting from the root document (see "Root Document" on page 41 in chapter 2, *The Organizer*) in the SDL System Structure chapter.

MSC

This option displays all messages that are available in the Message Sequence Charts that are submitted as input to the Signal Dictionary func-

tion. Each message is mapped to an SDL signal, with its parameters mapped to SDL signal parameters.

An MSC is submitted as input to the Signal Dictionary if associated to an SDL diagram that is present in the Organizer's diagram structure.

External

This option displays all signals that have been imported from an external signal dictionary. Importing external signals is done through the Telelogic Tau Public Interface. See *"Load Definition File" on page 625 in chapter 12, The Telelogic Tau Public Interface*.

Options

Issues a dialog where you can turn on and off all options individually (see "Select Menu" on page 1964 for a description of the options).

The default options for the Signal Dictionary are *Up* and *Down*.

Tools Menu

The Tools menu provide various convenience functions.

Show Editor

This command displays the parent SDL Editor window.

Show Definition

An SDL Editor comes up showing the definition of the item which is selected in the signal list.

Page Editing Functions

When you select *Edit* from the *Pages* menu, the *Edit Pages* dialog will be opened. In this dialog, you can add, rename, clear, cut, copy and paste an SDL page. The items in the dialog will be described below:

The Page List

This list includes all SDL pages in the SDL diagram. The operations available in the dialog, are only possible to perform if a page is selected in this list.

Edit

This button opens the selected page in the SDL Editor.

Cut

This button removes the selected SDL page from the diagram and saves it in the clipboard buffer. The information in the Organizer that is dependent on this page, will also be removed.

If the page contains reference symbols and these have an underlying structure in the Organizer this structure will be restored when you paste the page in a new position. See "Cutting, Copying and Pasting Reference Symbols" on page 1851.

The button is dimmed if the object you have selected is not an SDL page.

Copy

This button copies the selected page to the clipboard buffer.

The button is dimmed if the object you have selected is not an SDL page.

Paste

This button pastes a previously cut or copied page into the current diagram.

The *Paste* button will be dimmed if:

- The clipboard buffer is empty.
- The current clipboard contents would lead to a syntax violation in the target diagram.

The SDL Editor checks that the contents of the page are syntactically correct. If required a warning will be issued. Then a new dialog is displayed, where you should assign the page a new name. You can specify:

- The new name of the page (two pages with the same name are not allowed within a diagram)
- Possibly, an autonumbered name (see "The Autonumbered Option" on page 1970)
- If the new page should be pasted before or after the current page

Clear

This button removes the selected SDL page from the diagram. A confirmation dialog will be issued before the page is removed, and the SDL Editor will automatically rename autonumbered pages.

Caution!

As stated in the dialog, *Clear* on a page cannot be undone.

If the page contains reference symbols and these symbols have a substructure, this substructure will be removed from the Organizer. As there is no undo for the *Clear Page* command, a warning will be issued and it is possible to keep the substructure as a new hierarchy instead.

Move up

If an SDL diagram page is selected and you click the *Move up* button, the page will be moved up.

Note:

The undo function in the Organizer and in the SDL Suite will have no effect on page moves. It is, however, easy to undo a page move operation by just moving the page back again.

Move down

If an SDL diagram page is selected and you click the *Move down* button, the page will be moved down.

> **Note:**
>
> The undo function in the Organizer and in the SDL Suite will have no effect on page moves. It is, however, easy to undo a page move operation by just moving the page back again.

Add

This button opens a dialog where you can change some settings before new page is created:

- If autonumbering is turned off, you may specify any pagename, which must be in keeping with SDL conventions.
- If autonumbering is on (the default), the page is automatically named using the next available sequential number within the diagram (1, 2, etc).
- If the page is to be inserted before of after the current page.
- If you add a page to a process or process type diagram, you can select if the new page should be a graph page (the default) or a service interaction page.
- If you add a page to a block or block type diagram, you can select if the new page should be a block interaction page or a process interaction page.

New pages are listed in the same order that they were added.

Rename

This button opens a dialog where you can rename the selected SDL page. If the page is part of a block or block type diagram, you can also change the page type (process interaction page or block interaction page) in this dialog.

Autonumbered pages cannot be renamed, unless you turn the option *The Autonumbered Option* off first.

The *Autonumbered* Option

If the option *Autonumbered* is on, a numeric name (1, 2, etc) will be applied to the selected page. In a dialog, you can select to auto-number the selected page or all pages in the diagram.

The *Open This Page First* Option

If the option *Open This Page First* is on, the selected page will be the page that will be displayed by default when you open the diagram. Only one page at the time can have this option set.

Comparing and Merging Diagrams

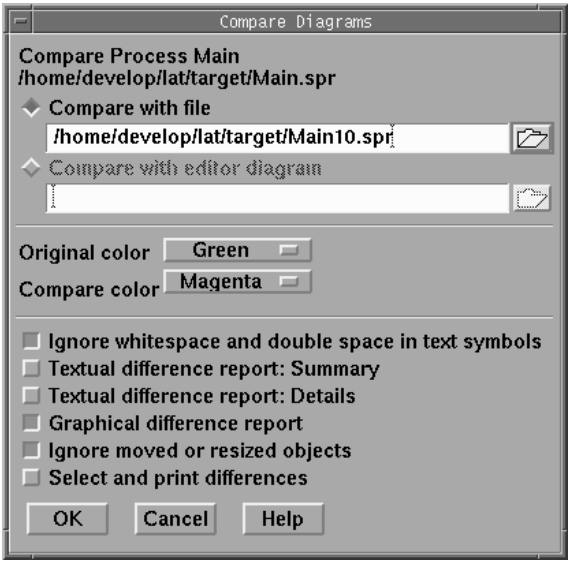


Figure 428: The Compare Diagrams dialog

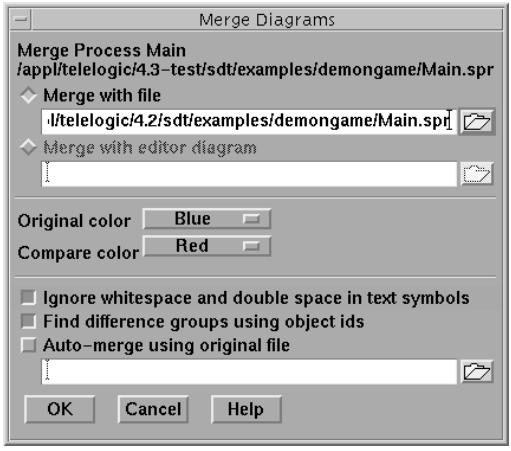


Figure 429 The Merge Diagrams dialog

When you select *Compare Diagrams* or *Merge Diagrams* from the *Tools* menu, a dialog will be opened where you can select the SDL diagram to compare with. (It is also possible to compare more than one diagram pair in one operation, but to do this you should use the corresponding menu choices in the Organizer. See "Compare SDL Diagrams" on page 146 in chapter 2, *The Organizer* and "Merge SDL Diagrams" on page 149 in chapter 2, *The Organizer*.

Specifying the Diagram to Compare With

There are two ways for specifying the diagram that should be compared with the currently displayed diagram:

- You may specify the file in the *Compare with file* field.
- You may specify another diagram already opened the SDL Editor in the *Compare with editor diagram* field. When you click the folder button, a dialog will be opened with a list of all diagrams opened in the SDL Editor (except for the currently displayed).

The option *Compare with editor diagram* has the two restrictions:

– It can only be used if more than one diagram is opened in the SDL Editor.

– The folder button can only be used if more than two diagrams are loaded into the SDL Editor.

Selecting the Report Type

In the Compare Diagrams dialog, there are six alternatives for reporting differences between diagrams: *Ignore white space and double space in text symbols*, *Textual Difference Report: Summary*, *Textual Difference Report: Details*, *Graphical Difference Report (with Merge Facility)* and *Select and Print Differences*.

The Merge Diagrams dialog does not allow you to get a textual difference report. Instead, you always get a graphical difference report and the only option available is *Auto-merge using original file*.

Ignore white space and double space in text symbols

When this option is on, differences caused by different usage of white space are ignored when comparing texts. One space character is considered equal to two space characters.

Textual Difference Report: Summary

This option is only available when comparing using the *Compare Diagrams* menu choice.

A summary of how many differences that has been found is presented in the Organizer log. The summary may look like this:

```
Comparing /home/lat/x.spr
     with /home/lat/y.spr
27 differences (16 symbols, 11 lines) :
  7 changed objects (7 symbols, 0 lines)
  13 removed objects (6 symbols, 7 lines)
  7 added objects (3 symbols, 4 lines)
```

When you read this report, notice the following:

- 27 is the total number of found differences in this diagram pair. This is the total of all changed, removed and added objects.

- An object is considered to be changed if you have moved or resized it, or if you have changed a text associated with the object.

If at least one of the two compared diagrams has more than one page, then a summary is also presented for each page pair. It may look like this:

```
Comparing /home/lat/version1/registeredcard.spt
     with /home/lat/version2/registeredcard.spt
5 differences (1 symbol, 4 lines) :
  1 changed object (1 symbol, 0 lines)
  2 removed objects (0 symbols, 2 lines)
  2 added objects (0 symbols, 2 lines)

Page details:

Page ValidateCard
5 differences (1 symbol, 4 lines) :
  1 changed object (1 symbol, 0 lines)
  2 removed objects (0 symbols, 2 lines)
  2 added objects (0 symbols, 2 lines)

Page RegisterMode
0 differences
```

Textual Difference Report: Details

This option is only available when comparing using the *Compare Diagrams* menu choice.

This option will print information about each found difference in the Organizer log. This makes it possible to navigate from the Organizer log to every found difference by using the *Show Error* operation in the Organizer log. See "Organizer Log Window" on page 179 in chapter 2, *The Organizer*.

The detailed textual difference report may look like this:

```
Comparing /home/lat/d1/db.sbk
     with /home/lat/d2/db.sbk

Comparing page 1

#SDTREF(SDL,/home/lat/d1/db.sbk(1),119(50,35))
ERROR This symbol has been changed...
#SDTREF(SDL,/home/lat/d2/db.sbk(1),119(50,40))
ERROR ...and now looks like this.

#SDTREF(SDL,/home/lat/d1/db.sbk(1),176(95,60))
ERROR This symbol has been removed.
```

```
#SDTREF(SDL,/home/lat/d2/db.sbk(1),182(60,60))
ERROR This line has been added.
```

Graphical Difference Report (with Merge Facility)

The merge facility is only available when merging with the *Merge Diagrams* menu choice, but you can get a graphical difference report both when comparing with the *Compare Diagrams* menu choice and when merging with the *Merge Diagrams* menu choice.

Two SDL Editor windows, named *Source 1* and *Source 2*, displaying the two diagram versions and their differences, and one dialog will be opened. For merge, there will be a third SDL Editor window, named *Result*, showing the merge result.

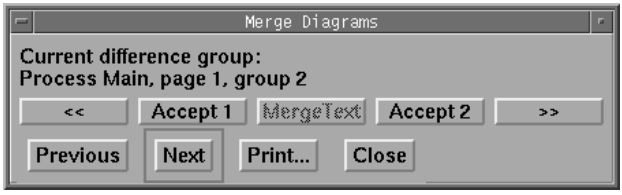


Figure 430: The Merge Diagrams dialog

The Compare Diagrams dialog is similar to the Merge Diagrams dialog. The difference is that the Compare Diagrams dialog is missing the first row of buttons. (“<<“, Accept1, Accept2 and “>>”.)

In the dialog that will be opened, you can:

– Navigate between difference groups with the *Next* and *Previous* buttons.

– (*Merge Diagrams* only.) Navigate between unresolved (not-yet merged) difference groups with the “<<“ and “>>” buttons.

– (*Merge Diagrams* only.) Do a merge, i.e. update the merge result diagram with information from the current difference group

and one of the compared diagram versions, with the *Accept1/Reject1* and *Accept2/Reject2* buttons.

– Compare or merge a text in a text symbol in the current difference group.

– Print the current difference found in the current page pair.

When the second compare dialog is closed, the editor windows are restored to how they appeared before the compare operation.

Comparing text in text symbols

When two text-symbols differ the *CompareText* button will be available in the Text Compare dialog and it will be possible to enter the text-compare dialog.

The Text Compare dialog consists of two lists of text that are placed side by side with a "gutter" that will contain extra information about the differences. This gutter-information consist of:

– A "-" indicates that there is no corresponding line in the text compared to the other text.

– A "|" character indicates that this line differs in the two texts.

The current difference is indicated with a highlight. Buttons make it possible to navigate to *Previous* and *Next* difference item.

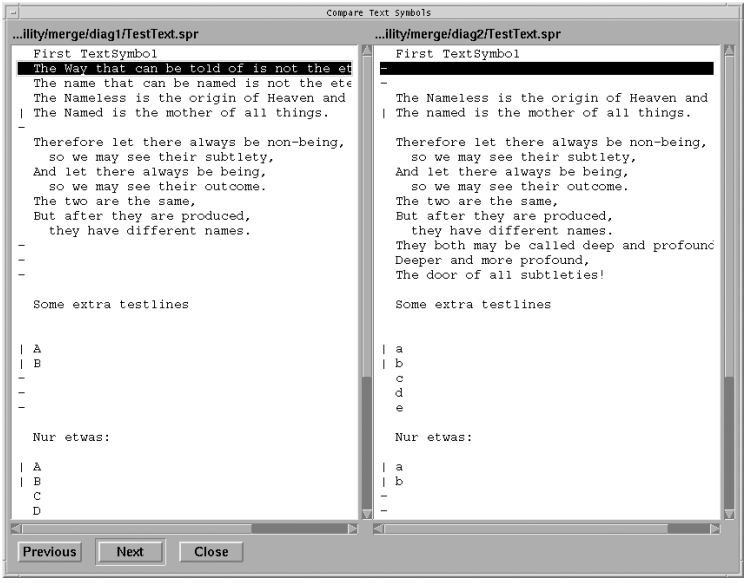


Figure 431: The Compare Text Symbols dialog

Merging text in text symbols

When manual merges are conducted the option to merge text-symbols will always be available in the navigation dialog. When merging a button *TextCompare* button is available as in the compare function. When a text-symbol is selected there will be an option to select either symbol or merge the text in the symbols into a new symbol.

If the text merge is selected a text-merge dialog is presented with three text-lists, the two source texts and the result text. It is possible to navigate between the unresolved differences or between all differences.

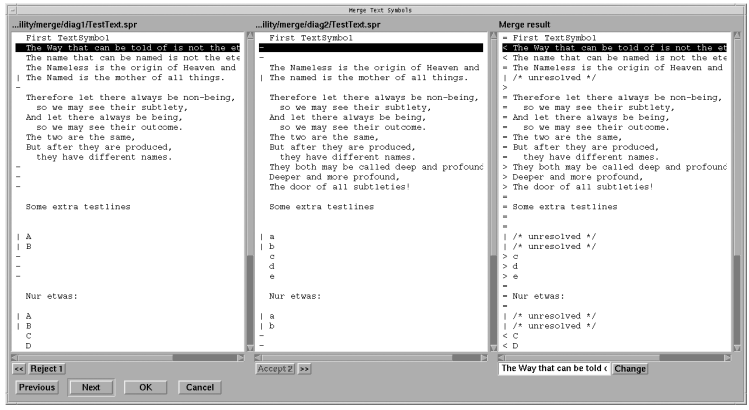


Figure 432: The Merge Text Symbols dialog

This works in the same way as in the normal merge. The buttons “<<“ and “>>” will jump between the unresolved differences and the *Previous* and *Next* buttons will jump between all differences.

The use of *Accept 1*/*Reject 1*, *Accept 2*/*Reject 2* buttons will select neither, either or both the texts for merge. The selections from the source files are presented in the result-list the gutter will show what version is selected. Additional symbols in the result list are:

- A “|” character indicates that this line differs in the two texts.
- A “=” character indicates that the selected text are equal in both versions.
- A “>” character indicates that the right text has been selected for merge.
- A “<” character indicates that the left text has been selected for merge.

It is also possible to edit the result line-by-line by selecting a line and edit it below the list. Clicking the *Change* button will replace the changed line in the result list.

Ignore moved or resized objects

If this option is on, the compare operation tries to ignore differences only caused by moved or resized symbols. You should only use this option if the two compared diagrams originates from the same diagram. (For instance, you have used Save as on the original diagram to get the second copy.) This option will not work well if you create two diagrams from scratch and compare them, because the operation uses object IDs to match moved or resized objects.

Select and Print Differences

If this option is on and there are differences, then a print dialog will pop up when the first dialog is closed. Only SDL diagram pages containing differences will be printed, and the differences will be highlighted with selection markers. If the same page containing differences can be found in both compared diagrams, then the two pages will be printed after each other, beginning with the page from the diagram that was visible in the editor when the compare operation was invoked.

Auto-merge using original file

This option is only available in the Merge Diagrams dialog. It allows you to specify the original diagram file that both the diagram versions that are compared originates from. If you specify the original diagram file, then the merge operation can auto-merge all non-conflicting changes for you. Note that if you give the wrong original diagram, then the auto-merge operation may merge the two versions in the wrong way. If you do not specify an original diagram, then the merge operation leaves all merging of differences to you.

Differences That Will Be Reported

When you use *Compare Diagrams* or *Merge Diagrams*, all visual object differences will be found:

- An object that has been moved, resized, removed, added or painted in a different color.
- A text, associated with an object, that has been changed

An object is considered to be moved if it has been moved relative to the upper left corner of the diagram frame. This means that if you move the frame, it will have no effect.

Diagrams are compared one page pair at a time. A page pair is one page from the original diagram and one page from the other diagram where both pages have the same name. If there is no page with the same name in the other diagram, then the complete page is considered to be a difference.

Difference Groups

Differing objects from the same page pair are grouped into *difference groups*. Two differing objects are placed in the same group if:

- The objects are from the same diagram and they are connected to each other. For instance, a flow line connected to a task symbol.
- The objects are from different diagrams and they would overlap if they were in the same diagram, that is, it would be impossible to place the two objects in the same diagram without changing the position within the diagram for one of them.
- The objects are from different diagrams and the difference operation can conclude that it is the same object that has been moved.

Color and the Current Difference Group

In the graphical difference report, all differing objects are given temporary background colors as specified in the initial dialog. As default, the color is green in the original diagram and magenta in the other diagram. Temporary colors are not saved in diagram files. How temporary colors are removed is described in "Temporary Colors > Remove" on page 1949.

The dialog that will be opened during the graphical difference report, allows you to navigate between difference groups using previous and next buttons. All objects in the current difference group are selected.

Normally, version 1 of the compared diagrams (the diagram that was in the SDL editor when the compare operation was invoked) is shown in a window in the top left corner of the screen. Version 2 (the diagram that was specified in the first compare dialog) is shown in the top right corner off the screen. For complete page differences (a difference where a page can not be found in the other diagram), the window that misses the page is hidden.

Note that it is possible to manually edit the compared diagrams while the compare operation is ongoing. This can be seen as a light-weight alternative to the full merge operation (described next).

Merging

During a merge operation, three windows are used in addition to the second merge dialog:

- The top left corner of the screen is used to show version 1 of the compared diagrams, in the editor window named *Source 1*. This is the diagram that was visible in the SDL editor window when the merge operation was invoked.
- The bottom left corner of the screen is used to show version 2 of the compared diagrams, in the editor window named *Source 2*. This is the diagram that was specified in the first merge dialog as the diagram to compare with.
- The right part of the screen is used to show the merge result diagram, in the editor window named *Result*. This is a new diagram that has been created from the two compared diagram versions.

The second Merge dialog does not only have previous and next buttons to navigate between differences. It does also have *Accept/Reject* buttons and "<<" (fast backward) / ">>" (fast forward) buttons.

- Pressing *Accept 1* (*2*) will add symbols to the merge result diagram. It is the symbols from version 1 (2) of the current difference group that will be added.
- Pressing *Reject 1* (*2*) will remove symbols from the merge result diagram. It is the symbols that can be found in version 1 (2) of the current difference group that will be removed.
- Pressing "<<" (fast backward) will navigate to the previous unresolved difference group. An unresolved difference group is a difference group where both version 1 and version 2 are not accepted.
- Pressing ">>" (fast forward) will navigate to the next unresolved difference group.

Note that it is possible to manually edit diagrams while a merge operation is ongoing. This can be useful if you want the merge result diagram to look different from both version 1 and version 2 of the compared diagrams.

When the second merge dialog is closed, the editor will ask you to save the merge result by displaying a save dialog above the window with the

merge result diagram buffer. When the save dialog is closed, the editor windows are restored to how they appeared before the merge operation was invoked.

GR to PR Conversion

It is possible to convert diagrams from SDL/GR to SDL/PR. In the conversion, SDL diagrams are translated from the binary files into PR files, possibly containing CIF information. All graphical symbols in that diagram will be replaced by their corresponding keywords and any text in the symbols will be copied to the output file. When possible, the definitions in the output file will appear in the same order as in the input.

The conversion is used by the SDL Analyzer as the first step in the analysis process, and also by the SDT2CIF Converter to convert to PR that preserve the graphical layout information. You can also convert to PR by the *Generate PR* command in the *Tools* menu, or by using the Public Interface (see "GRPR" on page 626 in chapter 12, *The Telelogic Tau Public Interface*).

The PR/CIF files produced by the SDL suite fill various purposes:

- As a way to export information to other SDL tools
- As a temporary storage format that is used as input to the static and semantic analyzer
- As a format that is suitable for reading by the user. To produce this format, the PR file should be pretty printed.

Mapping between GR and PR

The following list defines the mapping between the GR and PR forms:

- In general, when symbols of the same type are converted, symbols are ordered first of all by the page ordering within the diagram. Within the same page symbols are ordered such that a symbol with a low y-coordinate (closer to the top) will be converted first. If two symbols have the same y-coordinate the symbol with the lowest x-coordinate is treated first.

Note:

The **order in which text symbols are converted may be of importance** when it comes to code generation issues. This is, in particular, applicable when using the SDL to C Compiler. Therefore, you should use one text symbol only for type definitions.

- When analyzing systems using macros, all macros are converted to PR before they are expanded. This restricts the usability of macros, for instance the number of inlets and outlets must be 0 or 1.
- A merge symbol (a flow line connected to another flow line) is translated to PR using join and label constructs. The labels introduced will be named *grst0*, *grst1*, and so on, starting on *grst0* at each diagram. In macro diagrams, the labeling is *grst0%MACROID*... If the label already exists in the GR diagram, the sequential number is incremental until the label is unique.
- A block diagram may contain a block substructure diagram without its frame and heading if the block diagram does not contain any definitions but the substructure. This rule is applied to all the pages in the block diagram that are of the block interaction type.
- When specifying a header in GR, it must be specified according to the PR syntax and the valid input signal set should be written into the heading symbol and not in a text symbol as specified by the GR syntax.
- References to graphical symbols, which will be used by the Analyzer in error messages, are stored in the PR file using SDL comments beginning with the character "#". It is therefore recommended that you do not write comments beginning with "#".
- To be able to perform a correct conversion of SDL/GR *Substructure* diagrams to SDL/PR, the enclosing diagram type, block/channel or block type, must be known. For a channel going to the environment, a name outside the frame must be a gate name, generating the `via` construct, if the enclosing diagram is a block type, and a channel name, generating `connect` statements, for other cases.

  If nothing is known about the enclosing diagram type, it is assumed that the diagram is a block.

  To resolve the ambiguity, a qualifier can be used in the kernel heading to determine the enclosing diagram type. Alternatively the An-

alyzer directive #BLOCKTYPE can be used to indicate that the enclosing diagram is a block type diagram. This directive should be added to the diagram kernel heading.

Differences when generating CIF

In general the PR generated as input to the analyzer will be the same as generated for CIF but for some constructs there are differences due to language limitations or to get a better error detection.

- The conversion of a graphical comment symbol differs dependent on if the translation is made to CIF or not. When converting to CIF, the PR generated is the `<comment>` PR construct; in other cases, including when performing Analysis, the graphical comment is converted to "/* text inside graphical comment symbol */".

- The SDL editor supports the possibility to place the comment symbol without connecting it to another symbol. When generating PR without CIF this comment will be converted to PR as described above. When generating CIF the CIF comment will be
  /* CIF Comment (500,100) Right */
  /* text in unconnected comment */
  This standalone nonstandard CIF construct will be recognized by the CIF2SDT application such that the original comment symbol will be restored in the diagram.

- In the SDL editor you can connect many comments to the same symbol. This is an extension to the SDL language. For this graphical construct there is no immediate translation to PR. When generating PR without CIF the comments are generated according to the note above and there will be no problems. However, when generating CIF the comment will be generated using the <comment> PR construct. As there only can be one <comment> to each symbol there is no direct possible translation in PR for many comments. In this case the extra comment symbols are handled as if they were unconnected to the symbol. This means that when regenerating a diagram from the CIF code, the diagram will not be exactly the same as the original diagram as the extra comments will now be unconnected.

- When generating non-CIF all unattached comments will be attached to a symbol that is closest to the comment. In this way the comment text will be generated as if the text belongs to this symbol.

- There is a difference also for graphical connections. If a channel C1 has a graphical connector to the outer channel outerC and another channel C2 also has a graphical connector to outerC, but the points where outerC is drawn is not the same, the conversion for CIF will contain two connect statements:

  ```
  /* CIF ... */
  connect outerC and C1;
  /* CIF ... */
  connect outerC and C2;
  ```

  This PR will produce a semantic error according to SDL-92, but will be correct when using SDL-96.

  When using non-CIF conversion, the two graphical connections will be combined into one statement, provided that the two texts specifying outerC are lexically the same, thus writing outerC in one text and out_\nerC in the other (\n is a newline) will not be treated as the same text. They are still considered the same if only the case differs.

  ```
  connect outerC and C1,C2;
  ```

- When PR is generated for a macrodefinition the macro inlet and macro outlet symbols will differ whether CIF is generated or not. As there are no corresponding PR to these symbols nothing will be generated when generating CIF except of course for the CIF statements itself. For non-CIF a note will generated indicating that these symbols are present in the diagram.

  ```
  /* Macro Inlet Symbol  */
  ```

  A task having no text or having only a note/comment is allowed in GR. The correct PR correspondence is the compound statement construct having an empty statement

  ```
  task { };
  ```

  As this PR will restrict the use of the simulator command Next-Symbol the PR generated for non-CIF is an empty informal text

  ```
  task '';
  ```

  For CIF the compound statement will be generated.

- CIF generation for the class symbol is not supported. The generation for non-CIF is described in "Class Information" on page 1876.

Error and Warning Message

This section contains messages that may be produced during the GR to PR conversion.

When the Telelogic Tau Public Interface performs the conversion from GR to PR, the messages will be displayed in the Organizer log. The reply message on the service will indicate that the conversion has been done and it will also contain the number of errors produced.

If the conversion for some reason fails, the reply message will contain a text stating the kind of problem that has occurred. See further the *GRPR* service in the Public Interface.

ERROR 5001 Unexpected end of flow

This message indicates that a symbol has been found that has no following flowline and according to the SDL syntax there must be one. In addition to the error message the following line is generated in the PR:

```
***** Unexpected end of flow *****
```

WARNING 5002 More than one line entering macro outlet

This message indicates that inside a macro there are more than one line entering the macro outlet symbol. This situation should be avoided as this can lead to syntax errors (which are hard to track) if this macro is expanded inside decision branches.

The easiest way to avoid this situation is to add a no-action task symbol with the text ‘’ and use this as the last symbol in the macro. Let all lines enter this task symbol and add a single line between this task and the macro outlet.

Chapter 44 Using the SDL Editor