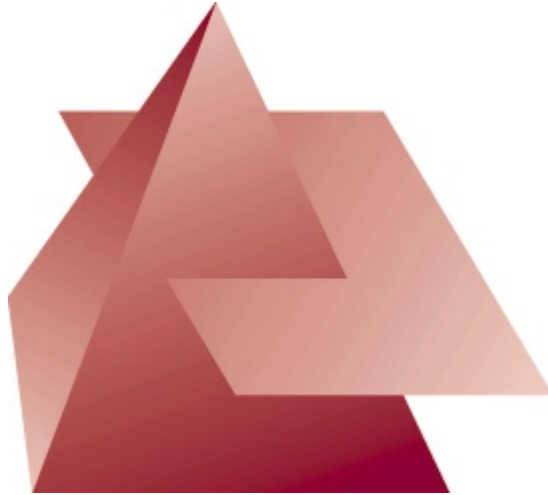


**CASEDOK™ DEVELOPER for COOL:Gen**  
**Version 4.0**



This manual has been created with the **CASEDOK™ DEVELOPER** workstation.

© Copyright 1994-2001 by Deron AG. All Rights Reserved. CASEDOK is a trademark of Deron AG. COOL:Gen is a trademark of Computer Associates International, Inc. Windows 95/98/NT/2000 and Internet Explorer are trademarks of Microsoft Corporation, Netscape is a trademark of Netscape Communications Corporation.

# Table of Contents

- 1 Contact Deron**
- 2 What's new in version 4.0**
- 3 What is CASEDOK DEVELOPER?**
- 4 Installation and Configuration**
- 5 How to edit a report**
  - 5.1 Report Editor*
  - 5.2 Node Command Menu*
    - 5.2.1 Conflict Situations
  - 5.3 Toolbox*
    - 5.3.1 Markup Window
    - 5.3.2 Context Window
    - 5.3.3 Property Window
- 6 How to run a report**
- 7 Getting started: Sample report**
  - 7.1 Sample Report: Create new report*
  - 7.2 Sample Report: Add a title page*
  - 7.3 Sample Report: Save & Run report*
  - 7.4 Sample Report: Add Entity properties*
  - 7.5 Sample Report: Add Relationship List*
  - 7.6 Sample Report: Add Index Entries*
  - 7.7 Sample Report: Add specific HTML Properties*
- 8 Reference**
  - 8.1 Managing Reports*
  - 8.2 Editing Reports*
    - 8.2.1 Contexts
      - 8.2.1.1 Divisions
      - 8.2.1.2 Cast Nodes
      - 8.2.1.3 Ordering
      - 8.2.1.4 Conditional Execution
    - 8.2.2 Properties
      - 8.2.2.1 Matching Patterns
    - 8.2.3 Markup Actions
      - 8.2.3.1 Comment
      - 8.2.3.2 Demoted Section
      - 8.2.3.3 Heading
      - 8.2.3.4 Hypertext Link (HTML only)
      - 8.2.3.5 Index Entry
      - 8.2.3.6 Library Link (HTML only)
      - 8.2.3.7 Link
      - 8.2.3.8 List
      - 8.2.3.9 List Item
      - 8.2.3.10 Page Break (RTF only)
      - 8.2.3.11 Paragraph
      - 8.2.3.12 Report
      - 8.2.3.13 Script Node
      - 8.2.3.14 Series
      - 8.2.3.15 Subroutine
      - 8.2.3.16 Table
      - 8.2.3.17 Table Cell
      - 8.2.3.18 Table Row
      - 8.2.3.19 Text
      - 8.2.3.20 Text Style
    - 8.2.4 Selecting and Sorting
      - 8.2.4.1 Collection
      - 8.2.4.2 Expression
      - 8.2.4.3 Variable
    - 8.2.5 Iterators
      - 8.2.5.1 Report Library
    - 8.2.6 Macros

8.2.6.1 User Macros

8.2.6.2 System Macros

8.2.6.3 Report Parameters and Subroutine Call Arguments

*8.3 Running Reports*

**9 Exhibit: Metamodel views**

## 1 Contact Deron

**Address:** **Deron AG**  
Webergutstrasse 4  
3052 Zollikofen  
Switzerland

**Phone:** +41 31 910 44 55

**Fax:** +41 31 910 44 49

**Email:** [info@deron.com](mailto:info@deron.com)

**Internet:** [www.deron.com](http://www.deron.com)

**In USA and CANADA, contact our partner:**

**Address:** **CANAM Software Labs, Inc.**  
200 Matheson Blvd. West Suite 103  
Mississauga, Ontario  
Canada L5R 3L7

**Phone:** (905) 712-8985

**Fax:** (905) 712-0043

**Email:** [support@canamssoftware.com](mailto:support@canamssoftware.com)

**Internet:** [www.canamssoftware.com](http://www.canamssoftware.com)

## 2 What's new in version 4.0

The following changes occurred since version 3.0:

- Support for COOL:Gen 6.0 schema release 9.0.A2
- Long, self speaking report definition file names
- Support for the organization of report definitions in any folders
- More flexible definition of subroutines
- Improved support for headings of any level

### Release History

#### *Version 3.0*

- Report Parameters introduced
- New possibilities for selection and sorting of any parts of the report based on flexible expressions
- Property text extraction capabilities with powerful string matching
- Improved, pixel-exact GUI reports

#### *Version 2.2*

- Support for COOL:Gen version 5 and the metamodel 8.0.A3
- Year 2000 compliance

## 3 What is CASEDOK DEVELOPER?

**CASEDOK DEVELOPER** is a powerful workbench that allows you to create high-quality documentation based on the COOL:Gen repository database. The tool provides access to the full range of metamodel objects. You can get any single piece of information from encyclopedia and put it in your document. Hereby, the document may be a text file in Rich Text Format (RTF), a hypertext file such as Windows Help or a HTML document. CASEDOK DEVELOPER provides a metamodel browser which displays a hierarchical view of the metamodel and allows you to understand how objects are related and what properties are available. The metamodel hierarchy is shown in a tree view, which enables you to browse the repository. You can enter the metamodel space by expanding division nodes or searching for an object name. When visiting an object, you can see the properties of the object and all incoming and outgoing associations with their cardinality. In addition, you also have access to the properties of objects that you have already visited on your way so far.

CASEDOK Developer provides a rich set of markup elements which enable you to define high-quality documents. The set includes elements such as headers, paragraphs, lists, series, tables, texts, text variables, index entries, chapter numbers, hypertext links, character formatting, macros, graphics (\*.bmp, \*.gif) and more. You can even embed native RTF or HTML code in your document profile. Especially for web documents, CASEDOK Developer provides a wide range of HTML features. This allows you to present your encyclopedia information in a very attractive form on the internet/intranet.

The CASEDOK Developer workstation also includes the **CASEDOK RUNNER** workstation. This allows you to produce the document "at your fingertips". Any time during development of a document, you can press the "Run" button and you will immediately see the changes you made. The integrated **CASEDOK WALKER** tool allows you to step through a model and see where and how model data is stored. You can open as many WALKER's as you need at the same time and on different models.

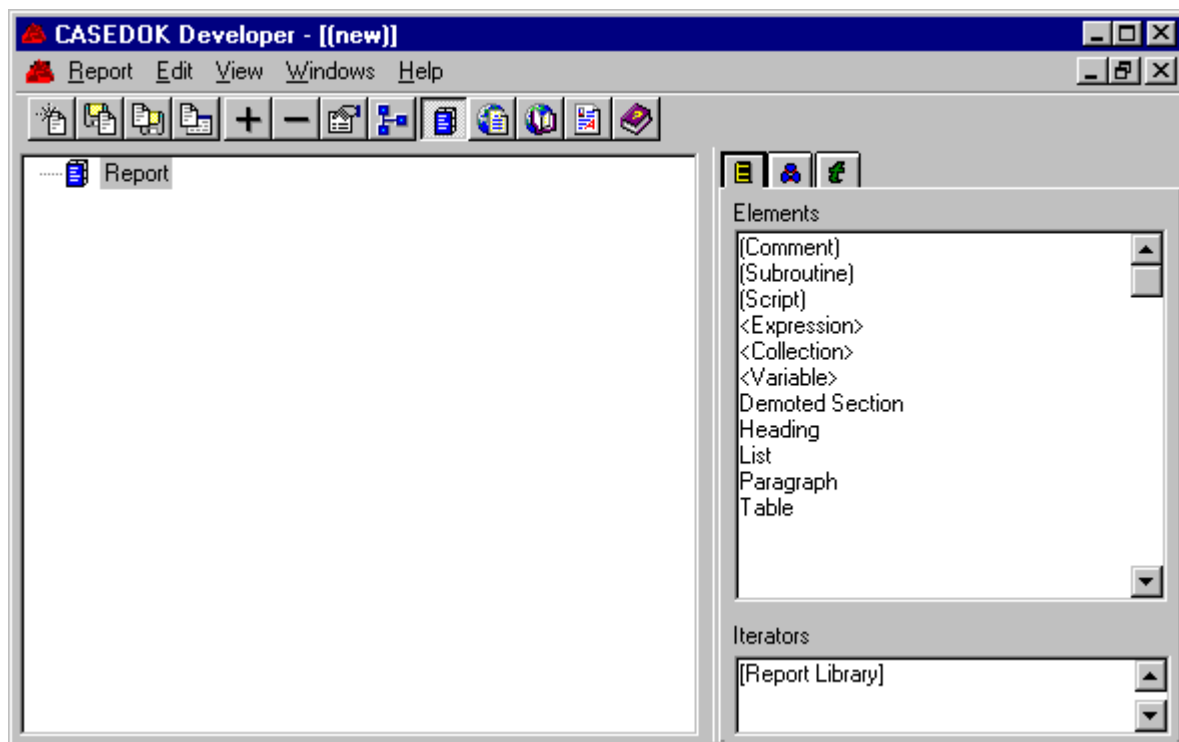
Document profiles created with **CASEDOK DEVELOPER** can be used by **CASEDOK RUNNER** and **CASEDOK WebEXPLORER**. CASEDOK WebExplorer enables repository access on the internet/intranet. This service allows you to navigate directly in your COOL:Gen encyclopedia using your favorite webbrowser.





## 4 Installation and Configuration

The installation of CASEDOK products is described in the CASEDOK RUNNER manual. Please refer to chapter [Installation and Configuration](#) in the **CASEDOK RUNNER User's Guide**.

## 5 How to edit a report

Select <DEVELOPER> from the Windows Start menu to launch the application.

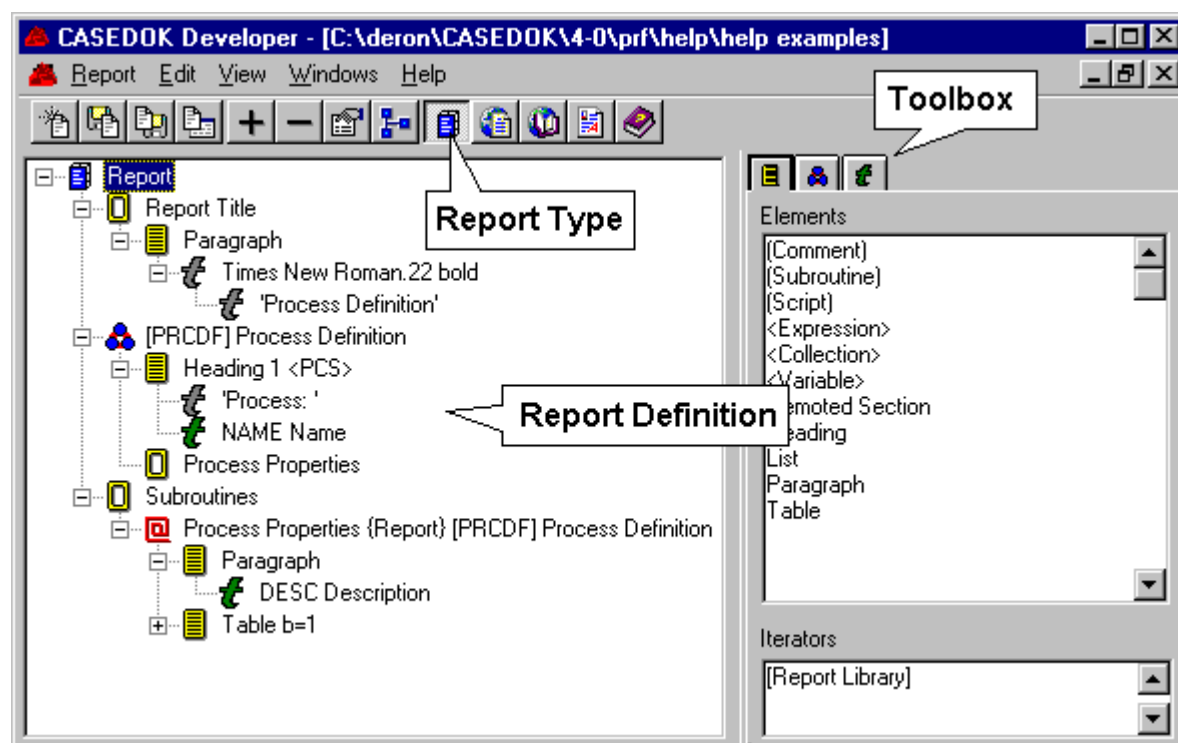


Menu	Items
Report	 <p>Report management functions: New, Open, Save, SaveAs, Close, Properties, Exit (see <a href="#">Managing Reports</a>).</p>
Edit	 <p>Tree view functions: Expand, Collapse. These functions are used to expand and collapse tree nodes in the <a href="#">Report Editor</a> window.</p>  <p>Report type: Generic, HTML (bulk), HTML (paged), Richtext or Winhelp mode. 'Generic mode' provides elements available for all document formats. Other modes provide additional elements or properties specific for the format (e.g. background color in HTML). Different report type specific enhancements may coexist in a report. During execution, the most specific definition is used.</p>
View	<p>Application view: Chain to related CASEDOK applications. From this menu you can start the <b>CASEDOK RUNNER</b> application or open the 'Settings' window to change the configuration parameters. For configuration refer to chapter <a href="#">Installation and Configuration</a> in the <b>CASEDOK RUNNER User's Guide</b>.</p>  <p>You can also start the <b>CASEDOK WALKER</b> tool which allows you to see real model data.</p>
Windows	Standard windows functionality to select and arrange windows. Use this menu to switch between open <a href="#">Report Editor</a> window(s).
Help	Online Help system for CASEDOK DEVELOPER and License information ('About').

### 5.1 Report Editor

You can edit a new report or open an existing report definition file. The structure of the report is displayed in a tree view, where nodes can be expanded and contracted. Use the **right mouse button** to open the [Node Command Menu](#). The [Toolbox](#) on the right side of the window provides context-sensitive actions. The system only shows actions available for the report node, which currently has focus. The toolbox allows you to add markup elements and metamodel information to your report.

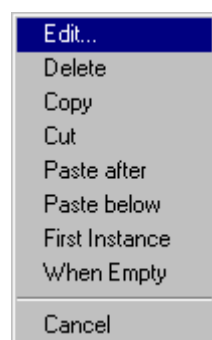
Refer also to 'Editing Reports' for detailed information. To display multiple report windows, use the <Windows> menu commands.



Changing the **Report type** allows you place additional format-specific elements or properties. Example: In HTML mode you can specify additional table properties such as background colors or border width. Format-specific properties can coexist in a report. During execution in CASEDOK RUNNER, the most specific definition is used. If there are no specific enhancements specified, the 'Generic mode' definition is used.

## 5.2 Node Command Menu

If the focus is on a report node, you can open the node commands menu by pressing the **right mouse button**. This context menu contains node commands, which allows you to manipulate the report structure or define node values. Some of the node commands are sensitive to the node context. Commands not available in a certain context are disabled.



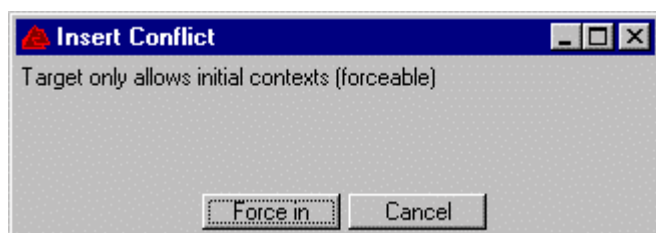
The following node commands are available:

Command	Description
Edit...	If available, the edit command opens the property dialog of the selected node. The dialog depends on the node type and the report type currently in effect. Not all node types have properties.
Delete	Deletes the selected node and its subtree.

Copy	Copies the node and its subtree to the clipboard.
Cut	Removes the selected node and its subtree from the report and puts it on the clipboard.
Paste after	Inserts the node or subtree from the clipboard after the selected node and its subtree. Do not mix up with 'paste below'. In case of a conflict, see <a href="#">Conflict Situations</a> .
Paste below	Inserts the node or subtree from the clipboard as first child below the selected node. Do not mix up with 'paste after'. In case of a conflict, see <a href="#">Conflict Situations</a> .
Each/First Instance	The nodes in a context with 0-N or 1-N cardinality can be given the 'First Instance' attribute. The <a href="#">Report Editor</a> displays these nodes with a ' ' character in front. Nodes having this attribute are executed only for the first object iterated by the enclosing context. The remaining nodes always must form a coherent core; you can only mark nodes at the beginning or end of the core, or enclosing the core. Select 'Each Instance' from the context menu to remove the attribute. See also chapter <a href="#">Conditional Execution</a> .
When Empty/Instance	The first node in a context with 0-1 or 0-N cardinality can be given the 'When Empty' attribute. The <a href="#">Report Editor</a> displays these nodes with a '*' character in front. The node and its tree are only executed when there are no objects matching the enclosing context type (similar to an else branch). Select 'When Instance' from the context menu to remove the attribute. See also chapter <a href="#">Conditional Execution</a> .

### 5.2.1 Conflict Situations

Conflict situations may occur when you try to insert report nodes or subtrees to an incompatible context (see **Paste after/below** in the [Node Command Menu](#)). There can be combinations of incompatible markup, context and property nodes. In conflict situations, the 'Insert Conflict' window appears:



If you force in incompatible nodes, they are converted to comment nodes. If there are compatible as well as incompatible nodes in the insertion block, the compatible nodes keep their type. Force in can be a fast way to copy existing node trees to a new context. But it only make sense when incompatible nodes are in lower branches of the node tree to be inserted.

Here are some of the most frequent conflict situations:

Conflict message	Conflict situation
Incompatible markup	You try to insert a markup node below an other, incompatible markup node (e.g. paragraph below paragraph).
Incompatible context types	You try to insert a context node below an other, incompatible context node (e.g. <Described by Attribute Type> below Subject Area).
Initial context not allowed in other contexts	You try to insert an initial context node (top level) below an other context node (e.g. Attribute Type below Subject Area).
Property does not exist in the new context	You try to insert a property node below a context node that does not have this property type (e.g. 'Elementary Process Indicator' below Function).
Property requires textual context	You try to insert a property node below a context node that has no markup (e.g. 'Elementary Process Indicator' below Process outside a paragraph, table node or list item).
Target only allows initial contexts (forceable)	You try to insert a context subtree on top level. If you force in, the context is changed into an initial context (e.g. <Described by Attribute Type> to Attribute Type).

## 5.3 Toolbox

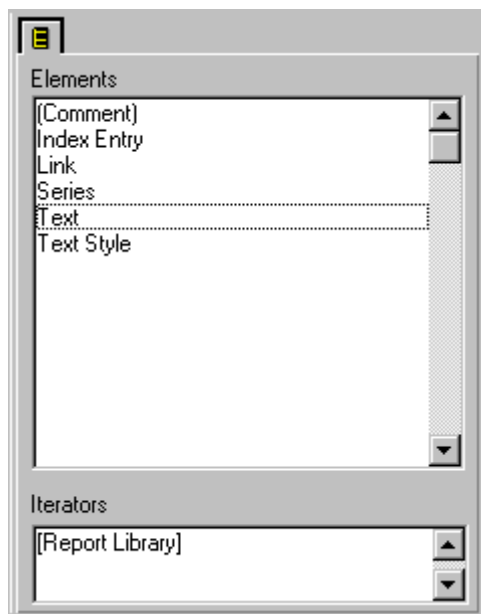
The toolbox consists of three windows: [Markup Window](#), [Context Window](#) and [Property Window](#). The contents of these windows are context-sensitive to the currently selected report node. Changing the focus in the report tree will dynamically update the contents of the toolbox windows.

### 5.3.1 Markup Window

The **Markup Window** shows the [Markup Actions](#) currently available for the selected report node. If the focus in the report is on:

- a markup node, markup actions which can be placed below it, are displayed.
- a context node, markup actions which can be placed below the enclosing markup node, are displayed.
- a property node, markup actions are not available.

**Double-click** on the desired markup element to insert it **below** the selected report node (as last child).

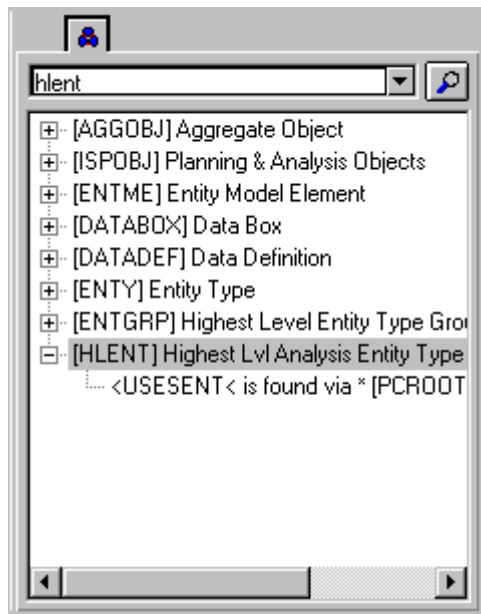



### 5.3.2 Context Window

The **Context Window** shows the [Contexts](#) currently available for the selected report node. A context can be seen as a loop in a program, iterating over all objects of its type. Inside the context, properties and associations of the context's type are available. If the focus in the report is on:

- a markup node, contexts available inside the enclosing context node are displayed.
- a context node, contexts available inside it are displayed, that is, object types reachable from its own type
- a property node, no contexts are available


**Double-click** on the desired context to insert it **below** the selected report node (as last child).




 Use the search function to find an object type by name or mnemonic. Press the search button to look for the next occurrence of your search string. The search function is a circular search applied to the contents of the context window, which is the full range of metamodel objects on top level or the range of accessible objects in the context of an object. The search string is not case-sensitive, wildcards are not supported.

On top level, the Context Window shows all objects available in the metamodel. The objects are organized in divisions and subdivisions. When the report focus is on a context node, the associated objects of this node are displayed. Associations are displayed with name, mnemonic and cardinality information. In the report structure, the cardinality is visualized as follows:

Example: 0/1/N -- 1

 <DETLBY< detail of [SUBJ] Subject Area

Example: 0/1/N -- 0/1

 >DERIVDBY> is derived by \* [ACBLKBAA] BAA Action Block

Example: 0/1/N -- N

 >DSCBYR> described by 0-N [RELMM] Relationship Membership

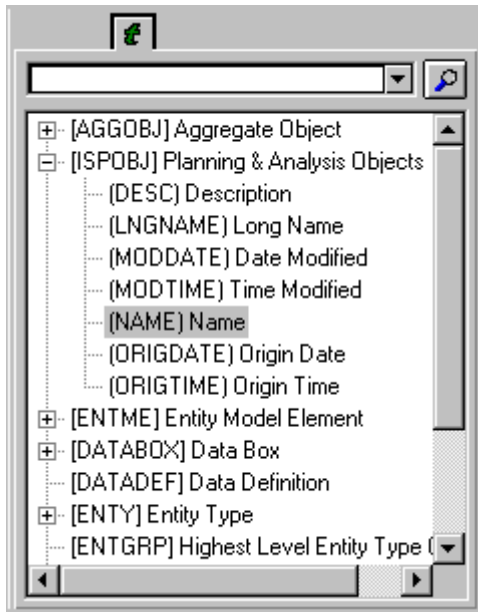
### 5.3.3 Property Window


The **Property Window** shows the [Properties](#) currently available for the selected report node.

If the report focus is on


- a markup node, properties of the enclosing context node are displayed. If there is no enclosing context or the markup node can not include properties (e.g. Table), the property window is empty.
- a context node, properties are only displayed when there is an enclosing markup element. If there is no enclosing markup node or the enclosing markup node can not include properties (e.g. Table), the property window is empty.
- a property node, properties are not available.

**Double-click** on the desired property element to insert it **below** the selected report node (as last child).



 enclosing context

Besides the properties of the current context, you have also access to the properties from all enclosing contexts. This allows you to combine information from different context levels in one markup element. Properties from outer contexts are referenced with the context name and level indication (e.g. \$.HLENT.NAME).

 Use the search function to find a property type by name or mnemonic. Press the search button to look for the next occurrence of your search string. The search function is a circular search applied to the contents of the property window, which is the full range of properties of the current context and all enclosing contexts. The search string is not case-sensitive, wildcards are not supported.

## 6 How to run a report

Reports can be executed with the CASEDOK RUNNER module. As soon as you have saved a report. You can immediately use the report in CASEDOK RUNNER. If the report is already part of a documentation job in RUNNER, the job is refreshed automatically.

To start the RUNNER module, select <**RUNNER**> from the Windows Start menu or from the DEVELOPER's 'View' menu.

For more information, refer to chapter ['How to run a report'](#) in the **CASEDOK RUNNER User's Guide**.

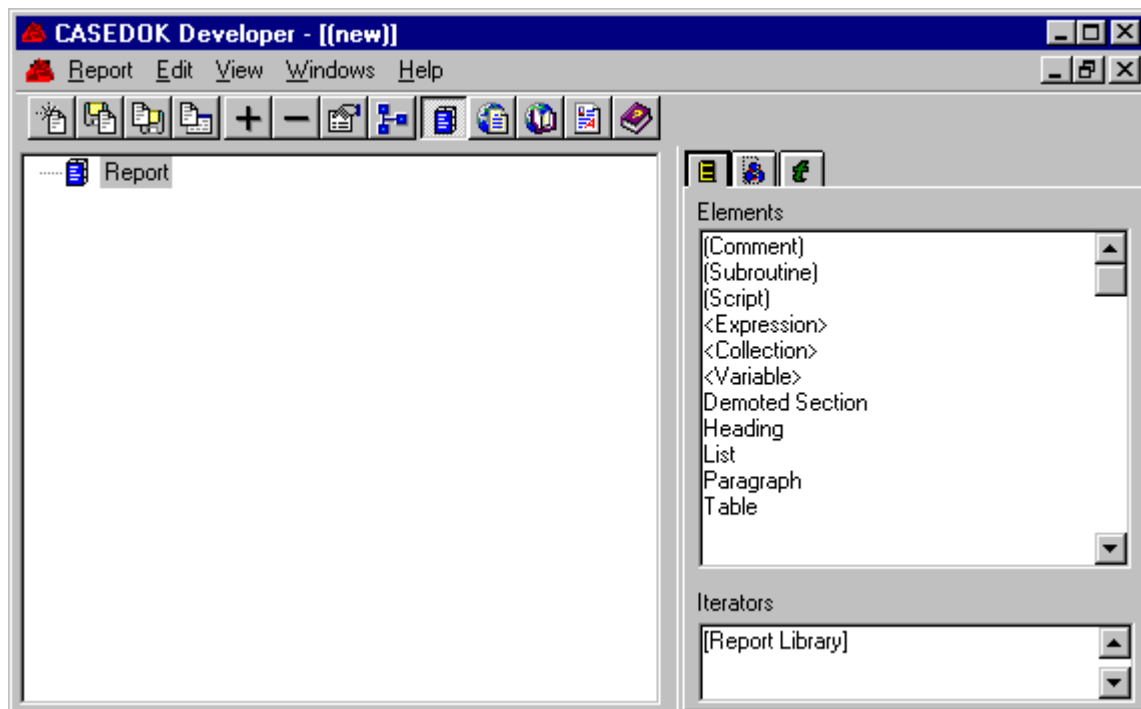
## 7 Getting started: Sample report

This chapter illustrates how to create a new report from scratch. A good way of learning how to work with DEVELOPER is to reproduce this sample report. Of course, this report does not show all the features of the tool. If you want to understand more complex reports, look at the standard reports delivered with the RUNNER module.

Usually, when defining a new report, you do not start from scratch. Often, you will copy an existing report and adapt it to your needs. As well, you will **reuse** existing **report components** in other reports. This is a kind of 'Component Based Development', where a component can be encapsulated in a comment node. A component can only be reused when it fits a target context. Pure markup components are most easy to reuse. A good example for a component with markup, context and property information, is 'Views', used in various standard reports.

### 7.1 Sample Report: Create new report

 Start CASEDOK DEVELOPER and open a new report.



### 7.2 Sample Report: Add a title page

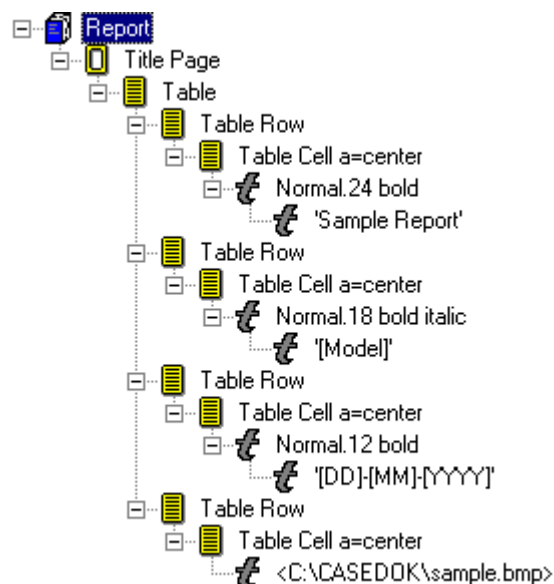
A title page will be created including a report name, the model name, the current date and a sample bitmap. For inclusion of model name and current date, system macros will be used.

**Covered features:** Comment, Table, Text Style, Text, System Macro, Bitmap, Edit, Copy & Paste


Step	Report focus on	Toolbox/Cell command
1.	<Report>	Activate Markup Window; double-click <Comment>; type <i>Title Page</i>
2.	<Comment> ('Title Page')	Double-click <Table>; no gridlines
3.	<Table>	Double-click <Table Row>
4.	<Table Row>	Double-click <Table Cell>; text alignment=center
5.	<Table Cell>	Double-click <Text Style>; enable <bold>-option, set size=24
6.	<Text Style> ('normal.24 bold')	Double-click <Text>; type <i>Sample Report</i>

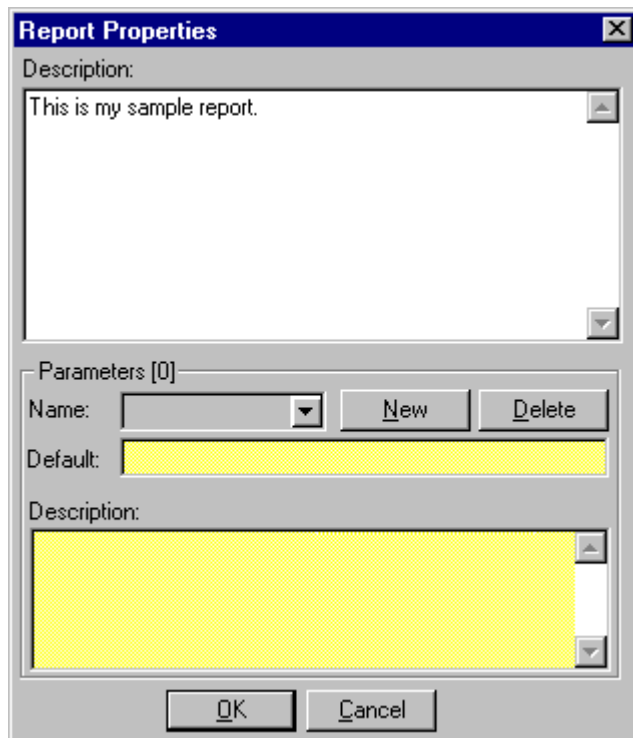
7.	<Table Row>	Select 'Copy' from the Node Context Menu (right mouse button)
8.	<Table Row>	Select 'Paste after' from the Node Context Menu (right mouse button)
9.	<Table Row> (second)	Expand Table Row node
10.	<Text Style> ('normal.24 bold')	Select 'Edit' from the Node Context Menu (right mouse button); enable <italic> option, set size=18
11.	<Text> ('Sample Report')	Select 'Edit' from the Node Context Menu (right mouse button); delete existing text and select system macro <i>[Model]</i> from the macro list
12.	<Table Row> (second)	Execute Steps 7-9: 'Copy' -> 'Paste after' -> expand node
13.	<Text Style> ('normal.18 bold italic')	Select 'Edit' from the Node Context Menu (right mouse button); disable <italic> option, set size=12
14.	<Text> ('[Model]')	Select 'Edit' from the Node Context Menu (right mouse button); replace existing text with system macros: <i>[DD]-[MM]-[YYYY]</i>
15.	<Table>	Execute Steps 3-4: <Table Row> -> <Table Cell>
16.	<Table Cell>	Double-click <Text>; select Type=Bitmap (*.bmp); type <directory>\sample.bmp where <directory> is your CASEDOK directory name


The report structure looks as follows:

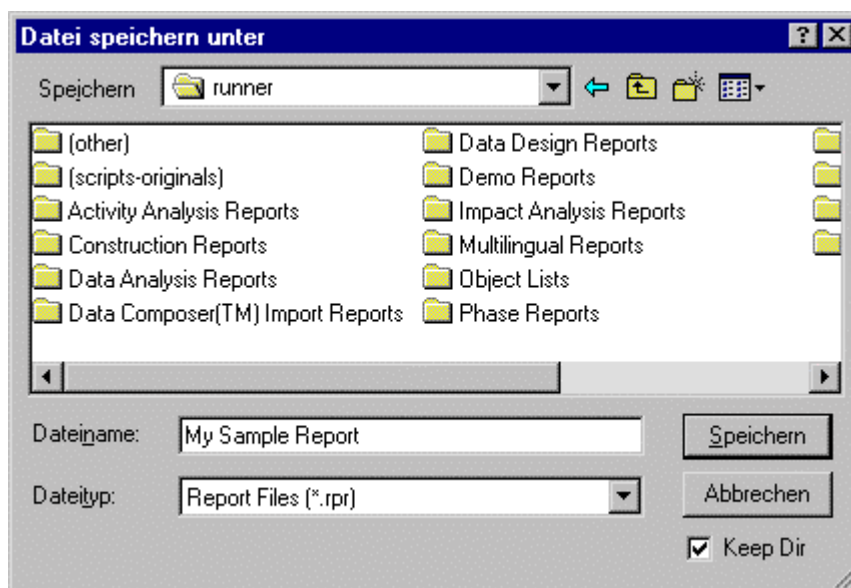


### 7.3 Sample Report: Save & Run report

 Open the report properties and enter a description for your report. This is the description that appears in CASEDOK RUNNER when the report is added to a job.



 Save the report in a report definition file.



Select 'Runner' from the <View> menu and run this sample report using your favorite format(s).

## 7.4 Sample Report: Add Entity properties

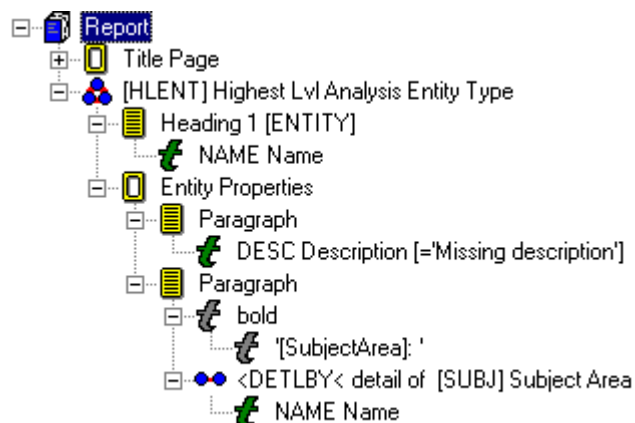
An Entity Type context will be opened including Entity name as heading, a description paragraph with text substitution for missing values and a paragraph with the name of the belonging Subject Area.

**Covered features:** Heading, Paragraph, User Macro, Context, Properties, Property Representation

Step	Report focus on	Toolbox/Cell command
1.	<Report>	Activate Context Window; search for 'HLENT' (mnemonic for Entity Type); when found -> double-click
2.	<HLENT>	Activate Markup Window; double-click <Heading>; level=1, label=ETY

3.	<Heading>	Activate Property Window; double-click <i>NAME</i> from the <i>ISPOBJ</i> division
4.	<HLENT>	Activate Markup Window; double-click <Comment>; type <i>Entity Properties</i>
5.	<Comment> ('Entity Properties')	Double-click <Paragraph>
6.	<Paragraph>	Activate Property Window; double-click <i>DESC</i> from the <i>ISPOBJ</i> division
7.	<DESC>	Select 'Edit' from the Node Context Menu (right mouse button); leave value field empty; type <i>Missing description</i> to the 'Represented by' field; press 'ADD' button
8.	<Comment> ('Entity Properties')	Activate Markup Window; double-click <Paragraph>
9.	<Paragraph>	Double-click <Text>; select [ <i>Subject Area</i> ] from the User macro list; add ' '
10.	<Paragraph>	Activate Context Window; double-click <DETLBY< <i>SUBJ</i> from the <i>ENTGRP</i> division
11.	<DETLBY>	Activate Property Window; double-click <i>NAME</i> from the <i>ISPOBJ</i> division

The report structure looks as follows:



==> Save and run the report using your favorite format(s).

## 7.5 Sample Report: Add Relationship List

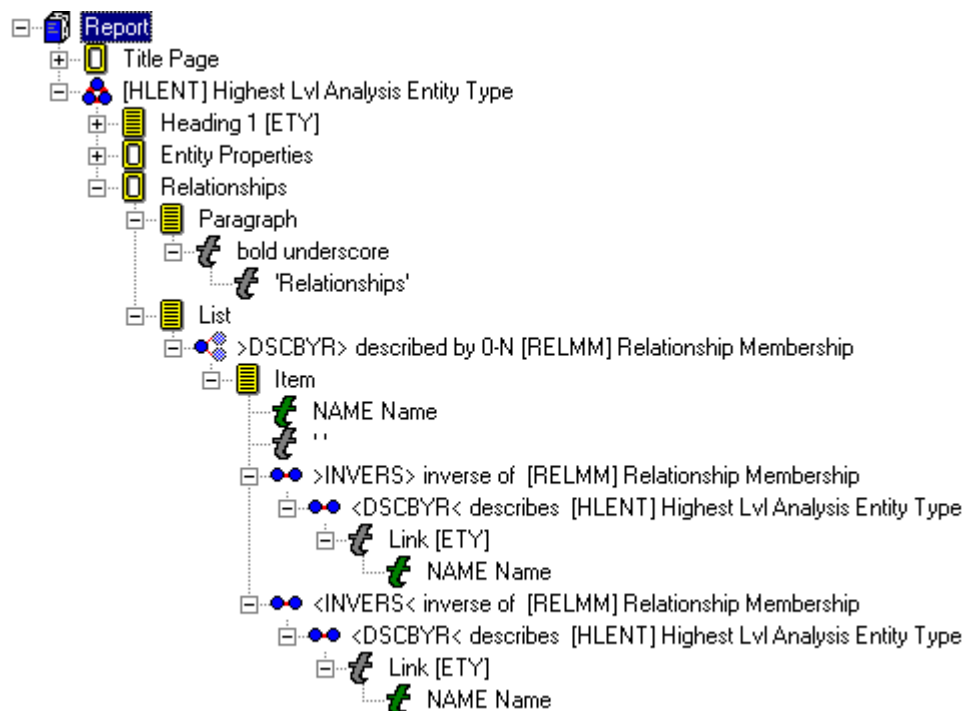
For each Entity Type, a bulleted list of all forward and backward Relationships will be created. For each Relationship Type, the partner Entity will be shown as a link pointing to Entity heading. The list begins with a title.

**Covered features:** List, Link, Copy & Paste Context with Properties

Step	Report focus on	Toolbox/Cell command
1.	<HLENT>	Activate Markup Window; double-click <Comment>; type <i>Relationships</i>
2.	<Comment> ('Relationships')	Double-click <Paragraph>
3.	<Paragraph>	Double-click <Text Style>; enable <bold> and <underscore> option
4.	<Text Style> ('bold underscore')	Double-click <Text>; type <i>Relationships</i>
5.	<Comment> ('Relationships')	Double-click <List>
6.	<List>	Activate Context Window; double-click > <i>DSCBYR</i> > <i>RELMM</i> from the <i>ENTY</i> division
7.	<DSCBYR>	Activate Markup Window; double-click <Item>
8.	<Item>	Activate Property Window; double-click <i>NAME</i> from the <i>ISPOBJ</i> division
9.	<Item>	Activate Markup Window; type ' ' (2 blanks)

10.	<Item>	Activate Context Window; double-click >INVERS> RELMM (forward) from the RELMM division
11.	<INVERS> (forward)	Double-click <DSCBYR< HLENT from the RELMM division
12.	<DSCBYR> (backward)	Activate Markup Window; double-click <Link>; set label=ETY
13.	<Link>	Activate Property Window; double-click NAME from the ISPOBJ division
14.	<Item>	Activate Context Window; double-click <INVERS< RELMM (backward) from the RELMM division
15.	<DSCBYR> (backward)	Select 'Copy' from the Node Context Menu (right mouse button)
16.	<INVERS> (backward)	Select 'Paste below' from the Node Context Menu (right mouse button)

The report structure looks as follows:



==> Save and run the report using your favorite format(s).

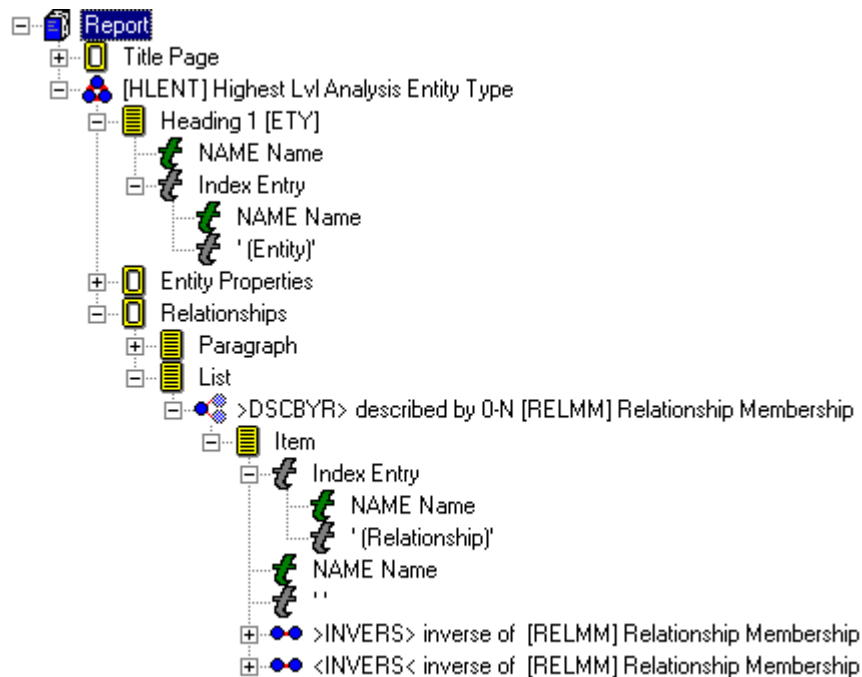
## 7.6 Sample Report: Add Index Entries

The sample report will be enriched with Index Entries for Entity Types and Relationship Types.

**Covered features:** Index Entry

Step	Report focus on	Toolbox/Cell command
1.	<Heading>	Activate Markup Window; double-click <Index Entry>
2.	<Index Entry>	Activate Property Window; double-click NAME from the ISPOBJ division
3.	<Index Entry>	Double-click <Text>; type '(Entity)'
4.	<Index Entry>	Select 'Copy' from the Node Context Menu (right mouse button)
5.	<Item>	Select 'Paste below' from the Node Context Menu (right mouse button)
6.	<Index Entry>	Expand Index Entry node
7.	<Text> ('(Entity)')	Select 'Edit' from the Node Context Menu (right mouse button); exchange existing string by '(Relationship)'

The report structure looks as follows:



==> Save and run the report using your favorite format(s).

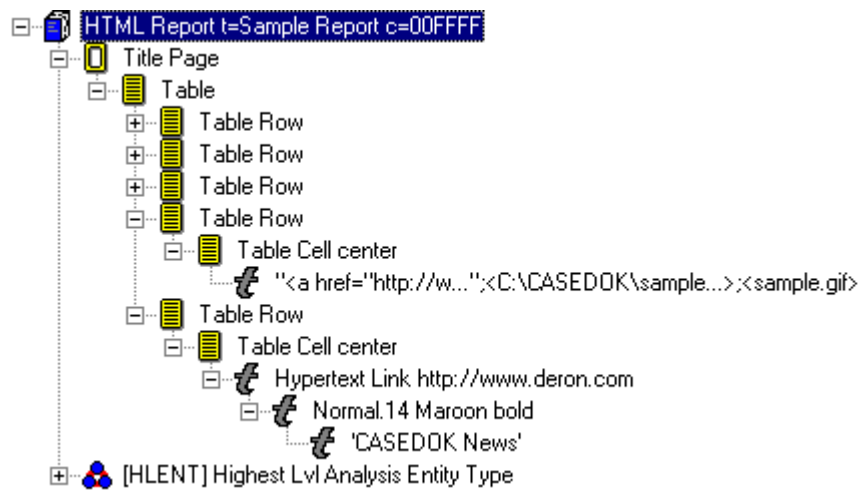
## 7.7 Sample Report: Add specific HTML Properties

The sample report will now be enriched with some specific HTML properties: Report Title, Background color, URL reference and GIF graphic. To see the Gif-graphic in your HTML browser, you must copy the file 'Sample.gif' from your CASEDOK directory to your document directory.

**Covered features:** Report Type, HTML Report Properties, Hypertext Link, Native HTML Code

Step	Report focus on	Toolbox/Cell command
1.	Change Report Type from 'Generic Mode' to 'HTML 3.0 Mode'	
2.	<Report>	Select 'Edit' from the Node Context Menu (right mouse button); set Title=Sample Report; set Background Color RRGGBB=00FFFF
3.	<Text> (Bitmap in Title Page)	Select 'Edit' from the Node Context Menu (right mouse button); select Type=HTML; type <a href="http://www.deron.com"></a>
4.	<Table>	Double-click <Table Row>
5.	<Table Row>	Double-click <Table Cell>; text alignment=center
6.	<Table Cell>	Double-click <Hypertext Link>; URL=http://www.deron.com
7.	<Hypertext Link>	Double-click <Text Style>; enable <bold> option; set size=14; set color=maroon
8.	<Text Style> ('normal.14 Maroon bold')	Double-click <Text>; type CASEDOK News

The report structure looks as follows:



**==> Save and run the report using your favorite HTML browser (Netscape, MS Internet Explorer).**

## 8 Reference

### 8.1 Managing Reports

Report definition files (\*.rpr) can be stored anywhere in the file system. A rich collection of predefined reports is delivered with CASEDOK. These reports are installed in directories within the CASEDOK installation directory. The directories contain reports for various themes. The <Report> menu in CASEDOK DEVELOPER provides the commands to create and manage reports. The commands are also available in the toolbar.


Menu item	Description
New	Creates and opens a new report in the <a href="#">Report Editor</a> . Save the report to give it a name. Open the report properties to enter a description of the report.
Open	Opens an existing report in the <a href="#">Report Editor</a> .
Close	Closes the current report; asks if you want to save any changes.
Save, SaveAs	Saves the current report in a report definition file. The file name you choose is the name of the report. The report is immediately available for use in CASEDOK RUNNER (the list of reports in CASEDOK RUNNER is refreshed when needed).
Properties	<p>Opens the Report Properties dialog where you can enter a description of the report and define report parameters.</p> <p><b>Description:</b> Enter a description of the report. This description is displayed in CASEDOK RUNNER when the report is opened.</p> <p><b>Parameters:</b> <a href="#">Parameters</a> of the report. The parameters are displayed in CASEDOK RUNNER and give a user the ability to specify some behavior when using the report.</p> <p>Hit the 'New' button to create a new parameter. Enter the parameter name, parameter default value and parameter description. Select a parameter from the dropdown list and hit the 'Delete' button to remove it.</p> <p>Parameter values can be accessed in the report exactly like <a href="#">User Macros</a>.</p>
Exit	Closes the DEVELOPER application module. Does not close the RUNNER or WALKER modules.

### 8.2 Editing Reports

The [Report Editor](#) shows a report as a tree consisting of nodes of various types. Nodes describing metamodel objects are mixed with nodes providing markup. The right mouse button opens a [Node Command Menu](#) providing operations to manipulate the tree nodes. New nodes can be added by double-clicking on an element in the toolbox. From the toolbox you can add markups or metamodel information.

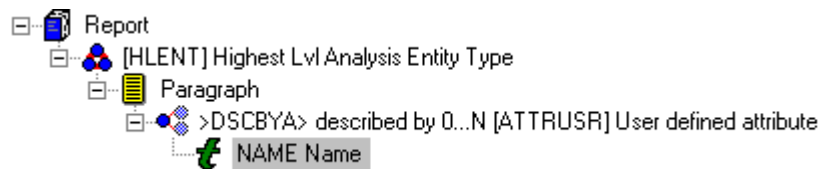
#### 8.2.1 Contexts

Contexts represent the metamodel view of the objects and associations in a model. Contexts are hierarchically organized and grouped into divisions. The [Context Window](#) always shows the contexts reachable from the report node that currently has focus. For some illustrations about the metamodel see [Exhibit: Metamodel views](#).

 Start **CASEDOK WALKER** if you want to see real model data for the selected context.

Contexts can be understood as loops in a program. A context iterates over all objects in a model having the context's type. A nested context iterates over all objects associated with the current object of it's enclosing context. The nodes inside the context are executed for every iteration.

Example:



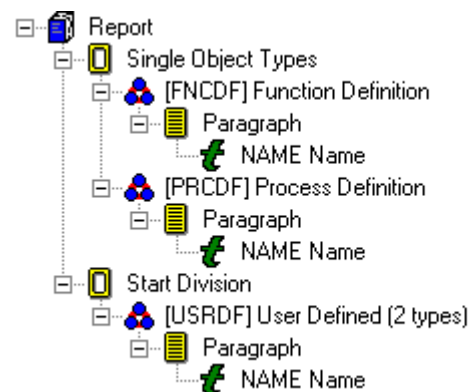
The HLENT entity type context is a loop; the ATTRUSR attribute context is a nested loop. Every entity creates a paragraph; every attribute of the entity writes the name of the attribute to the paragraph.

### 8.2.1.1 Divisions

In the metamodel, object types are grouped into divisions. All object types of a division have the same common properties and are associated to the same object types or divisions. By using divisions, report actions can be executed for a group of objects belonging to different elementary types. The order of the objects in the group is defined for the group as a whole. If some special actions must be defined for a subtype of the division, but the order should go over the entire division, [Cast nodes](#) can be used.

If the division was reached by an ordered association (0..N or 1..N), the order of the objects in the division is defined by the association order. Otherwise, if there is a NAME property in the division, the objects are ordered by name. Otherwise the order is random.


Example:



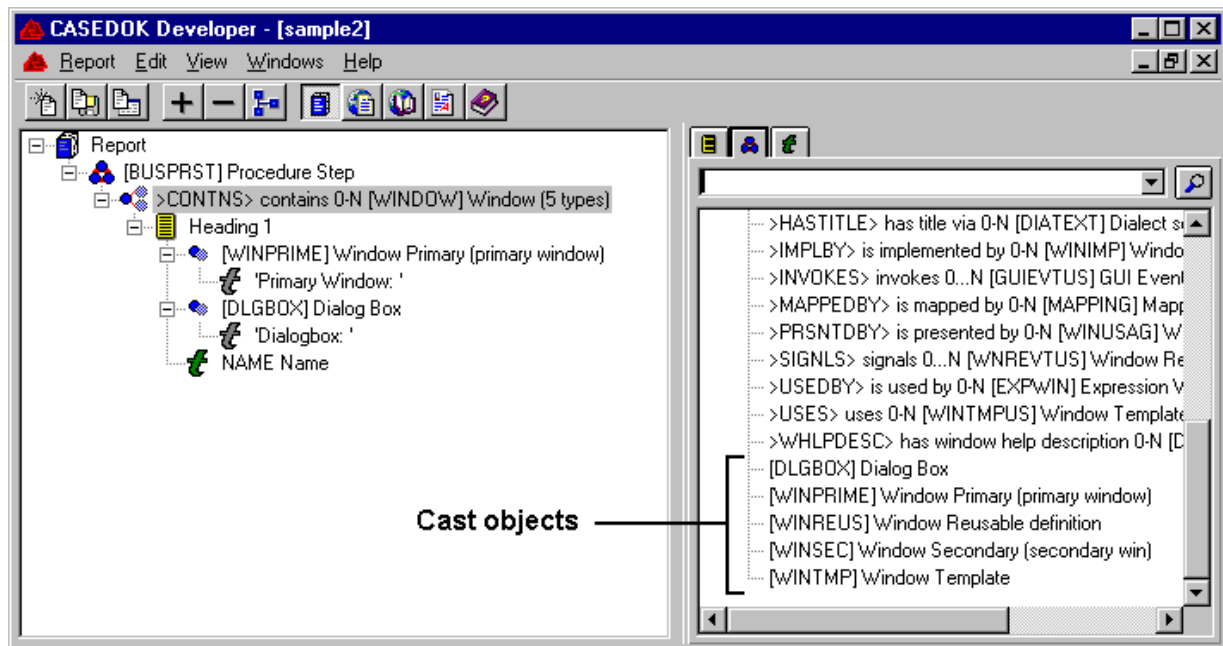
This report produces a list of functions (alphabetical), a list of processes (alphabetical) and a list of function and processes (alphabetical, mixed types).

### 8.2.1.2 Cast Nodes

Cast nodes are used to specialize [division](#) contexts. Within a division, cast nodes can be used for different markup actions depending on the specific object type.

 Cast objects are displayed at the bottom of the context window, if the report focus is on a division context node.

Example:



The shown report produces a heading structure with the name and type information for Primary Windows and Dialogboxes. These two types are selected out of five possible types. The sequence of the headings is random, because of the unordered association (0-N).

### 8.2.1.3 Ordering

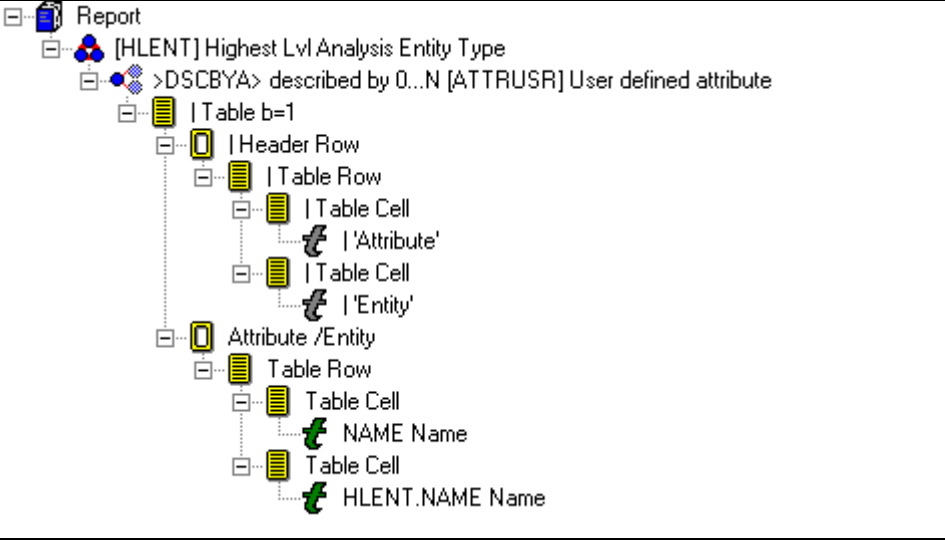
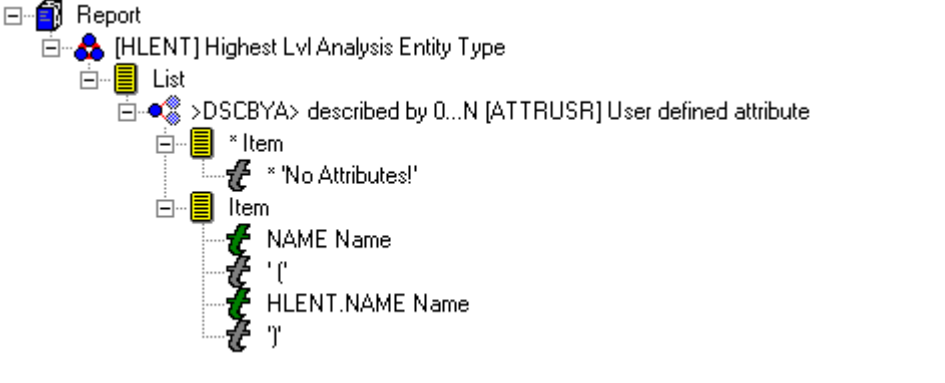
The model objects represented by a context are iterated in the following order:

Initial context	Ordered alphabetically using the NAME property (random order if not present).
Ordered (0..N or 1..N) association	The objects are iterated in the order stored with the association.
Unordered (0-N or 1-N) association	Ordered alphabetically using the NAME property (random order if not present).

### 8.2.1.4 Conditional Execution

Depending on the cardinality of a context's association, the nodes contained in a context can be marked with conditional execution. Open the [Node Command menu](#) to set the appropriate attributes.

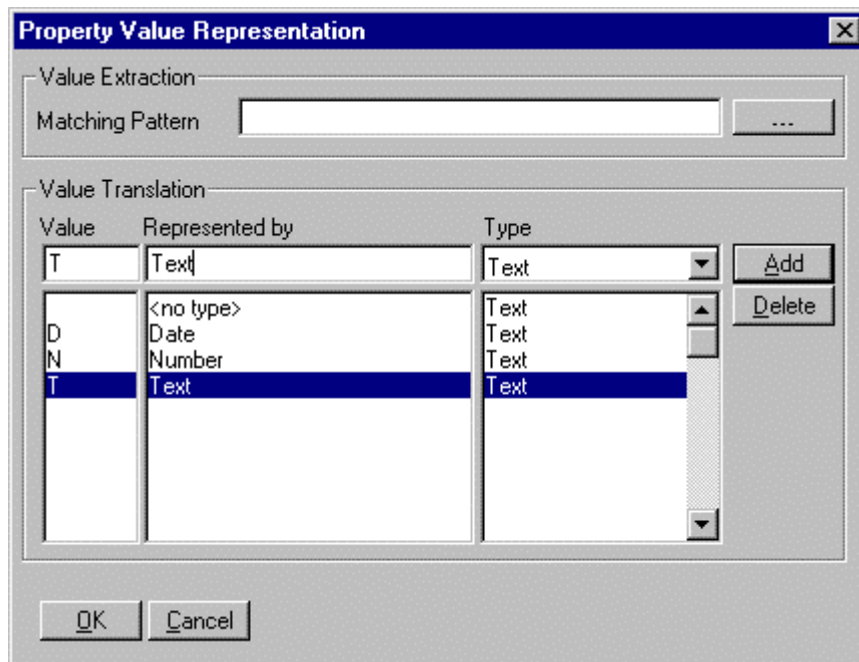
Attribute	Description
not marked	The nodes are executed once for each object of the enclosing context.
First Instance	The nodes in a context with 0-N or 1-N cardinality can be given the 'First Instance' attribute. The <a href="#">Report Editor</a> displays these nodes with a 'I' character in front. Nodes having this attribute are executed only for the first object iterated by the enclosing context. The remaining nodes always must form a coherent core; you can only mark nodes at the beginning or end of the core, or enclosing the core. Select 'Each Instance' from the context menu to remove the attribute.

	
When Empty	<p>The first node in a context with 0-1 or 0-N cardinality can be given the 'When Empty' attribute. The <a href="#">Report Editor</a> displays these nodes with a '*' character in front. The node and its tree are only executed when there are no objects matching the enclosing context type (similar to an else branch). Select 'When Instance' from the context menu to remove the attribute.</p> 

## 8.2.2 Properties

Like [Contexts](#), properties are grouped by divisions. From the [Property Window](#) you can select the properties available in the enclosing context. Properties of outer contexts may be accessed stepwise by opening the 'enclosing context' nodes in the tree.

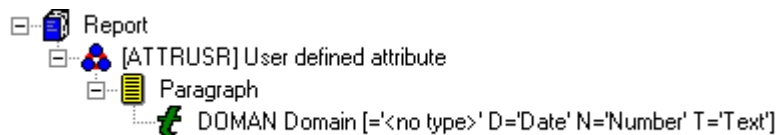
Property values may be transformed into arbitrary representations using the 'Property Value Representation' dialog. Use the 'Edit' item in the [Node Command Menu](#) (right mouse button) to open this dialog.



By supplying a [matching pattern](#), any part of a property's value can be extracted. The matching pattern allows extraction of text marked by tags, appearing on a certain position or following a certain structure. The [...] button shows a dialog which helps you edit and test matching patterns.

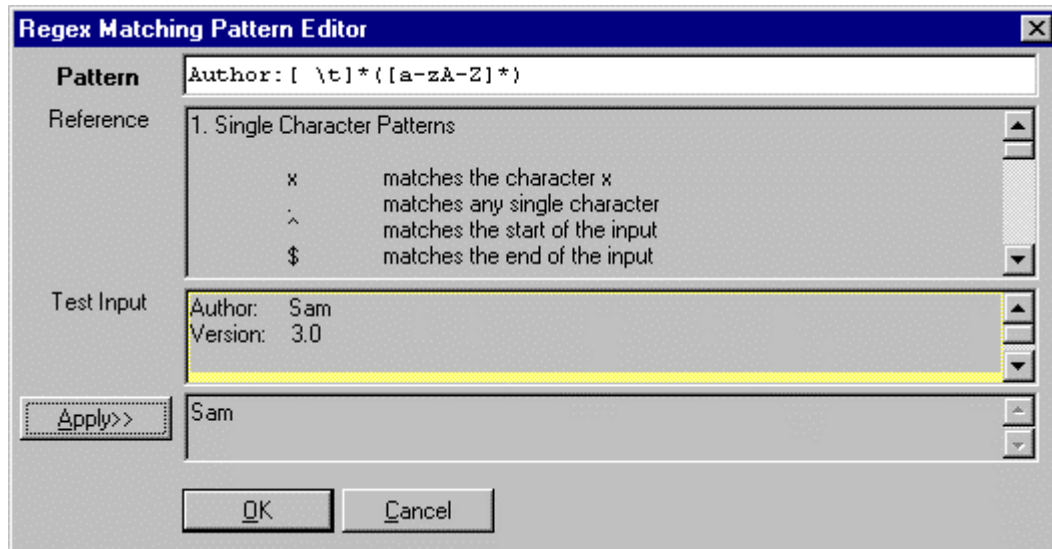
For every possible value the property may have (or results of the value extraction), representations of different types may be specified. See the [Text](#) markup element for a detailed description. [Macros](#) may also be used in the representation text. If the system does not find a supported representation for a value, it adds the value literally to the report.

Example:



### 8.2.2.1 Matching Patterns

Matching Patterns is a powerful method to extract any part of a property value. It is based on Regular Expressions, popular in UNIX tools, the Perl language and other. Select property node context menu <Edit>, 'Property Representation' dialog button [...] to open the 'Matching Pattern Editor':



### Single Character Patterns

x	matches the character x
.	matches any single character
^	matches the start of the input
\$	matches the end of the input
\n	matches a linefeed character
\t	matches a tab character
\0nnn	matches the character with octal code nnn
\x	matches x if x is a special character
[xyz]	matches any of the enclosed characters
[x-z]	matches any of the characters from x to z
^[xyz]	matches any character not enclosed

### Character Strings

PQ	matches P followed by Q
P?	matches zero or one occurrence of P
P+	matches one or more occurrences of P
P*	matches zero or more occurrences of P
P{n}	matches exactly n occurrences of P
P{m,n}	matches m to n occurrences of P

### Substrings

X(P)Y	matches XPY but returns only the input matched by P
-------	---

### Examples

[a-zA-Z]+	returns the first word
^required	returns 'required' if the input starts with this word; nothing else
Author:[ \t]*([a-zA-Z]*)	looks for the 'Author.' tag and returns the word following it, skipping any spaces or tabs

.	returns the first character
.(.*)	returns all except the first character
\n([\n]*)	returns the second line of text

## 8.2.3 Markup Actions

Markup actions are used to structure a report and provide constant text information. The [Markup Window](#) shows the actions available for the selected report node. In a report, markup elements can exist without a metamodel context.

### 8.2.3.1 Comment



Inserts a comment node, consisting of an arbitrary text string. A comment can be used to group nodes which belong together or to make a report profile more readable. A comment node has no effect in the output and is transparent to the nodes below it. You can put comments anywhere in a report.



Nodes which have become invalid for any reason are converted to comment nodes by the system (see [Conflict Situations](#)).

### 8.2.3.2 Demoted Section

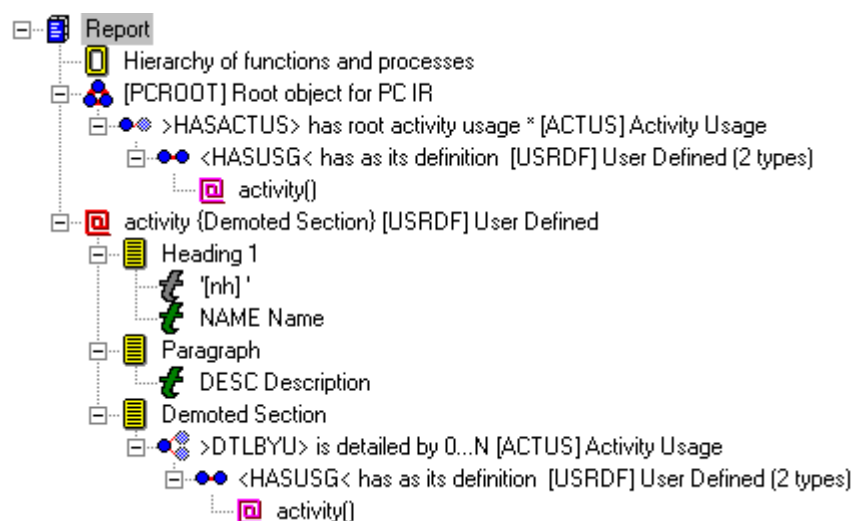


Encloses a logical section of the report that should have deeper [heading](#) levels. The [headings](#) in the tree below are output to the report with a level deeper by 1 than specified by the [heading](#) nodes.

By nesting of several Demoted Section nodes, you can create [headings](#) of any level.

Demoted Sections are especially useful in conjunction with a [subroutine](#), for example when recursively reporting a tree of objects: The headings created by the subroutine are output at different levels, reflecting the subroutine call nesting.

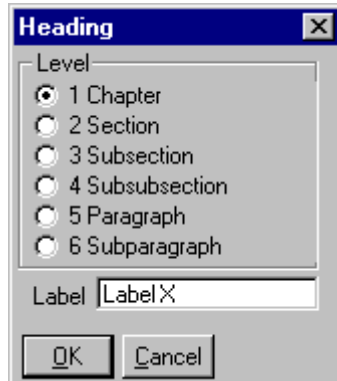
Example:



### 8.2.3.3 Heading



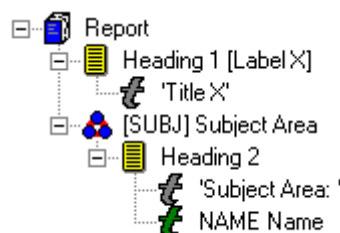
Creates a heading. The nodes below it determines the text that appears in the heading. Headings can have a label, which is used to distinguish between headings appearing in the same context. You can not use a heading inside a paragraph, a table or a list.



**Level:** Select one of 6 possible levels. You can create headings of deeper levels using [Demoted Sections](#).

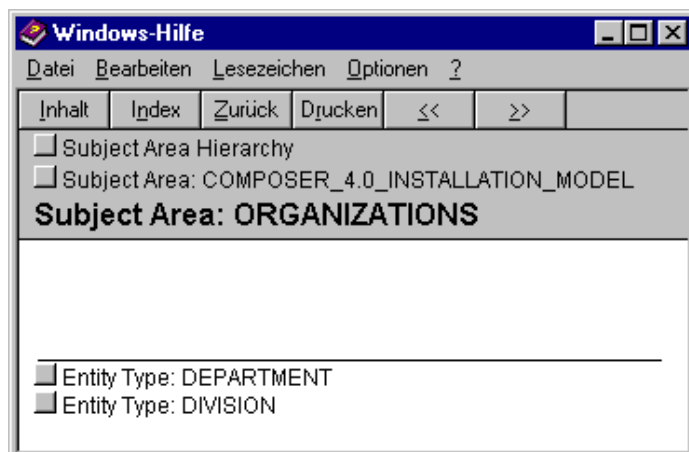
**Label:** String which identifies the heading in this report. The label is used together with the [Link](#) element. The label combined with the instance of the enclosing context is used to form a unique address. Examples: 'Intro.0', 'ATT.443765'. If more than one heading appear in the same context, it is strongly recommended to distinguish them by labels.

**Example:**



**Winhelp:**

CASEDOK creates a new help topic panel for every heading. Links to parent topics are placed on top of the topic, links to subordinate topics are placed at the bottom of the window. You can browse the topics sequentially using '<<' and '>>' buttons.



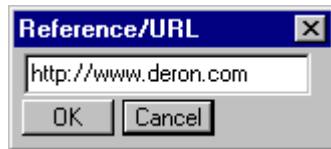
**Note:**

If you want to create a help file to be used as an application help, add a [MAP] section to the .hjp file and recompile it with the Microsoft Help Workshop (hcx.exe). By default, the title topic uses ID 0.

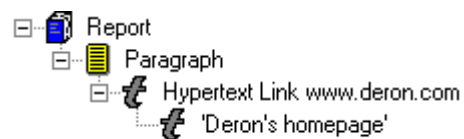
### 8.2.3.4 Hypertext Link (HTML only)



Inserts a hypertext link, addressing the given URL. The enclosed nodes form the text displayed on the link.



Example:

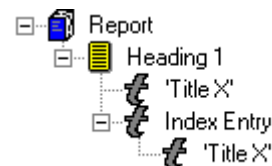


### 8.2.3.5 Index Entry



Creates an index entry; the tree below it determines the text of the index entry. If supported by the report type, an invisible anchor is placed at the current position in the report, to which the index entry is linked.

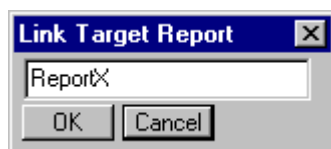
Example:



### 8.2.3.6 Library Link (HTML only)

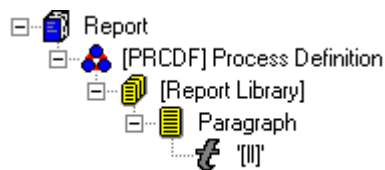


Inserts a hypertext link, which when clicked, causes **CASEDOK WebEXPLORER** to dynamically create a new document. The enclosed nodes form the text displayed on the link; if empty, the report name is used.



If a valid report name (filename without extension .rpr) is specified, this report is dynamically created. If no report name is specified, the report is created based on the current iteration of the next enclosing **Report Library** iterator. The created document is restricted to the current model and current object. By following the Library Links, the user can continuously explore a model.

Example:



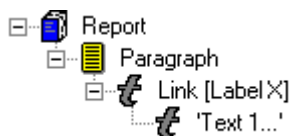
### 8.2.3.7 Link



Creates a hyperlink to the [heading](#) carrying the same label and referring to **exactly the same context**. Links are not supported in RTF reports.

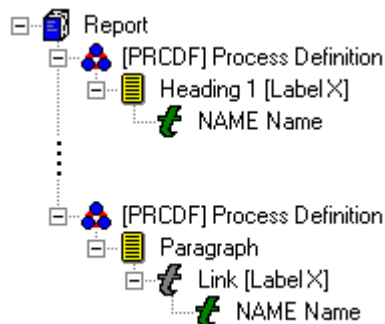


Example 1:



Text 1 is marked as a link to a heading with Label X. The label is required to identify the heading.

Example 2:



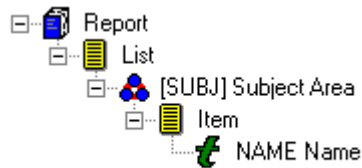
The Link element must have exactly the same context as the target heading. The internal jump address is the object ID. The label is used to build a unique address in case multiple occurrences of the heading context.

### 8.2.3.8 List



Creates a simple bulleted list. The list element is the container for [List Items](#). You can not use a list inside a paragraph, a table or a heading.

Example:



Creates a bulleted list with Subject Area names as items.

### 8.2.3.9 List Item



Adds a list item to the enclosing [List](#). The tree below determines the contents of the item.

### 8.2.3.10 Page Break (RTF only)



Inserts a page break in a RTF report. This element is not available in other report types.

### 8.2.3.11 Paragraph

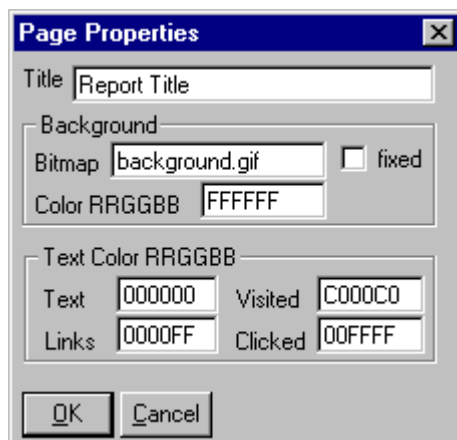


Creates a paragraph. The tree below it determines the contents of the paragraph. You can not use a paragraph inside a table, a heading or a list.

### 8.2.3.12 Report



This is the root of every report. Usually, the report element has no properties, but format-specific reports may have some specific properties: e.g. HTML



### 8.2.3.13 Script Node



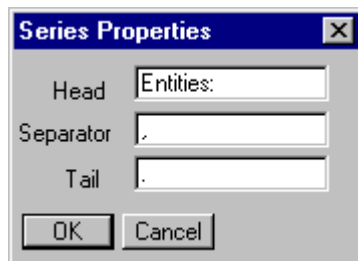
Script nodes provide an open concept to extend the Developer functionality. Experts can use script nodes to add new report features dynamically. Some reports in the standard CASEDOK RUNNER library are build with script nodes.

The script node concept is a very powerful and flexible way to customize reports in detail. The usage of script nodes is reserved for experts who know the internal object model of CASEDOK.

### 8.2.3.14 Series



Helps formatting a series of objects, providing an opening, a separating and a closing text. Series are always used within a context. The tree below the series contains the text (properties, literals etc.) for a single object in the series.



Head: String added to the document on the first instance of the enclosing context.

Separator: String added to the document on a repeated instance of the enclosing context.

Tail: String added to the document on the last instance of the enclosing context.

Example:



The data context (Entity Type) containing the series head='Entities: ', separator=',', and tail='.', with the property NAME, produces the text 'Entities: DEPARTMENT, DIVISION, EMPLOYEE, PROJECT, TEAM.'

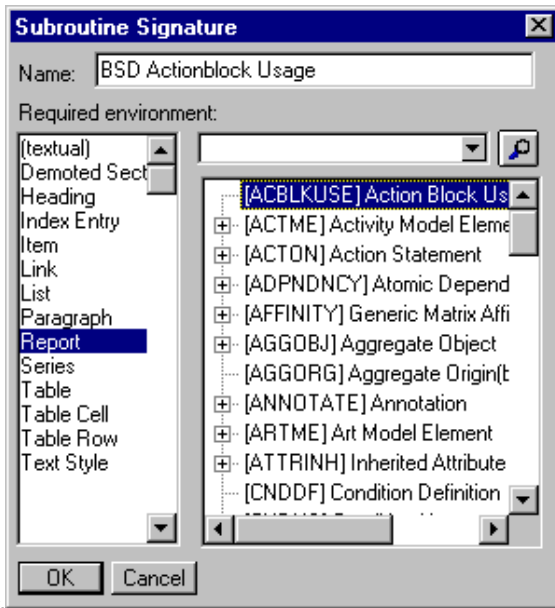
### 8.2.3.15 Subroutine



Creates a subroutine with a specific context and markup environment. Subroutines can be used to better structure a report and are especially useful to handle recursive data structures, e.g. hierarchies.

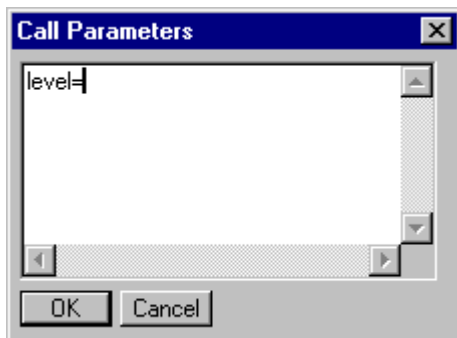
The subroutine markup element is only available on [Report](#) level. You have to define the subroutine before you can select the subroutine call from the markup window. The subroutine definition consists of a unique subroutine name, the markup action required by the subroutine and the context type expected by the subroutine.

The markup action '(textual)' allows a subroutine to be called within any textual environment like paragraphs, table cells, styled



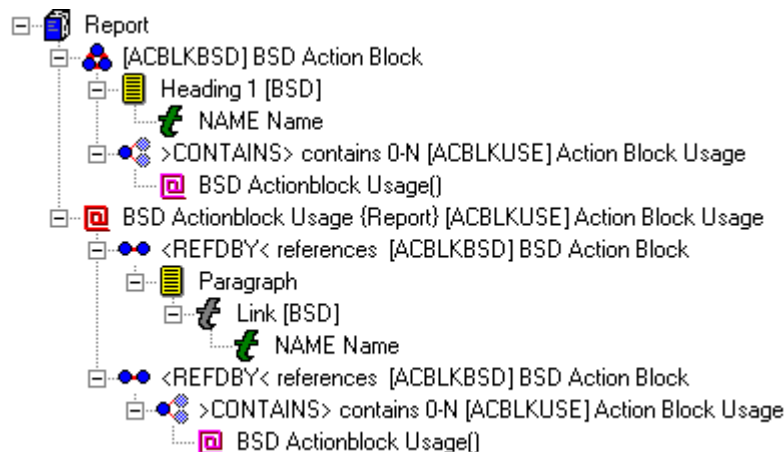
text etc.

The subroutine call element is automatically generated and displayed in the markup window (only visible when the report focus matches the adequate environment). The subroutine can be called with arguments. Each argument is written on a separate line, in the form name=value. The value assigned in the call can be used from [text literals](#) in the subroutine body exactly like a [macro](#). The value can contain macro references as well as references to values assigned in outer calls. Example: Number=[Number].1 gives a hierarchical numbering in a subroutine call tree.



**Attention:**  
**Never call a subroutine recursively from the same context!**  
**For example: BSD Actionblock subroutine directly calls BSD Actionblock subroutine**  
**The application do not handle such endless loops. Recursive calls always have to be placed within another context (see example below).**

Example:

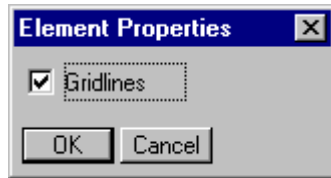


This report shows a recursive subroutine call to find nested BSD Ationblocks.

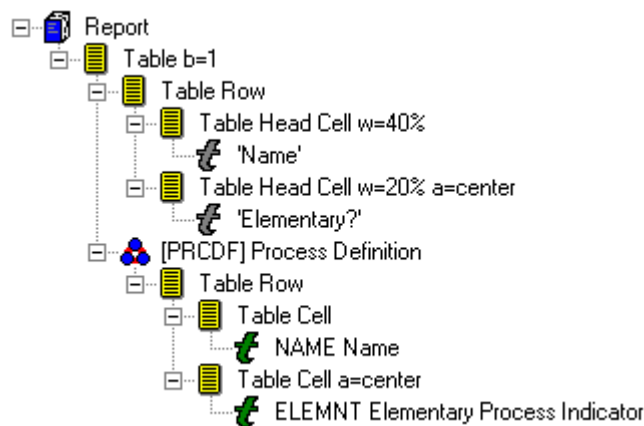
### 8.2.3.16 Table



Creates a table. A table contains table rows. If you activate the checkbox, the table is created with gridlines. You can not use a table inside a paragraph, a heading or a list.



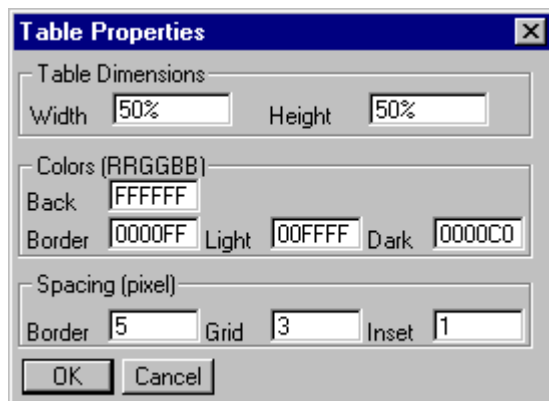
Example:



This example creates a table with one header row containing two header cells with column width information. For every process in the model, a data row is created, containing cells for the name and the 'Elementary Process Indicator' property.

#### HTML:

In HTML format, the table element has some specific properties:



Dimensions: Table width and height can be specified (in pixels or percent; example: '200' means 200 pixel, '50%' means half the page).

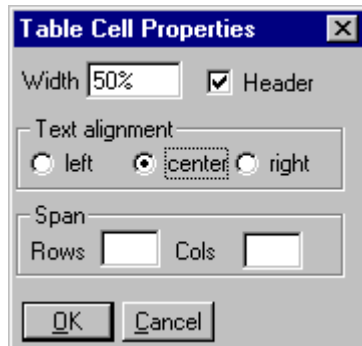
Colors: Table background color and border colors may be specified (as RRGGBB values in Hex; example: 'FFFFFF' means white).

Spacing: Border width, cellspacing and cellpadding may be specified (in pixels; example: '5' means 5 pixels).

### 8.2.3.17 Table Cell



Adds a table cell to the enclosing [table row](#). The tree below it determines the contents of the cell.



**Dimensions:** Width of the cell. Relative or absolute width may be used. Examples: '5cm', '20%', '0.8in', '0.2'. The following units are recognized: %, m, mm, cm, in, pt (1/72in), tw (twips; 1/20pt). Unitless means fraction of 1.

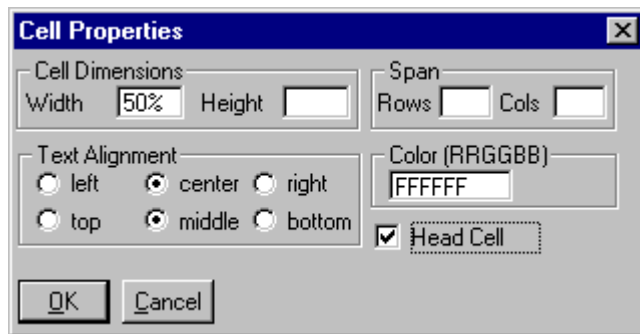
**Header:** If activated, the cell is set as table header cell. Header cells are specially highlighted.

**Text alignment:** Alignment of the text inside the cell.

**Spanning:** Cell spanning factors (empty or positive integers). Declares a cell to occupy multiple columns and/or multiple rows.

#### HTML:

In HTML format, the table cell element has some additional properties:



**Dimensions:** The cell width and height can be specified (in pixels or percent; example: '20' means 20 pixel, '50%' means half the page).

**Text alignment:** Alignment of the text inside the cell (additional: top, middle, bottom).

**Color:** Cell background color (in RRGGBB Hex notation; example: 'FF0000' means red).

### 8.2.3.18 Table Row

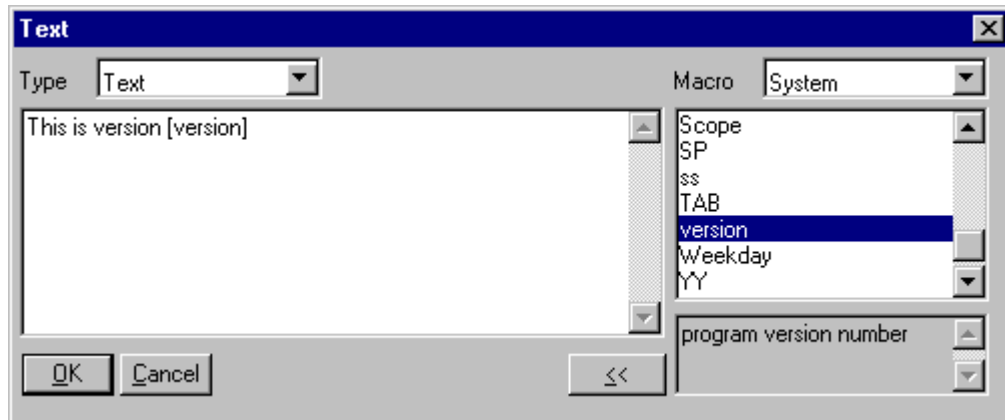


Adds a row to the enclosing [table](#). The tree below determines the contents of the row ([cells](#)).

### 8.2.3.19 Text



Adds continuous text to the document. Besides plain text, the text element may also include native Rich Text code, native HTML code or graphics. The text element needs an enclosing markup element.



**Type:** Designates the type of data entered in the text field. You may enter a different text for each type. The system chooses the most specific representation supported in a target report type. If, for example, a plain text, a native HTML code and a .GIF filename are specified, the system chooses the native HTML code and the .GIF file when generating a HTML report and ignores the plain text. If the target report type is Winhelp, the plain text is chosen and the HTML parts are ignored.

**Text:** Plain text for all document types (carriage returns are supported).

**Rich Text:** Native RTF code (e.g. {page}); supported in Winhelp and in most publishers such as Word, WordPerfect or WordPro. For example, you can use native RTF code to activate Word field functions.

**HTML:** Native HTML code (e.g. [p align=left]); supported in HTML browsers such as Netscape or MS Internet Explorer.

**Bitmap:** Filename of a bitmap (\*.bmp) including path information (e.g. c:\doc\l.bmp); supported in Winhelp and text documents. Use only 16bit color bitmaps to reduce the file size in RTF documents.

**Gif:** Filename of a .gif graphic (e.g. x.gif); supported in HTML documents. No path information supported; graphic must be located in the target document directory.

**Value:** Text value to be added to the document. It's interpretation depends on the type. Text in a native format (RTF, HTML) is written to the report as literal; the user is responsible for the correct syntax. Bitmap types (.BMP, .GIF) are interpreted as filenames.

**Macros:** Macros, which can be inserted in the text value. You may select a macro from the list or type the macro name directly using the bracket syntax [name]. You can use macros in all document types.

**System:** [System macros](#) can be inserted in text. System-defined macros are values provided from the system (e.g. model name, object id).

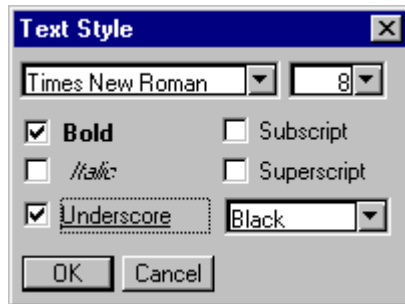
**User:** [User macros](#) can be inserted in text. User-defined macros are terms provided by the user and maintained in an external macrofile (\*.mco). See chapter 'Macrofiles' in the CASEDOK RUNNER manual.

Additionally, the value of [Report Parameters](#) and [Subroutine Call](#) arguments can be accessed in the same form.

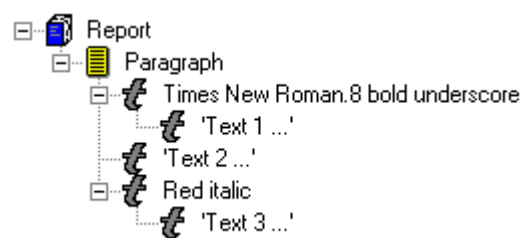
### 8.2.3.20 Text Style



Applies character format properties to the nodes it encloses. Text style nodes may be nested: inner properties are added to outer properties.



Example:



The example shows a paragraph with three text parts, where Text 1 and Text 3 are especially highlighted. For Text 2 the system default style is used.

## 8.2.4 Selecting and Sorting

There are two features which you can use for reporting on a specific selection of objects in a model:

- Write a report starting from a single object type, so a user can select from a list objects of that type to run the report on (see [How to run a report in CASEDOK RUNNER](#))
- Enclose parts of the report in a [Collection](#) and create an [Expression](#), which selects from the Collection instances at report run time, depending on [Variables](#) placed in the Collection

The second technique gives much more control over what is to appear in a report, although it has the drawback of processing objects whether they show up in the final output or not. Both techniques may be combined.

Expressions give you the ability to select and sort output in any order, based on arbitrary properties, or even extracts of properties.

### 8.2.4.1 Collection



Collects the output created by the enclosed nodes, without writing anything to the report. Processing is deferred to an enclosing [Expression](#), which decides whether the output is shown in the report or not.

Usually, a Collection is placed in the scope of a [Context](#), which instantiates the Collection once for every object it encounters.



The Collection label helps you identify the Collection in the report tree. It is stored in the special [Variable LABEL](#).

### 8.2.4.2 Expression



Expressions are used to control the output - collected by [Collections](#) - go into the report. An Expression refers to [Variables](#) associated with the Collections, in order to select the Collections of which the output is to show up in the report. Also based on Variables, an Expression can do sorting and duplicate suppression.

**Label:** Enter a descriptive comment

**Select Where:** Enter a condition or compose it from the dropdown lists. For any Collection instance treated by the Expression, if the condition evaluates to true, output is passed to the report, else it is suppressed.

#### Terms

- 0... Literal integer. In relation with a Variable representing a numeric property, numeric comparison is used. In all other cases, string comparison is used.  
Example: If max = 9, max < 10 is true, but max < '10' is false
- '...'  
Literal string.  
Example: 'CUSTOMER'
- a... Variable reference. Enter the name of a Variable you will later add in a Collection, or select from existing Variables. Variables referenced, but not existing in a Collection, have an empty value by default. Variable names are case sensitive and can contain letters, digits and \_.  
Example: title
- LABEL Special Variable; refers to a Collection's label
- [...] [Macro](#) or [Report Parameter](#) reference. Can be used to do selections based on parameter values entered in [CASEDOK RUNNER](#).  
Example: QA = [Responsible]

#### Relational Operators

- <, <=, >, >= Less Than, Less or Equal, Greater Than, Greater or Equal
- =, <> Equal, Not Equal

~, !~ Matching, Not Matching. The left-side operand is compared with the pattern provided in the right-side operand. The pattern can include wildcards \* (any characters) and ? (any single character).  
Example: name ~ 'ADD\*'

#### Boolean Combinations

(...) AND (...) Select Collection instances which are subject to both conditions.  
Example: (min > 1) AND (max < 100)

(...) OR (...) Select Collection instances which are subject to one or both of the conditions.  
Example: (YesNo = 'yes') OR (YesNo = 'Yes')

((...) AND (...)) OR (...) Nested condition. Helps construct switches based on a 'choice' Report Parameter.  
Example:  
`(([choice] = 'A') AND (LABEL = 'First')) OR (`  
`(([choice] = 'B') AND (LABEL = 'Second')) OR (`  
`...)`  
`)`

**Distinct In:** Enter one or more Variable names or select from the dropdown list. If more than one Collection instances have the same values in these Variables, only the first is selected.

**Order By:** Enter one or more Variable names or select from the list. Collection instances are sorted following these Variables' values. Prefix a Variable name with a '+' for ascending, with a '-' for descending order.

Example:



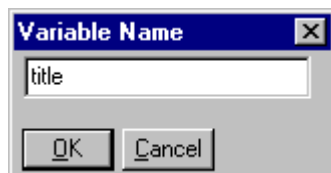
Every Window instantiates a Collection with its own Variable values. The Expression selects all Collections which have a specific value in the 'QA' Variable and sorts them by the value in the 'title' variable.

Expressions can be nested: The output passed to the report by an inner Expression can be caught by an enclosing Collection and treated by an outer Expression.

### 8.2.4.3 Variable

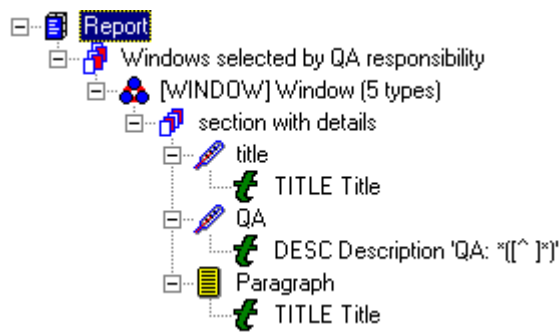


Variables are used for storing values in [Collections](#), based on which [Expressions](#) do the selection and sorting of the Collections.



The Variable name is entered in the dialog. The [Property](#) and/or [Text](#) nodes enclosed by the Variable provide the Variable's value; they do not create any output. Variables are associated with the next enclosing Collection. A Collection can contain Variables anywhere in its subtree, even in different Contexts.

Example:



The first variable, named 'title', is assigned the value of the 'Title' property. The second variable, named 'QA', is assigned an extract of the 'Description' property.

## 8.2.5 Iterators

Like [Contexts](#), iterators are elements that execute the tree they enclose for a number of times. The iterated elements may also include [Macros](#).

### 8.2.5.1 Report Library



This element is specifically used for **CASEDOK WebEXPLORER** reports. The element executes the tree below it once for every report in the report library which matches the enclosing context. The iteration includes all reports containing an initial context which is a superset of the enclosing context of the Report Library element. If Report Library is used outside a metamodel context, the iteration includes all reports belonging to the HTML phase.

Inside the tree enclosed by Report Library, the [System macros](#) [li], [ll] and [ln] can be used to refer to the current report number, the current report name and the number of reports. The [Library Link](#) action also refers to the current iteration.

The tree below Report Library is not allowed to contain context nodes.

## 8.2.6 Macros

Macros are text constants that may be used in the markup element [Text](#) or in value tables of [Properties](#). When used in a text, [name] references a macro. A literal '[' must be entered as [<] and a literal ']' as [>].

There are two types of macros: [System-defined macros](#) and [User-defined macros](#). If a macro name is not found in the system macros, it is searched in the macro file defined by the 'Macros' parameter in the configuration settings (please refer to chapter [Installation and Configuration](#) in the **CASEDOK RUNNER User's Guide**). If a macro is not found in the macro file, the macro name is written to the report.

### 8.2.6.1 User Macros

User-defined macros can be used to substitute report text by terms maintained in an external macro file (\*.mco). The default macro file is 'English.mco', which includes all the user-defined macros of the standards CASEDOK report library. You are free to adapt or extend this file with your own terms, or to create new macro files (e.g to provide different languages for the same report).

Example: [Entity]=Entität

Please refer also to the explanation inside the macro file 'English.mco'.

**Macro names are case-sensitive!**

### 8.2.6.2 System Macros

The following system macros are available:

Macro	Description
[<], [>]	A literal [, a literal ].
[LF]	Line Feed (new line)
[TAB]	Tabulator
[SP]	Protected blank
[ci], [cn]	The current instance number and instance list size of the enclosing context.
[ctmne], [ctype], [cid]	The type mnemonic, type name and current object ID of the enclosing context.
[nh]	The current decimal heading number. Example: '3.1.8'.
[li], [ll], [ln]	The current iteration position, iteration element and iteration size of the enclosing iterator. Example: in <a href="#">Report Library</a> , [ll] is replaced with a report name.
[Model], [Scope]	The name of the model being documented, the selection criteria being used for the document.
[hh], [mm], [ss]	The current hour, minute and second printed as two digits.
[DD], [MM], [YY]	The day, month and year of the current date, printed as two digits.
[YYYY], [Month], [Weekday]	The four digit year, month name, weekday name of the current date.
[version], [level]	The version and release level of CASEDOK being used for the report.

### 8.2.6.3 Report Parameters and Subroutine Call Arguments

The Report Parameters created when [saving a report](#) and the arguments supplied to a [Subroutine Call](#) can both be referred to exactly like [User Macros](#), although they do not appear in the lists.

## 8.3 Running Reports

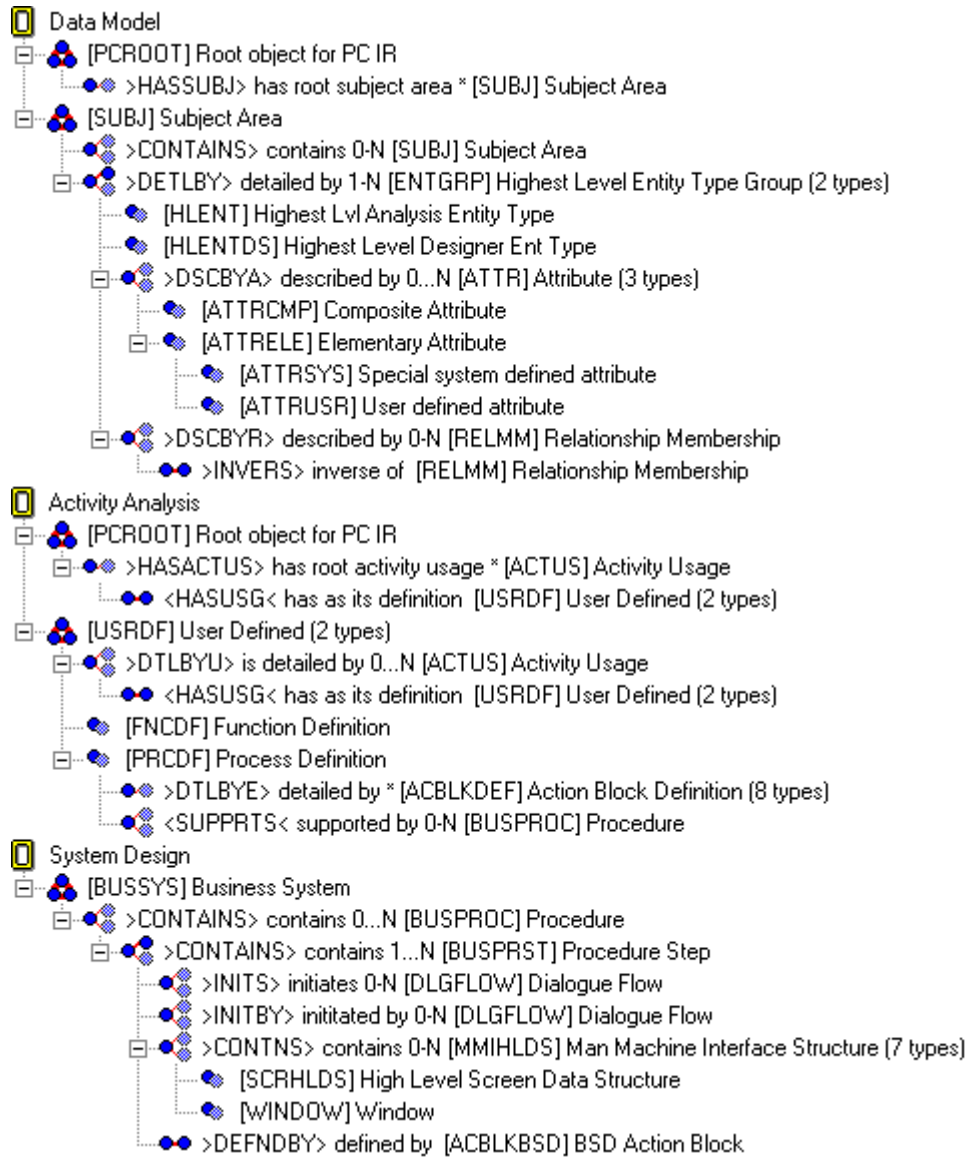
Reports can be executed with the CASEDOK RUNNER module. As soon as you have saved a report to the library, the changes take effect in the next documentation run. This works, even if the report you are working on, is already part of the current run job.

To start the RUNNER module, select <RUNNER> from the Windows Start menu or from the DEVELOPER's 'View' menu.

For more information, refer to chapter ['How to run a report'](#) in the **CASEDOK RUNNER User's Guide**.

## 9 Exhibit: Metamodel views

Please refer to the paper version of the **CASEDOK DEVELOPER User's Guide** to see some helpful illustrations of the COOL:Gen metamodel.



## INDEX

Actions .....	§8.2.3
Cast Nodes .....	§8.2.1.2
Cell .....	§8.2.3.17
Collection .....	§8.2.4.1
Comment .....	§8.2.3.1
Conditional Execution .....	§8.2.1.4
Conflict Situations .....	§5.2.1
Contact Deron .....	§1
Context Window .....	§5.3.2
Contexts .....	§8.2.1
Demoted Section .....	§8.2.3.2
Divisions .....	§8.2.1.1
Each Instance .....	§8.2.1.4
Editing Reports .....	§8.2
Exhibit: Metamodel views .....	§9
Expression .....	§8.2.4.2
First Instance .....	§8.2.1.4
Getting started: Sample report .....	§7
Heading .....	§8.2.3.2, §8.2.3.3
How to edit a report .....	§5
How to run a report .....	§6
Hypertext Link .....	§8.2.3.4
Index Entry .....	§8.2.3.5
Installation and Configuration .....	§4
Iterators .....	§8.2.5
Library Link .....	§8.2.3.6
Link .....	§8.2.3.7
List .....	§8.2.3.8
List Item .....	§8.2.3.9
Macros .....	§8.2.6
Managing Reports .....	§8.1
Markup Window .....	§5.3.1
Matching Patterns .....	§8.2.2.1
Metamodel views .....	§9
Node Command Menu .....	§5.2
Ordering .....	§8.2.1.3
Page Break .....	§8.2.3.10
Paragraph .....	§8.2.3.11
Properties .....	§8.2.2
Property Window .....	§5.3.3
Reference .....	§8
Report .....	§8.2.3.12
Report Editor .....	§5.1
Report Library .....	§8.2.5.1
Row .....	§8.2.3.18
Sample Report: Add a title page .....	§7.2
Sample Report: Add Entity properties .....	§7.4
Sample Report: Add Index Entries .....	§7.6
Sample Report: Add Relationship List .....	§7.5
Sample Report: Add specific HTML Properties .....	§7.7
Sample Report: Create new report .....	§7.1
Sample Report: Save & Run report .....	§7.3
Script Node .....	§8.2.3.13
Selecting and Sorting .....	§8.2.4
Series .....	§8.2.3.14
Style .....	§8.2.3.20
Subroutine .....	§8.2.3.15
System Macros .....	§8.2.6.2
Table .....	§8.2.3.16

---

Table Cell .....	§8.2.3.17
Table Row .....	§8.2.3.18
Text .....	§8.2.3.19
Text Style .....	§8.2.3.20
Toolbox .....	§5.3
User Macros .....	§8.2.6.1
Variable .....	§8.2.4.3
What is CASEDOK DEVELOPER? .....	§3
What's new in version 4.0 .....	§2
When Empty .....	§8.2.1.4
When Instance .....	§8.2.1.4