

Telelogic Publishing Engine

Reference Guide

Release 1.0

Before using this information, be sure to read the general information under Appendix, [“Notices” on page 177](#).

This edition applies to **VERSION 1.0, Telelogic Publishing Engine** and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2008**

US Government Users Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Table of contents

Telelogic Publishing Engine - Reference	1
Introduction	1
Software requirements	1
TPE and DocExpress	1
Key differences	2
Concepts	2
Document template	2
Data source schema	3
Document specification	3
Concrete data sources	3
Output	5
Workflow	5
Publishing documents with TPE from a data source	6
Data	6
Queries and contexts	6
Filtering data	8
Sorting data	8
DOORS Data	9
Configuring a concrete DOORS Data Source	9
DOORS Schema	10
Images	13
OLEs	14
Tables	14
What cannot be extracted	15
Tau Data	15
Tau Schema	16
Queries	16
Filter	18
Sort	19
Type casting	19
Attributes	23
Special attributes	23
The query element	24

XML	25
Configuring an actual XML Data Source	25
What can be extracted	25
Output	25
Stylesheets	26
Word	26
Defining a Word output	26
Output	26
Stylesheets	26
Unicode data	27
OLEs	27
Post-processing	27
HTML	28
Defining an HTML output	29
Output	29
Stylesheets	29
Unicode data	29
PDF	29
Defining a PDF output	29
Stylesheets	29
Unicode data	30
TPE Launcher	31
Defining a document specification	32
Metadata	32
Multi – template document specifications	33
Functions	33
Preferences	34
Engine preferences	34
Preview preferences	34
Synchronizing document specifications with templates	34
Generate document	35
Open results	37
Command line switches	39
TPE Document Studio	40

Template elements	41
Components	44
Palette	44
Schema view	45
Editor area	45
Outline	45
Properties	46
Content	46
Adding elements	46
Editing element properties	46
Formatting properties	47
Calculated properties	48
Data	48
Adding data source schemas	48
Queries	50
Using data attributes in template elements	52
Filter	53
Sort	56
Contexts	59
Nested queries	60
Advanced usage of data attributes	61
Calculated values	62
Conditions	63
Filter vs. Conditions	64
Conditional formatting	65
Styles	67
Creating a style	67
Editing a style	72
Changing style properties	73
Applying a style	73
Using external styles	74
Style inheritance	74
Reserved style names	74
Master pages	75
Adding a master page	75
Using master pages	77
Data attributes in master pages	79
Advanced template design	82
Bookmarks	82
Hyperlinks	82

Variables	82
TPE Predefined Template	83
TPE launcher integration	83
Synchronized mode	84
Generate & preview	85
Unsynchronized mode	85
Schema Discovery	85
DOORS Schema Discovery	85
Tau Schema Discovery	93
DOORS Addin	98
Installation	99
Usage	100
Select the document template/specification.	100
Select the data sources to be used	101
Configure the data sources	102
Configure the data sources	103
Configuring a data source	104
Configure the output.	105
Document generation options.	105
Summary page	107
Tau Addin	108
Installation	108
Shared Document Library	109
Usage	110
Deployment scenarios	117
Local engine	117
Remote engine	118
Deploying the web service	118
How to	119

Tips & tricks	119
Editable elements	119
Table of Contents	119
Figure captions	120
Included files	120
Heading styles	120
Formatting properties vs. Styles	120
TPE Styles vs. External Styles	120
Numbering headings for Microsoft Word	120
Dynamic images	120
Dynamic included files	121
Unicode data in output	121
Moving an element outside the visible editor region	121
Table fit to window	121
PDF Tables	122
Table merge	122
Once per table	122
Frequently asked questions	123
Troubleshooting TPE	124
Common problems	124
TPE Core Logging	124
TPE UI Logging	124
Examples	125
A template for DOORS data	125
Basic setup	125
Advanced template options	135
More advanced settings	146
Generate the document	148
A template for Tau data	153
Define the template content	157
Generate the document	172
Appendix A: Notices	177
Trademarks	179

Telelogic Publishing Engine - Reference

Introduction

Documents continue to be the preferred and most effective way to communicate necessary information across all disciplines, both internally and across contractual boundaries. Creating documents and keeping them up-to-date can be time consuming and error prone. Documents can even become outdated by the time you have finished writing them.

Telelogic Publishing Engine automates documentation generation from Telelogic products and select third party applications, helping organizations generate documents for ad-hoc use, formal reviews, contractual obligations, or compliance with standards.

With Telelogic Publishing Engine you spend less time on the tedious and manual task of creating documents and keeping them up to date. Instead, you can leave these activities to Telelogic Publishing Engine and focus on your primary business objectives.

Benefits

- Reduce Risk of Errors through systematic creation of documents
- Enhance Customer Satisfaction with more effective communication
- Improve Regulatory Compliance by automatically generating required documents

Features

- Built-in extractors support many data sources
- Multiple output formats with complete flexibility in appearance
- Concurrent document generation to multiple target formats from a single template
- Composite reports containing data from multiple sources
- Predefined templates included out-of-the-box for rapid adoption

Software requirements

TPE requires Java Runtime Environment 1.6 or later. (IBM and Sun JREs supported)

For DOORS document generation a DOORS 9.0 or newer client installation is required.

For Tau document generation a Tau 4.2 or newer client installation is required.

NOTE You need to manually add Tau's bin folder, usually *C:\Program Files\Telelogic\TAU_4.2\bin*, to the *PATH* system variable.

TPE and DocExpress

While serving the same purpose, TPE and DocExpress are fundamentally different in their approach to data generation. TPE borrows some DocExpress concepts that proved valid, such as

embedding document generation capabilities within the data source application GUI, but its architecture, design and implementation are completely new and different from DocExpress.

To minimize the learning curve it is important to view TPE as a brand new product.

Key differences

DocExpress	TPE
Data driven generation	Template driven generation
-	Supports multiple data sources in the same document generation
Uses clipboard for data transfer	Does not use clipboard for document generation
Requires Microsoft Word	Does not require Microsoft Word
-	Can generate HTML, PDF
-	Improved performance
-	Improved formatting capabilities
-	Minimal impact access to data sources.

Concepts

TPE relies on several key concepts, described in the following sections:

- Document Template
 - Data source schema
 - Template content
- Document Specification
 - Concrete data source
 - Document output

Document template

This is the *blueprint* for the document generation.

A template is built of static and dynamic content. The static content is defined by data provided by the template designer at the time of the template. The dynamic content is represented by data extracted from the data sources at document generation time.

Formatting information can be defined in the template, but any formatting information in the data source can also be retained.

Ultimately, a template defines what data is to be extracted from the data source (through *queries*) and how to format all the information.

A template can also contain additional needed resources, such as static image files, and stylesheets.

NOTE A document template is self contained. It is stored as an archive file with the extension *.dta* (Document Template Archive). You can share/move/copy a template freely with no implications.

Data source schema

In TPE, the *queries* in a template are defined against the representation of the data source's structure (*schema*) and not on the actual data source. This allows for the template to be applied to any data source with a structure matching the one used when defining the template.

A TPE template can contain any number of data source schemas or no data schemas at all.

TPE uses standard XML Schema Definition for data source schemas.

NOTE A single data schema can represent any number of concrete data sources.

NOTE When a data source does not perfectly match the data schema, TPE will process the elements for which a match is found, and ignore these non-critical errors. For example if a DOORS Attribute named "status" is used in the template and the actual data source (a DOORS module) does not have this attribute, then the document generation will (by default) continue.

Document specification

Document templates define the content of the document while the document specification defines the templates to be used, the concrete data sources assigned to each template data source schema, and the desired output formats.

NOTE A single document specification can use any number of templates.

NOTE The document specification does NOT package any of the templates it uses or other resources. If you copy or move a document specification to a different location/machine then you need to copy all of the templates/resources used unless the templates are located on a shared location visible from the new document specification location.

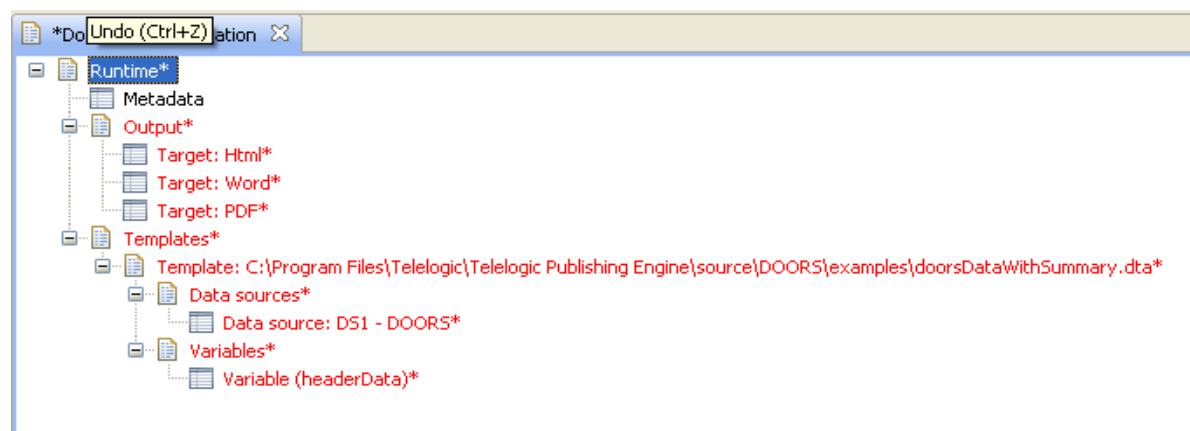


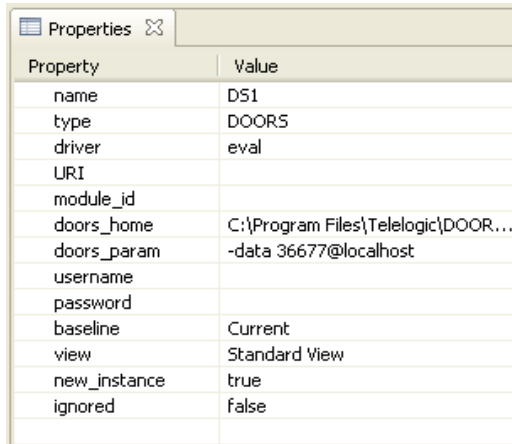
Figure 1

Concrete data sources

Configuring a data source means providing TPE with the details needed to locate and access the actual data source. Currently TPE supports 3 data source types:

- DOORS
- Tau
- Generic XML

Each type of data source has different properties that need to be configured. See the Input section for details on specific properties.



Property	Value
name	DS1
type	DOORS
driver	eval
URI	
module_id	
doors_home	C:\Program Files\Telelogic\DOOR...
doors_param	-data 36677@localhost
username	
password	
baseline	Current
view	Standard View
new_instance	true
ignored	false

Figure 2 A Concrete DOORS Data source

The properties shared by all data sources are listed below:

Property	Description
URI	The URI of the data source. The form and shape of the URI depends on the actual data source. For some data sources the URI is a URL.
Name	The identifier of the associated data source schema in the template. Read-only
Type	The data source type. Read-only
Driver	The TPE load driver to be used when extracting the data. Default value is "eval".
Ignored	Possible values: true/false. Default value: false If set to true, TPE will ignore all the template queries using the data source.

Output

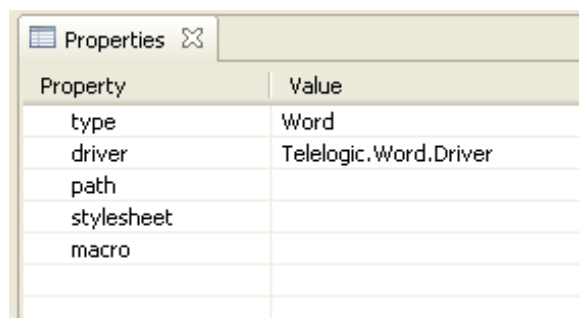
TPE 1.0 supports three output formats:

- Microsoft Word (97-2003 and 2007 format)
- HTML
- PDF

Configuring an output target means providing TPE with all the information needed to produce the output.

NOTE Not all output formats have the same set of capabilities. For a detailed description of formatting capabilities check the annex.

Each type of output format has different properties that need to be configured. See the Output section for details on specific properties.



Property	Value
type	Word
driver	Telelogic.Word.Driver
path	
stylesheet	
macro	

Figure 3 Word output properties

The properties shared by all output formats are listed below:

Property	Description
Type	The output type. Can be Word, HTML, and PDF. Read-only
Driver	The output driver used to create the output. Read-only
Path	The location of the output file. If empty TPE will generate a unique name in the user's temp folder.

Workflow

To produce a document from any supported data source type you need to do the following

- Create a document template

- Create a new document template using TPE Document Studio
- Define the schema for your data source. For some data types, such as DOORS, TPE provides assistance for building the schema.
- Insert the data source schema to the template
- Define the template's content
- Save the template
- Generate the output
 - Create a document specification in Launcher or Document Studio
 - Add the template defined previously to the document specification. At this time TPE will detect the data source schemas used by the template and will generate the required fields in the Document Specification that need to be completed with information about the actual data source
 - Configure the actual data sources by filling in the required fields
 - Provide values for the template variables as needed
 - Configure the output formats required by filling in the required fields

Publishing documents with TPE from a data source

In this scenario, a template or document specification can be selected via wizards in the data source application.

The wizard will then take the user through providing the needed information such as actual data sources and output formats desired. In many cases, the selection of actual data sources is sensitive to context and automated (e.g. when running from a DOORS Formal Module, the current Module is automatically selected as the actual data source).

Data

TPE currently supports DOORS, Tau and generic XML data sources.

Queries and contexts

As mentioned in the Document Template section, a TPE template specifies the data to be extracted using *queries*. A *query* is a path in the data source schema.

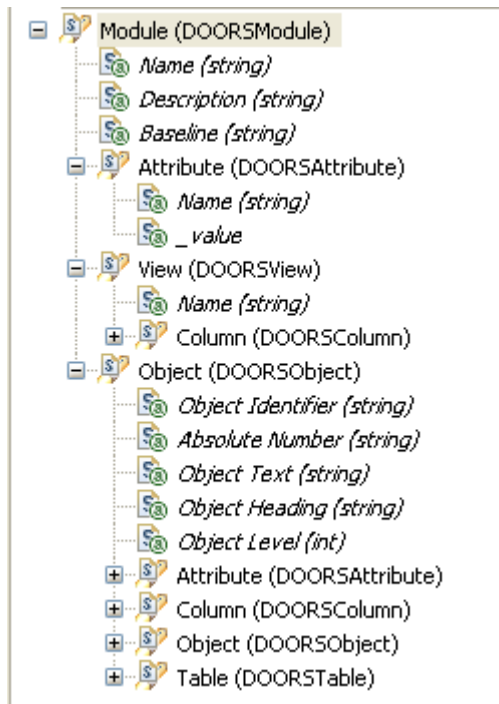


Figure 4 Sample DOORS Data Schema

NOTE For easier identification attributes are rendered in italic.

The Document Studio abstracts users from much of the complexity of manually writing queries with features such as drag and drop of schema elements. Nevertheless, it is still useful for template authors to understand the concepts of schema and queries and how they are constructed.

In the above data source schema some valid queries include:

Query	Description
module	returns a single result, the source module
module.object	Returns all the objects in the source module, as filtered/sorted by the source view
module.object.attribute	If used in a module.object context returns all the attributes for the current object.

A query can exist only in a template element. The template element and all its children can use the attributes of the entities returned by the current query and all of the queries from parent elements.

In the above example, if the query is *module.object* than any Object attribute from the schema can be used: Object Text, Object Heading etc

TPE template elements can be nested. Setting queries on elements and their children defines context. The query in the child element will be executed in the context of the parent's element query results.

Example:

Element 1: module.object

Element 1.1 (child of Element 1): module.object.attribute

The second query will only return the list of attribute names for the current object returned by the query of element 1. In element 1 only the attributes of DOORS Objects can be used while in element 1.1 the attributes of DOORS *Object* attributes can be used (i.e. the names of those Object attributes).

NOTE TPE Studio fully assists the user in building the queries and assigning the appropriate contexts. At no time you will be required to manually type a query.

Filtering data

Sometimes not all the data source is needed. In these cases you can limit the amount of processed data by setting a *filter* on the query. You can specify a filter in two ways:

- *TPE Filter* – JavaScript expression using the data attributes of the entities returned by the query
- *Native filter* – plain text that specific to each data source type.

When the query is performed, only the data entities matching the filter will be included in the output.

NOTE Not all data sources support native filtering.

NOTE For those data sources that support native filtering it is mandatory for the native filter to be a valid filter. TPE cannot and will not perform any validation on the native filter. Providing an invalid native filter can have results ranging from incorrect data in the output to the tool crashing.

NOTE For data sources accessed through TPE's Generic XML input driver it is not possible to define native filtering. The only exception to the rule is for the data sources where the filtering can be specified in the URL.

NOTE It is more efficient for filtering to be performed by the data source so whenever possible it is recommended to use a native filter as it should yield better document generation times than when a TPE filter is used.

Sorting data

Query results can be sorted. You can specify a filter in two ways:

- *TPE Sort* – the list of attributes and the sort direction (ascending/descending)
- *Native sort* – plain text that specific to each data source type. For DOORS this text must be in the format of the DOORS Sort

When the query will be performed, the elements will be displayed in the output document in the correct sort order.

NOTE Not all data sources support native sorting.

NOTE It is more efficient for sorting to be performed by the data source, so whenever possible it is recommended to use a native sort as it should yield better generation times than when a TPE sort is used.

DOORS Data

A **concrete** DOORS Data source is defined by a **view** from a **version** (i.e. Current version or a baseline) of a DOORS **module**. In this context the View is only used to define the filtered and sorted subset of data to use. By default the 'Standard View' is used that contains all Objects.

TPE can extract data from a DOORS database as long as a DOORS 9.1 Client is installed on the same machine. DOORS data can be extracted in two ways:

- Using a headless DOORS client run in the background
- Using an already running DOORS instance

The first method has the advantage of allowing continued use of any already running DOORS instances unhindered. The second method is slightly faster as the overhead of starting DOORS does not exist. The run mode is specified in the Document Specification using the TPE Launcher, by setting the *new_instance* property for each DOORS data source defined in the template.

NOTE TPE opens all the required modules in read-only access mode.

NOTE The data is extracted using DOORS DXL. On average the DXL execution time accounts for ~90% of the document generation time.

NOTE If the interactive run mode (using an existing DOORS instance) is set for a DOORS data source and no DOORS instance is running the data extraction will fail for that data source.

Configuring a concrete DOORS Data Source

When a DOORS data source is present in a Document template you need to define the following properties for the concrete data source:

Property	Description	Interactive DOORS Client	Headless DOORS Client
URI	The absolute path of the DOORS module in the database. Case sensitive	required	required
module_id	The module's unique ID. Used if the URI is not specified, ignored otherwise.	optional	optional
doors_home	The absolute file path of doors.exe	required	required

doors_param	The database to connect to and any other valid DOORS command line switch. Default: <i>-data 36677@localhost</i>	required	required
username	The DOORS account name to use for data extraction	not used	required
password	The DOORS account password (encrypted)	not used	required
baseline	The module version to use. Case sensitive Default: <i>Current</i>	required	required
view	The view to use for filtering/sorting. Case sensitive Default: <i>Standard View</i>	required	required
new_instance	If set to true a headless DOORS client is used otherwise TPE will attempt to use an existing DOORS instance. Values: <i>true/false</i> Default: <i>true</i>	-	-

NOTE Providing an incorrect value for any field marked as required (except *view* and *baseline*) will result in the output not being generated.

Providing an incorrect value *view* or *baseline* will result in the output being generated from the Standard View of the current module version.

DOORS Schema

The DOORS schema was designed to be simple to use and to match closely the DOORS module structure. TPE comes with a predefined DOORS schema that is generic and valid for all DOORS Formal Modules, but new DOORS schemas can be created using the “Schema Discovery” wizard in TPE Studio.

NOTE_The predefined DOORS schema is suitable when you are not interested in extracting particular user defined attributes from DOORS but rather the whole content of a view or all of the attributes of the DOORS Objects. When particular user defined attributes are required it is recommended to use the Schema Discovery wizard to generate a schema on your specific data.

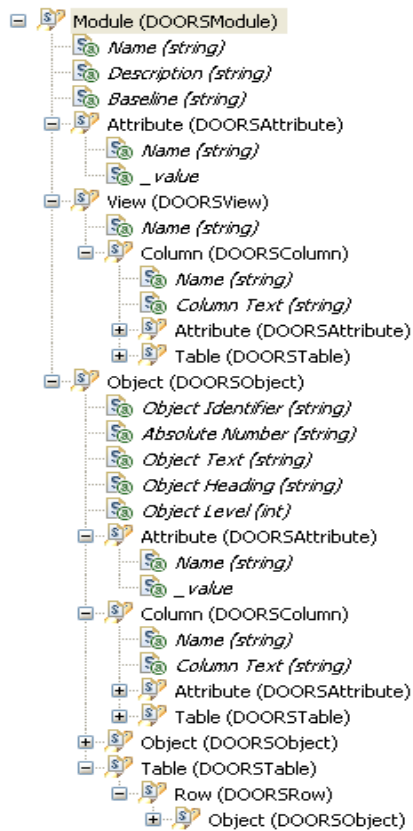


Figure 5 Predefined DOORS Schema

Queries and attributes

The following queries and attributes are from the default DOORS schema provided by TPE. User defined schemas will contain additional queried and/or attributes.

Query	Results	Attributes
Module	a single entity, the Module specified in the data source configuration	<ul style="list-style-type: none"> • <i>Name</i> - the Module's name • <i>Description</i> – the Module's description • <i>Baseline</i> – the Module version (baseline) used

module.attribute	the list of Module level attributes for the current Module	<ul style="list-style-type: none"> • <i>Name</i> – the attribute's name • <i>_value</i> – the attribute's value
module.view	a single result, the View defined for each DOORS data source in the document specification	<ul style="list-style-type: none"> • <i>Name</i> – the name of the View
module.view.column	<p>the list of columns for the selected View</p> <p>NOTE_The purpose of the <i>module.view.column</i> query is to provide a list of the column names without having to iterate the Module Objects. The result does not contain column data.</p>	<ul style="list-style-type: none"> • <i>Name</i> – the name of the column • <i>_value</i> – empty
module.object	the list of all Objects of the specified version of the current Module's as filtered/sorted by the selected View	<ul style="list-style-type: none"> • Object Identifier • Object Text • Object Heading • Absolute Number • Object Level • Any attribute elevated by the user in the schema discovery wizard
module.object.attribute	the list of attributes for the current Object if this query is in the context of a <i>module.object</i> query, or the list of all attributes for all Objects in the Module	<ul style="list-style-type: none"> • <i>Name</i>: the attribute's name • <i>_value</i>: the attribute's value
module.object.column	the list of columns in the selected View for the current Object	<ul style="list-style-type: none"> • <i>Name</i>: the column's name • <i>_value</i>: the column's value for the current Object
module.object.table	<p>no results if the current Object is not a DOORS table</p> <p>a single result, (the DOORS table) if the Object is a table header Object</p>	<ul style="list-style-type: none"> • none
module.object.table.row	the current table's rows	<ul style="list-style-type: none"> • none

module.object.table.row.object	A collection of Objects; the current rows' cells. Same attribute list available as for the <i>module.object</i> query	<ul style="list-style-type: none"> • Object Identifier • Object Text • Object Heading • Absolute Number • Object Level
--------------------------------	---	---

Images

Images are extracted with the attribute's value. You do not have to (nor can you) specify that you want to extract the images in a DOORS module. What you can configure is the size of the extracted images. The max size is specified through the "image max width" and "image max height" properties. These properties can be specified in two places:

- *element format info*– defines the images min/max size for the images contained in that template element

[-] Formatting
+ common
+ date
+ data
+ image
[-] image size
image width
image height
image max width
image max height
image min width
image min height

Figure 6

- *document specification metadata* - defines the image's min/max size for all the images in all the templates. The element level values override these global values.

date	Nov 5, 2008
time	4:52:18 PM
client	Launcher
machine	spurlos
build	1_0_20081104
data formatting	mixed
date pattern	yyyy.M.d
output locale	
image max width	
image max height	
OLEs as static images	true

Figure 7

OLEs

TPE can extract OLEs from a DOORS data source. How the OLEs reach the output document depends on the output type and the options selected in the document specification.

OLEs will always be rendered as images in HTML and PDF output as those formats do not support OLEs.

For Word output the OLEs will be rendered as static images or as OLEs depending on the “OLEs as static images” flag in the metadata section of the document specification.

If “OLEs as static images” is set to TRUE, OLEs will be included in the output document as static images.

If “OLEs as static images” is set to FALSE, TPE will generate a “ref” folder in the same location as the Word output document. The ref folder contains RTF files for the OLE objects in the DOORS data source. The word output will have one include field pointing to a RTF file for each OLE exported from DOORS.

NOTE TPE cannot update Microsoft Word fields. As a consequence the include fields will not be visible when you open the generated Word document. To make the fields visible you need to take one of the following actions:

Action	Result
select the entire document content and use the “Update fields” function in Word	The OLEs are displayed in the document. The document is not self-contained.
use the “updateFields” macro provided by TPE	The OLEs are displayed in the document. The document is not self-contained.
use the “insertOLEs” macro provided by TPE	The OLEs are displayed in the document. The document is self-contained.

NOTE Updating the fields in the document will not make the Word document self contained. This means that moving such a Word document from the machine it was generated on will prevent editing the OLEs. To make the document self contained you need to run the “insertOLEs” macro.

Tables

You need to explicitly query for DOORS tables as they are not extracted automatically from DOORS. Manually adding the queries for extracting a table requires extra effort when building the template but it has the advantage of allowing fine grained control over the formatting of the table.

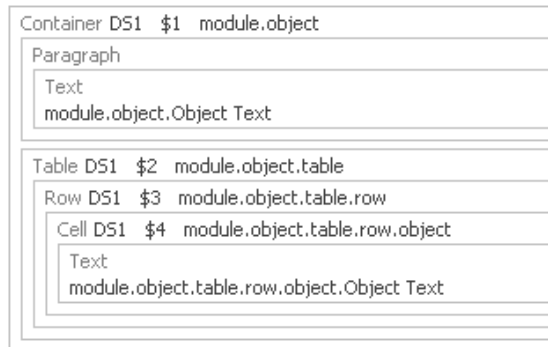


Figure 8 Extracting tables from DOORS

NOTE Due to how TPE handles elements, the table will only get created for those DOORS objects that start a table.

NOTE The cells of a DOORS table do not have a dedicated type in the DOORS schema. The content of the cells can be retrieved through the *module.object.table.row.object* query.

NOTE In the case of objects that are DOORS Table cells their *Object Text* attribute is a combination between *Object Heading* and *Object Text*.

What cannot be extracted

TPE does not allow the user to produce documents on the DOORS database structure. Furthermore TPE does not provide the means to query the *module's baseline list* or the *module's list of views*.

NOTE A DOORS Data Source in TPE is defined by the *<module, baseline, view>*. If you want to extract data from more than one module or baseline or view, you can do this in three ways:

- Define more than one DOORS Data Source in the template and add the corresponding template elements
- or*
- Add the same template multiple times to the document specification and configure the data sources for each template instance to the desired *<module, baseline, view>*
- Add multiple templates to the document specification and configure the data sources for each template instance to the desired *<module, baseline, view>*

Tau Data

A Tau source is defined by a single Telelogic Tau Project (*ttp* file). TPE can extract Tau data as long as a Telelogic Tau 4.2 or later software is installed on the same machine.

NOTE With the current build you need to add Tau *bin* folder to the system's PATH environment variable.

Tau Schema

The Tau schema used by TPE is automatically generated from a Tau metamodel using the provided GenerateSchema.tcl. The script is located in `source\Tau\schema` in the TPE installation folder.

NOTE The predefined Tau schema is built from the Tau metamodel. In order to build Tau documents you need to be familiar with the metamodel structure and its relationship with the model displayed in Tau's model browser.

The schema view will display all the elements reachable from the root element (usually model).

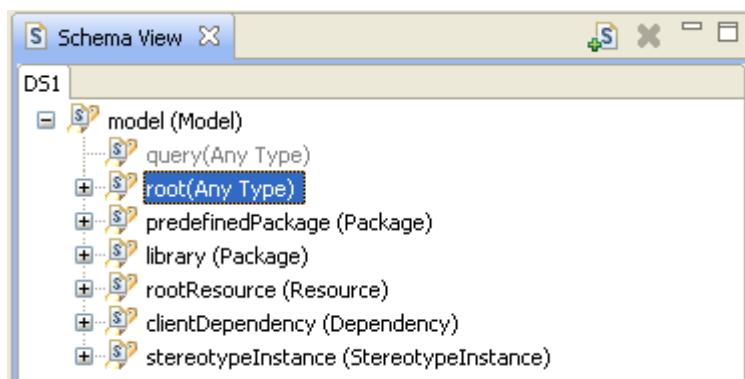


Figure 9 Tau Default Schema

Queries

A TPE query on a schema is a path in the schema, ex: `model.rootResource`. A query is applied to a TPE template element and defines the data context for that element and all its children.

NOTE The TPE syntax for the queries is similar to the XPath syntax. A major difference between the two is that a TPE query does not specify the filter in the query. TPE defines the filter and sort clauses separate from the query itself.

NOTE Each schema element (excepting the query) has an underline Tau native query (expressed in OCL) that will be used to fetch Tau data. For example, the `root` element under the `model` element has the following Tau query attached: `GetModelRoots()`

Nested queries

Adding queries to template elements and their children creates nested query.

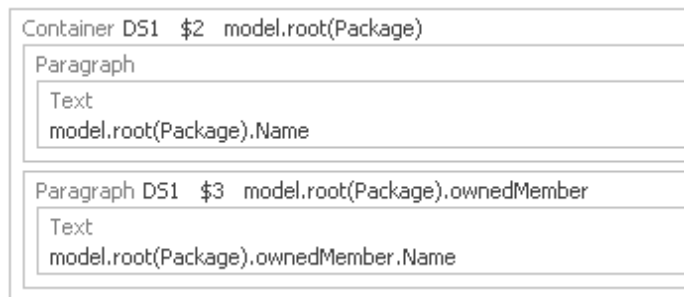


Figure 10 Nested queries

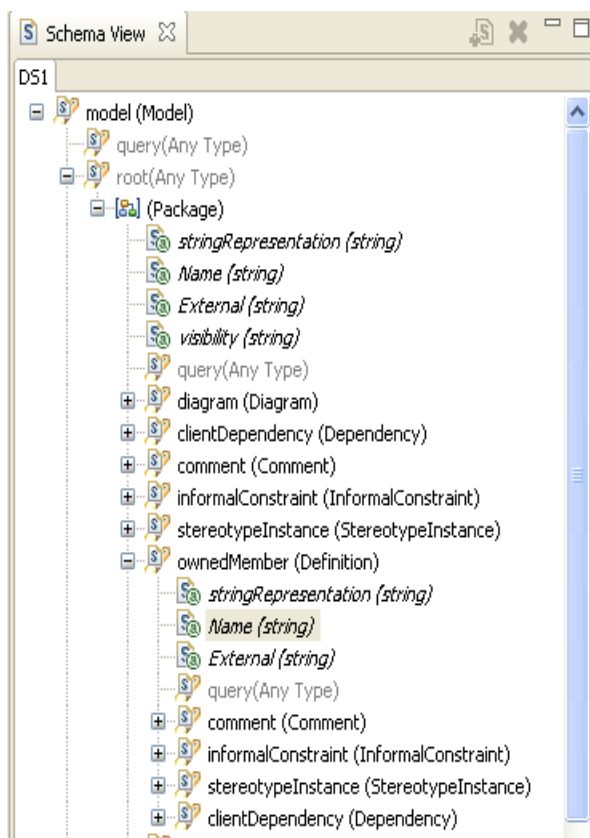


Figure 11 Schema

A nested query will be executed in the context of the results of the parent element. The first query, *model.root(Package)* will be performed in the context of the Tau model. The second query, *model.root(Package).ownedMember* will be performed in the context of each package returned by first query.

How TPE builds queries

Using the above example, if the user wants the list of all classes from the top level packages in model, the query would be: *model.root(Package).ownedMember(Class)*

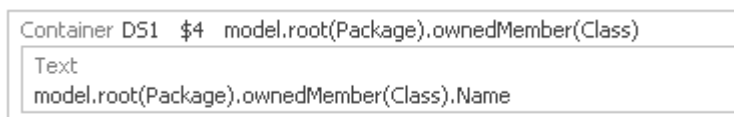


Figure 12 query

While in this form the output document will no longer contain the name of each package, TPE will build the class list in the same way as it did in the first case. The query is split into its component queries, and each query is executed in the context defined by the previous queries:

Sub query	Context	Result
model	-	the model
model.root(Package)	model	list of packages
ownedMember(Class)	list of packages	list of classes

Each sub query is performed once for every element in the context, and the results of each execution are concatenated. These results become the context for the next sub query, or the results list if the sub query is the last one.

Filter

For Tau Data Sources both native and TPE filters can be used.

Native Filter

The native filter for a query is specified in the “native filter” tab of the filter editor in Document Studio. As for any native filtering, the job is delegated to Tau, with no processing done in TPE.

A Tau native filter must be a valid Tau OCL script that returns a collection of elements. The native filter is concatenated to the underlying Tau query for the current schema element.

Example

Schema element:

model.predefinedPackage

Underlying Tau query:

GetEntities(“predefinedPackage”).select(IsKindOf(“Package”))

Native filter:

```
select(HasPropertyWithValue("Name", "Predefined"))
```

The query that will be performed in Tau is:

```
GetEntities("predefinedPackage").select(IsKindOf("Package")).select(HasPropertyWithValue("Name", "Predefined"))
```

Scripted filter

The scripted filter for a query is specified in the “script filter” tab of the filter editor in Document Studio. Scripted filtering is performed by TPE itself and follows the same rules and limitations as for any other data source.

Sort

Tau data sources do not have support for native sorting. For these data sources only TPE sorting can be used.

Type casting

Some schema elements do not have a type assigned to them as there can be more than 1 valid type for that. In such cases you can define which type to use through the “Cast to type” function in the schema view bar.

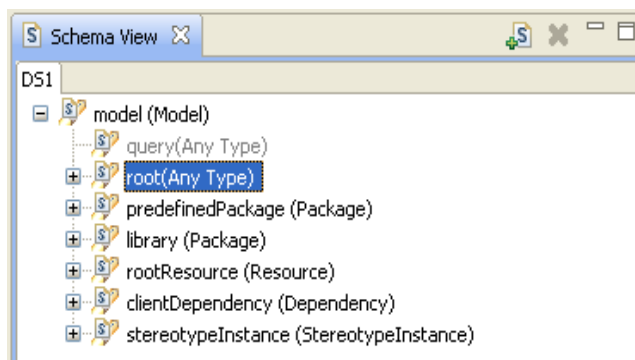


Figure 13 An element with no type

NOTE A cast query will filter the results of the regular query to return only the elements that can be casted to the selected type. Internally this is implemented by adding the *select(IsKindOf("Type"))* filter to the query.

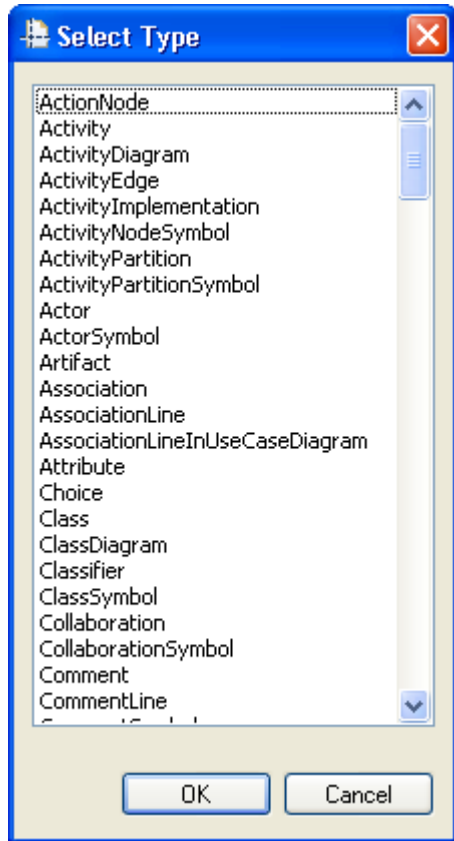


Figure 14 Select type

After the type is selected, it becomes available in the Schema View under the “anyType” element.

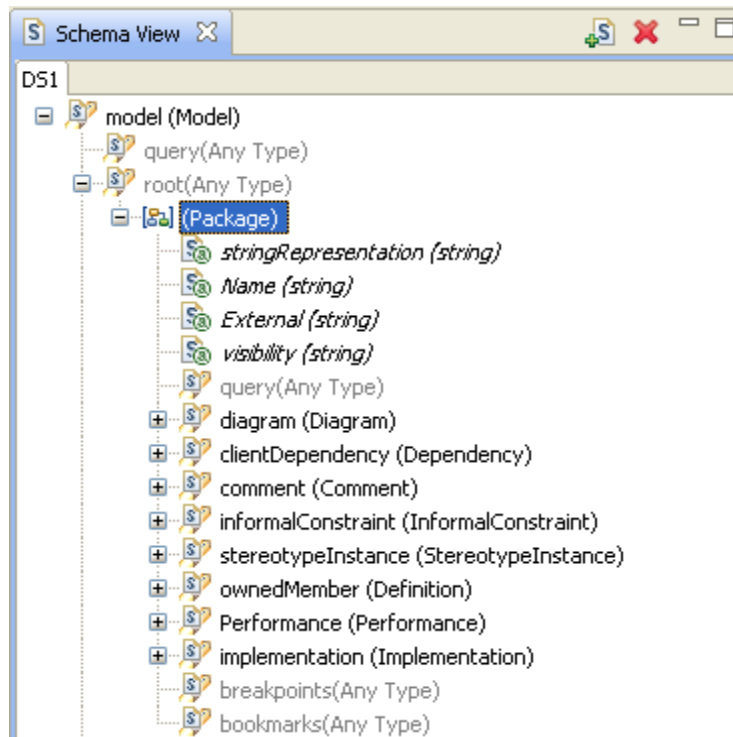


Figure 15 A "cast" to Package

Build queries using the "cast" allows access to all the child elements and attributes of the cast type..

Paragraph DS1	\$5	model.root(Package)
Text		model.root(Package).Name

Figure 16 A query using a cast type

Type casting can be used also when you want to refine the results of a query. Basically type casting works as a secondary filter for Tau elements.

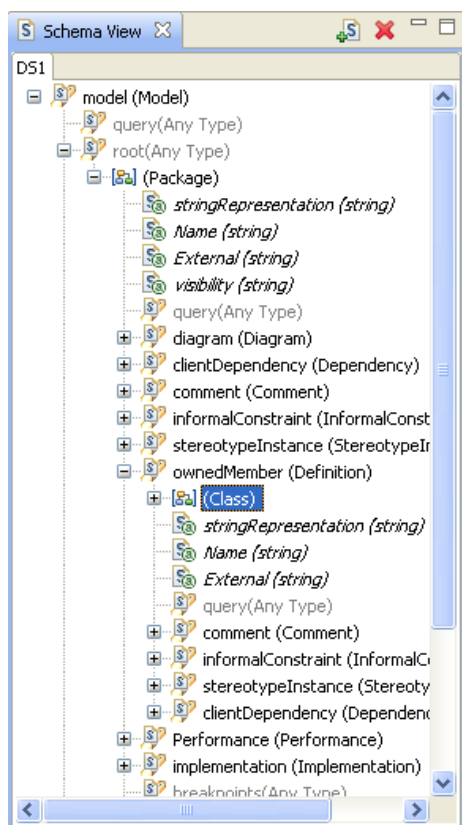


Figure 17

Adding a “Class” cast to the “ownedMember” element of a Package will allow defining the following query:

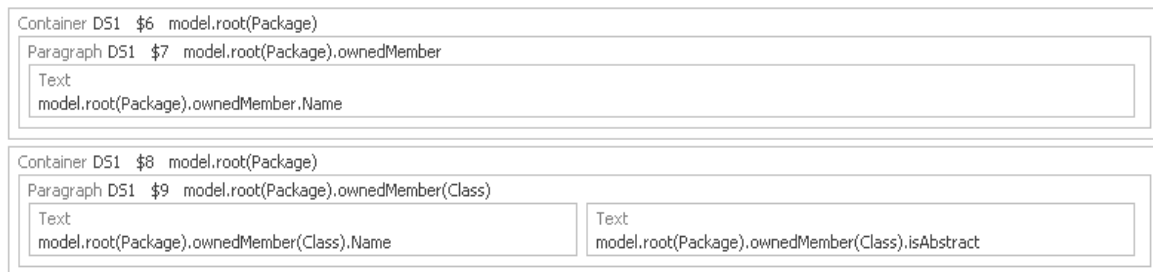


Figure 18

In the above example the query \$7 retrieves all the definitions contained in each top level package while query \$9 will return only the definitions that are classes from the same context.

NOTE With the current TPE version you can specify casts that are not correct (ex: from Class to Package). This query will yield no results.

Cast vs. Filter

The result set returned by a *cast query* is identical with the result set returned by query having a filter using an equivalent *IsKindOf* predicate. The main difference is that using a cast query gives access to the cast type attributes and child elements while using a query with a filter does not.

Attributes

Once queries are assigned to template elements, all the attributes of the elements returned by the queries can be used.

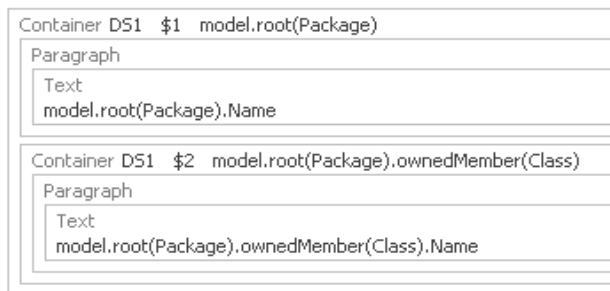


Figure 19

An attribute can be used in multiple ways:

- To define the content of a template element
- To be used in expressions calculating the content of a template element or its formatting capabilities

Special attributes

GUID

Every schema element has a special attribute named “guid”. This attribute is filled by TPE with the unique GUID of the current model element.

image

Every schema element representing a Tau diagram has a special attribute named *_image*. Using this attribute will generate an image file for the current diagram, image that can be included in the output.

stringRepresentation

Available for all expressions, actions and definitions, this attribute holds the unparsed representation of the element.

The query element

Every schema element has a special child element named “*query*”. Unlike the other elements in the schema, a “*query*” has no underline Tau query assigned and no type. Using the query element as is will return no results.

The purpose of the query element is to offer another level of customization for the TPE document generation. Should the existing elements should not suffice (or be optimal) for a given task, the user can define what he wants to extract (the type) and how he wants that extracted (the query) using the query element. The *type* is defined by adding a *type cast* to the query element while the *query* is defined in the template element’s filter, as *native filter*.

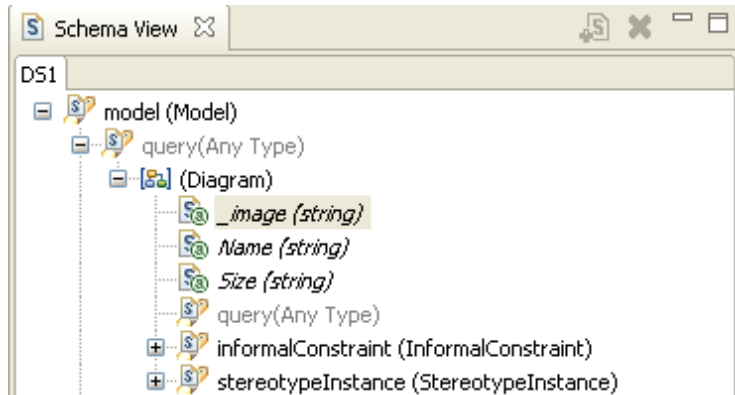


Figure 20 A cast added to a "query" element

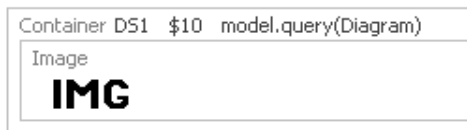


Figure 21 The query element with cast is used in the template

For the above scenario, a valid and meaningful query could be:

```
GetAllEntities().select( IsKindOf("Diagram"))
```

NOTE TPE will filter the results of the query based on the type cast used. If the query is syntactically correct, the result will contain only elements matching the specified type cast.

XML

TPE can extract data from any valid XML file if provided with the schema of the XML.

NOTE The document generation process will fail if the XML document is not valid (according to W3C XML definition – www.w3c.org/XML). A simple way to verify the file's validity is to open it in any web browser supporting XML. If errors are reported, TPE will not be able to handle the XML file either.

NOTE By default TPE does not try to validate the XML source against the schema used in the template. This will lead to documents being generated even if the XML file does not respect the schema. How much of the data can be generated depends on how different the XML schema used in the template and the actual XML file are.

Configuring an actual XML Data Source

The only property that needs to be configured for an XML data source is the URI of the XML file. The file can reside on the local file system, an accessible network file system, or on a remote system accessible through a URL.

NOTE If a web browser can open the XML file than TPE should also be able to access the file using the same URL.

What can be extracted

Any attribute specified in the schema. For XML simple types TPE provides an attribute named “_value” that allows access to the content of the element.

NOTE If an attribute is not specified in the schema it is not accessible from TPE.

Output

TPE can produce Microsoft Word, HTML or PDF documents. No additional software is required for the document generation process. You need additional software to view the results of document generation or to perform post processing operations where applicable.

TPE's offers a broad range of template elements and formatting operations but it is not, and doesn't attempt to be, as rich as a fully fledged Word Processing environment such as Microsoft Office or Open Office.

TPE's Template Editor does not attempt to be a replacement for your Word Processor. Although TPE's Template Editor allows building templates visually, its primary purpose is to define the template structure, and not all of the formatting that can be specified is rendered visually.

While you can specify that a paragraph should be centered, the Template Editor will provide no visual feedback. However the generated document will contain the paragraph centered as expected.

Stylesheets

The HTML and Word output formats allow specifying stylesheets to be used in the document generation. A stylesheet can be used to ensure certain styles are available in the generated document. A stylesheet can also be used to add content to the generated document: standard front pages with company logos, legal information etc.

Word

The Microsoft Word documents generated by TPE are in the Word 97-2003 format. You do not need Microsoft Word or any additional software installed on your machine unless you want to perform post-processing operations on the output.

Defining a Word output

By default any newly created document specification contains a Word Output. If you have deleted it you can add it at a later time. A Word output is defined by the following properties:

Property	Description	Required
path	The full path where TPE should produce the output document. If not specified, TPE will produce the file in the temp folder (with a random generated name).	no
stylesheet	The full path to the stylesheet to be used.	no
macro	The macro to run after the document is generated. NOTE requires Microsoft Word installed on the machine hosting the TPE engine.	no

Output

TPE's Word output consists of a single Microsoft Word document. You need to manually (or via a macro) update any fields you might have added, such as Tables of Contents, Tables of Figures, Tables of Tables, figure captions and label captions.

NOTE Due to implementation details TPE cannot update Microsoft Word fields in the generated document.

Stylesheets

You can specify any Microsoft Word 97-2003 document or template (.doc or .dot). TPE will create the output document based on the stylesheet. This translates into:

- Any content in the stylesheet will be present in the generated document
- All styles from the stylesheet will be available in the generated document

NOTE TPE does not yet support Microsoft Word 2007 format, though Microsoft Word 2007 may be used to open documents generated by TPE

Unicode data

Unicode data will appear as Unicode as long as the font used in the output can display such characters. The specified font must exist on the machine used to view the document.

The font family used for a template element can be specified through:

Location	Applies to
"font family" property of the element	The element and all the child elements that do not explicitly set the property.
"font family" property of the style applied to the element	The element and all the child elements that do not explicitly set the property.
font properties of the styles defined in the stylesheet (other than normal)	All the template elements using the specified style
Normal style in the template	All template elements that do not have a different style applied.

OLEs

TPE can extract OLEs from data sources supporting them, such as DOORS. Depending on the "OLEs as static images" flag in the metadata section of the document specification the OLEs are inserted in the output document as static images or as include fields to rtf files containing the OLEs.

If "OLEs as static images" is set to FALSE, TPE will generate a "ref" folder in the same location as the Word output document. The ref folder contains RTF files for the OLE objects in the DOORS data source. The word output will have one include field pointing to a RTF file for each OLE exported from DOORS.

NOTE TPE cannot update Microsoft Word fields. As a consequence the include fields will not be visible when you open the generated Word document. To make the fields visible you need to do one of the following:

- Select all the document content and use the "Update fields" function in Word
- Use the "updateFields" macro provided by TPE
- Use the "insertOLEs" macro provided by TPE

NOTE Updating the fields in the document will not insert the OLEs in the document. Moving such a Word document from the machine it was generated on will prevent editing the OLEs. To make the document self contained you need to run the "insertOLEs" macro.

Post-processing

When generating documents in Word format it is possible to specify a macro to be executed at the end of the process. The macro will be executed only if Microsoft Word is installed and available on the machine hosting the TPE engine.

NOTE The macro needs to be defined in the stylesheet used for Word output.

NOTE The macro is run with Microsoft Word in visible mode. This behavior is intended so that you can confirm/cancel security messages or any other messages added by the macros.

NOTE The macro execution might take a while depending on how large your document is and how complex the macro is. You should wait for the macro execution to complete before opening the Word document.

Word macros

TPE comes with a predefined set of Word macros meant to assist you in post processing TPE generated Word documents.

updateTOCs

Location: TPE_INSTALLATION/utils/word/updateTOCs.txt

Updates all the Table of Contents in the document.

updateFields

Location: TPE_INSTALLATION/utils/word/updateFields.txt

Updates all the fields in the document.

updateTablesOfFigures

Location: TPE_INSTALLATION/utils/word/updateTablesOfFigures.txt

Updates all the Tables of Figures in the document.

insertOLEs

Location: TPE_INSTALLATION/utils/word/insertOLEs.txt

Embeds all the OLEs from the linked RTF files in the output document.

includeLinkedFiles

Location: TPE_INSTALLATION/utils/word/includeLinkedFiles.txt

Breaks all the links to other documents (whenever possible) making the document self contained.

Tpe

Location: TPE_INSTALLATION/utils/word/tpe.txt

The *tpe* macro performs the actions from all the above macros.

For your convenience TPE provides a Word dot file that contains all the above macros. The file is located in TPE_INSTALLATION/utils/word/tpe.dot You can use this file as a stylesheet for Word output, in which case you can use any of the above macros.

HTML

By default any newly created document specification contains an HTML Output. If you have deleted it you can add it at a later time. An HTML output is defined by the following properties:

Defining an HTML output

Property	Description	Required
path	The full path where TPE should produce the output document. If not specified, TPE will produce the file in the temp folder (with a random generated name).	no
stylesheet	The full path to the stylesheet to be used.	no

Output

A HTML file in the specified location and a .css file with the same name. A folder named *img* will be created in the same location and will contain all the images referred to by the template. If a stylesheet is specified, the stylesheet will also be copied into the same folder as the result file.

Stylesheets

You can specify any .css file to be used by the generated HTML. The generated HTML file will use that .css file along with the one generated by TPE.

Unicode data

Unicode data will appear as Unicode as long as the font used in the output can display such characters.

PDF

By default any newly created document specification contains a PDF Output. A PDF output is defined by the following properties:

Defining a PDF output

Property	Description	Required
path	The full path where TPE should produce the output document. If not specified, TPE will produce the file in the temp folder (with a random generated name).	no
default font	The font to be used when this information is not specified in the template	no
unicode output	Set this value to true if the input data is Unicode.	no

Stylesheets

The PDF output does not support stylesheets, all formatting needs to be specified in the template.

Unicode data

This value can be safely set to true as long as the following conditions are met:

- the *default font* is a True Type Font
- no template element uses a non-True Type Font family

If “*unicode output*” is set to false, any non-English characters will not be visible in the output.

If “*unicode output*” is set to true, all non-English characters will be visible in the output as long as the default font or the element’s font can render them.

NOTE The above behavior is a limitation of the technical solution for TPE 1.0 and we are looking to improve it in future versions of the tool.

TPE Launcher

This TPE component allows building specification for TPE document templates and generating output documents from document specifications.

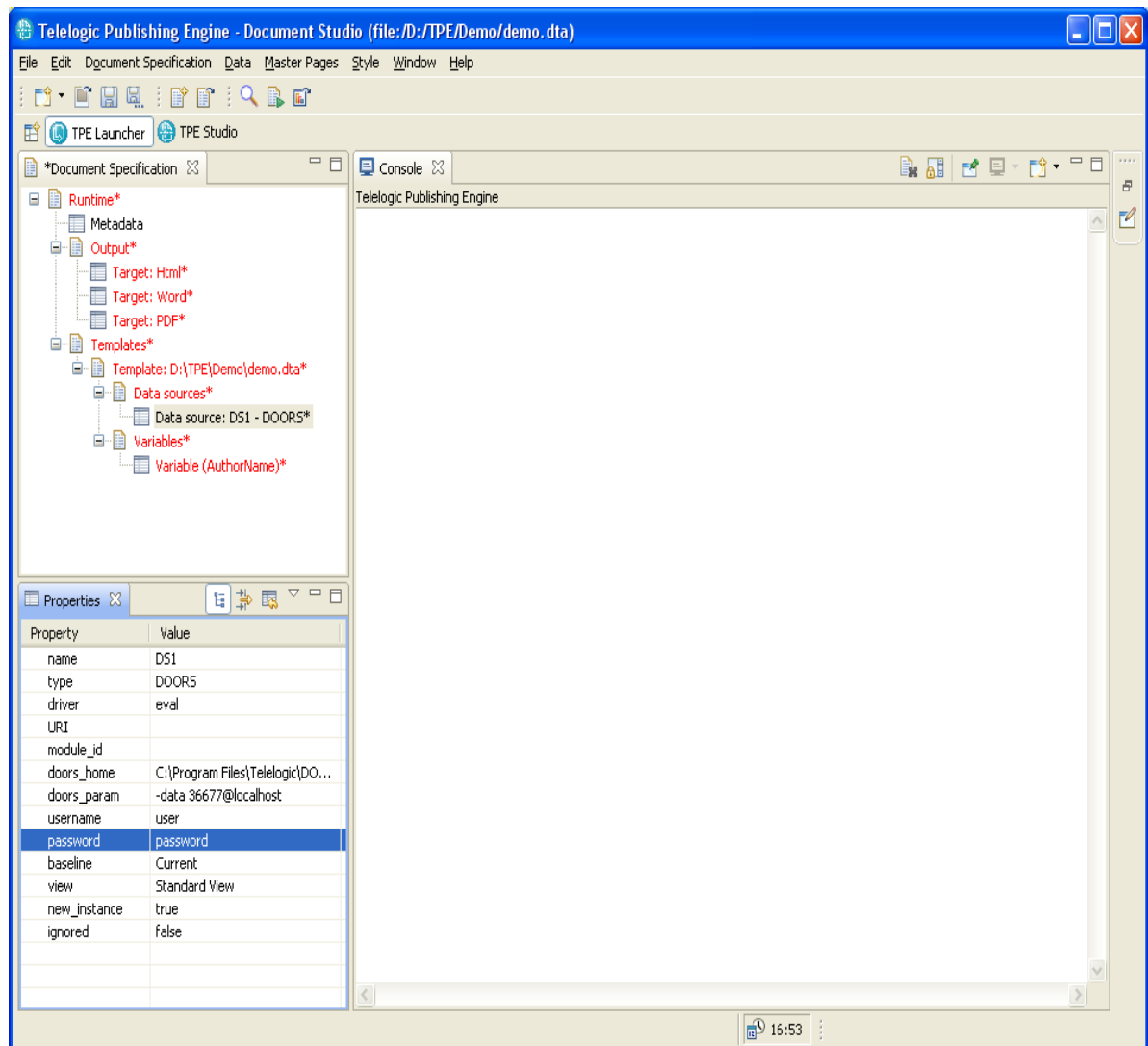


Figure 22 TPE Launcher

Defining a document specification

A document specification consists of:

- metadata – global settings for document generation
- output – the selected output formats
 - target – configures the document output (see Output sections for details)
- templates – the list of templates used by the current document specification
 - template
 - data sources – the list of data sources that need to be configured for the template. Each data source in the document specification matches exactly one data source schema in the template
 - variables – the list of variables defined in the template. Allows the user to define the values to be used when generating the output

Metadata

The metadata section of a document specification contains non-editable data generated by TPE meant to serve for informational purposes and editable data meant to configure the document generation.

Property	Description	Editable
Date	The date when the document specification was created	No
Time	The time when the document specification was created	No
Machine	The machine name on which the document specification was created	No
client	The application used to create the document specification	No
Build	The build number of the application used to create the document specification	No
Data_formatting	Flag controlling how TPE mixes template and data source formatting: Mixed – mixes source and template formatting Source – discards template formatting preserving data source formatting only Template – discards source formatting preserving data source formatting only	Yes
Data_pattern	The format in which dates are to be rendered in the output.	Yes
Output_locale	The output locale to be used (for default date formats) if it needs to be different from the system's locale	Yes
Image max width	The maximum width of all images in the output documents	Yes

Image max height	The maximum height of all images in the output documents	Yes
OLEs as static images	Flag controlling how TPE handles OLEs: <ul style="list-style-type: none"> • True – OLEs reach the output as static images • False – for Word output only OLEs are preserved in the output 	yes

NOTE None of the metadata properties are mandatory to be filled by the user.

Multi – template document specifications

A document specification can contain as many templates as desired. A document template can be referred to more than once in a single document specification but you will have to configure its data sources for each instance. See the Input Section for details on how to configure each data source type.

Functions

Function	Location	Description
New	Menu, toolbar	Creates a new document specification
Open	Menu, toolbar	Opens an existing document specification
Close	Menu	Closes any existing documentation specification
Save	Menu, toolbar	Saves the current document specification. If this is a new document specification, the “Save As” dialog is presented to the user.
Save as	Menu	Saves the current document specification in a new location.
Synchronize Document Specification	Menu	see details below
Generate document	Menu, toolbar	“Runs” the current document specification, generating the output documents.
Preview document	Menu, toolbar	“Previews” the current document specification, generating the output documents. Identical to the Generate Document function but internally limits the number of results for all queries in all templates to the number given in the preference pages.
Open results	Menu, toolbar	Opens a dialog containing the list of the most recent generation results. You can click on the links to open/save the results or right click to save them.

Preferences

Engine preferences

Defines which TPE engine to use. Default is the local installation but that can be changed with a remote TPE engine.

A remote TPE engine installation is specified with the URL to the TPE webservice.

Ex: *http://machine:8080/tpe/services/tpeserver?wsdl*

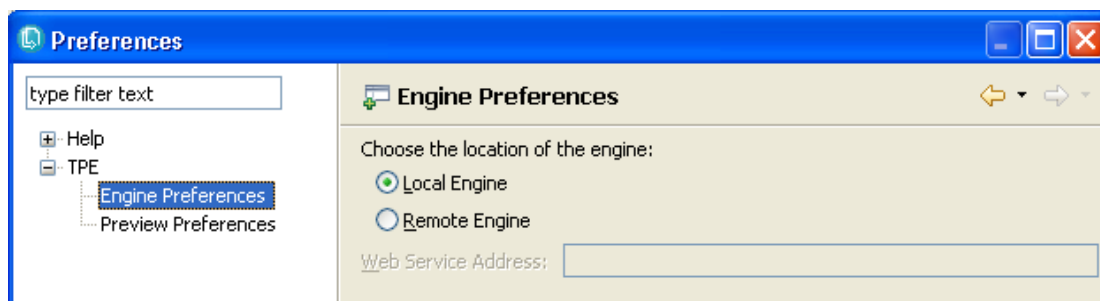


Figure 23

NOTE You need to contact your system administrator to get the TPE engine URL valid in your environment.

Preview preferences

Controls the maximum number of elements per query when doing a preview

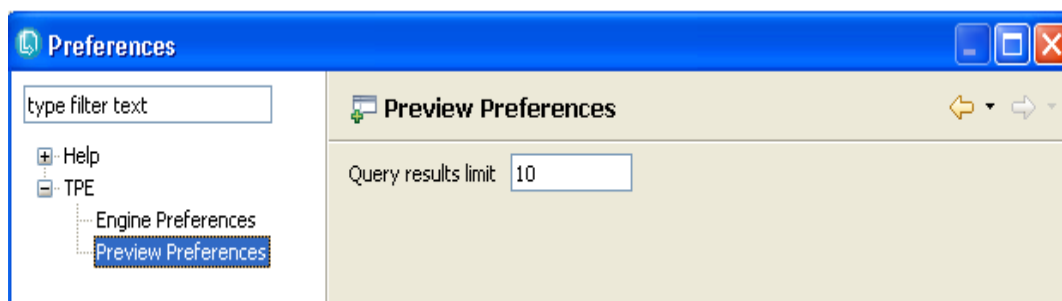


Figure 24

Synchronizing document specifications with templates

If data source schemas are added to a template after the template was assigned to a document specification, the document specification is considered out of synch with the template.

This function scans all the document templates assigned to the current document specification, and adds Data Sources and variables for all the Data Source Schemas and variables from templates.

- Existing data source – schema associations are preserved
- Data sources for which a schema association no longer exists are removed
- New data sources are added for new schemas in the templates
- New variables defined in the templates become available in the document specification
- Variables deleted from the template are also deleted from the document specification

Generate document

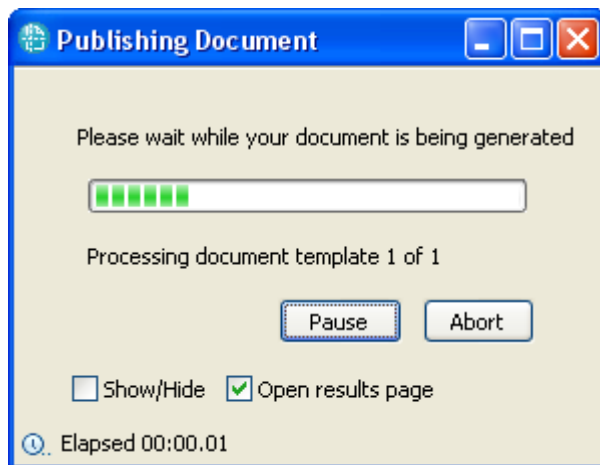


Figure 25 Generate document

While the output is being generated, a progress dialog is displayed.

NOTE TPE does not know beforehand the size of the data that will be processed so it cannot offer an accurate estimation of the time needed. In consequence the progress bar is cyclic. The messages displayed in the progress bar and console view are good indicators on the status of the document generation.

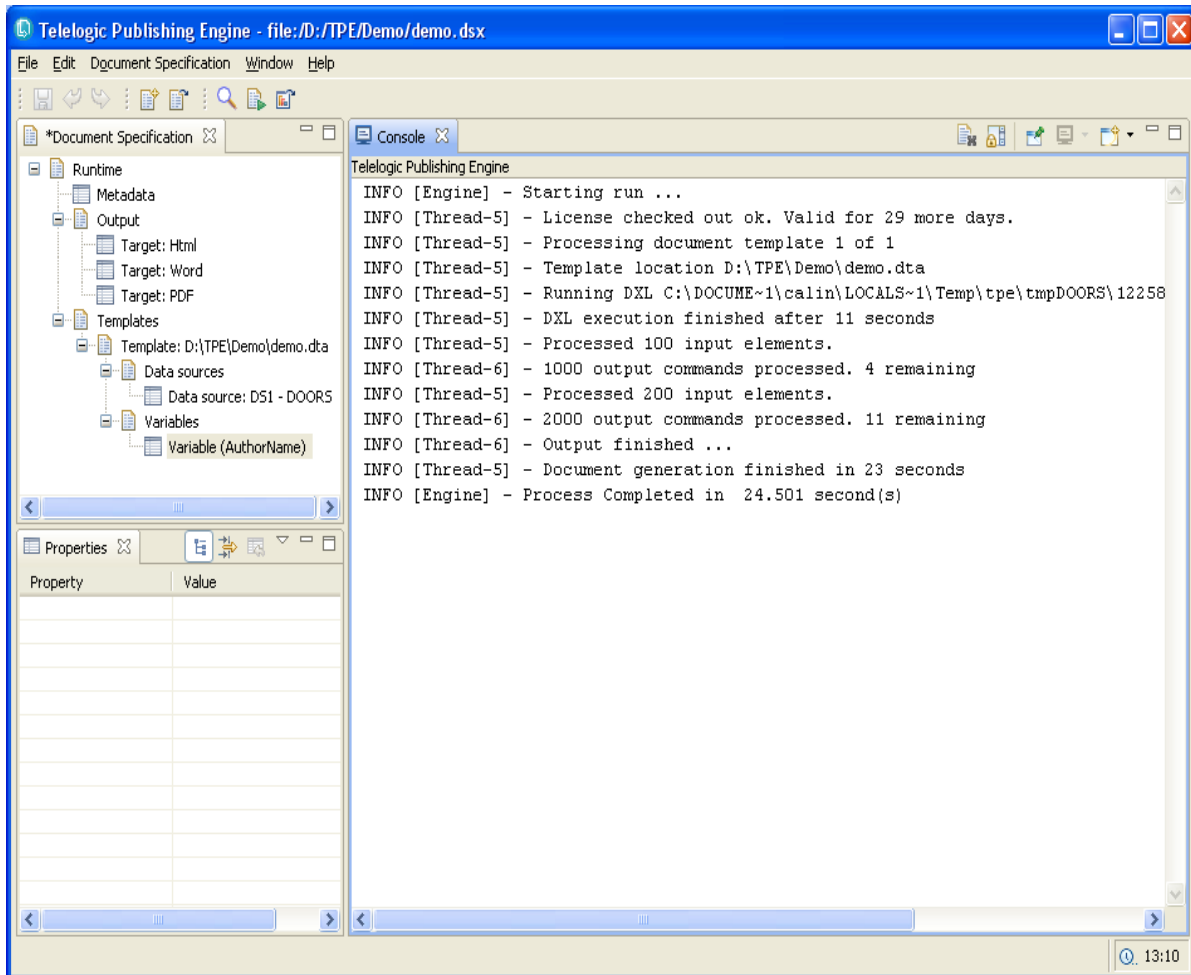


Figure 26 Console view

Pause and abort

At any time during the document generation you can request TPE to temporarily suspend the document generation process or to cancel it altogether.

NOTE In the current TPE build for DOORS extractions the document generation will wait for DXL code to finish before aborting the process. This means that the process will not instantly pause/abort if DXL execution is in progress.

Open results

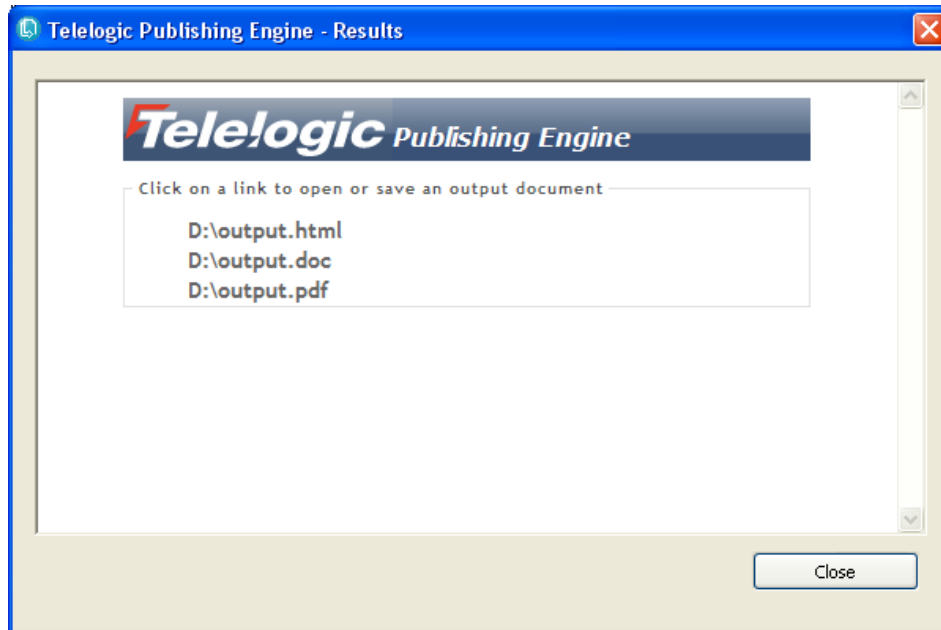


Figure 27 Results window

This function is accessible after at least one document generation process has been completed. The window displays an HTML page listing the documents generated by TPE. You can click on any of the links to open the output document (if an association is made for the file extension and the required software is installed on your machine). You can also right click to save the output in a different location.

NOTE If the output path is not specified, the output is considered to be un-configured and marked in red. TPE will generate unique names for each output type in the current user temp folder.

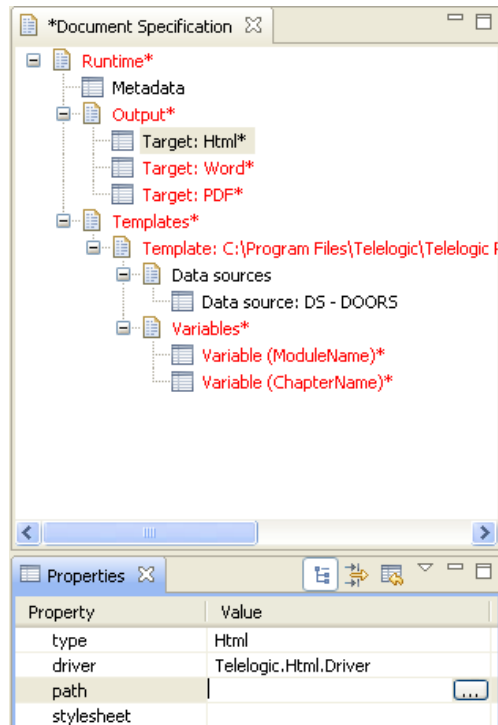


Figure 28

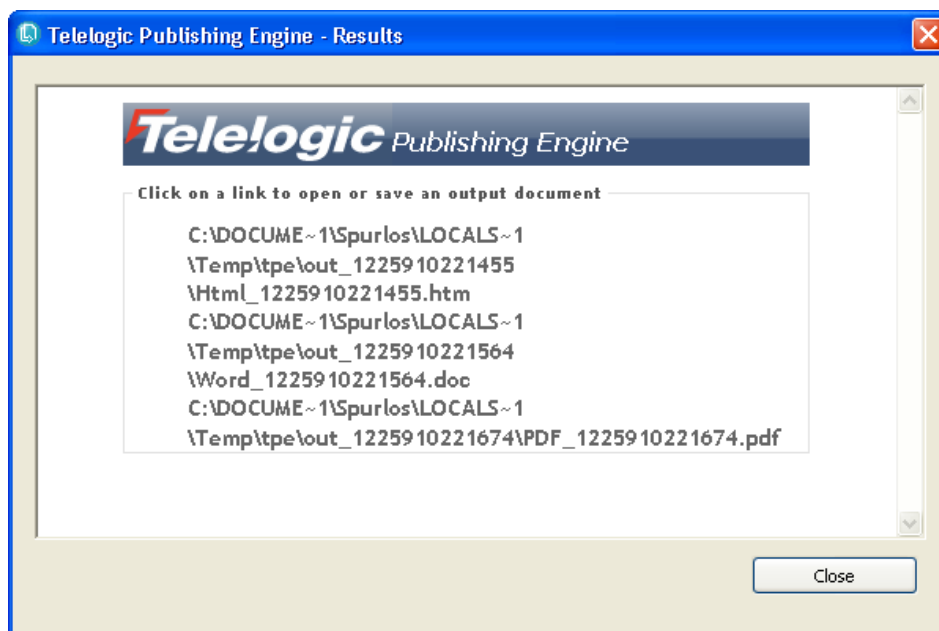


Figure 29

NOTE With the current build if you choose to save the HTML output then you need to manually copy the additional files generated (the img folder and the stylesheets).

Command line switches

TPE Launcher can be run from command line to automate the document generation process. The application supports the following command line switches:

-ds *path*

Starts the TPE Launcher and runs the provided document specification document (*path*). When the document generation process ends, the application displays the results page (if `-noresult` is not specified) and closes.

The *path* argument is the full path of the document specification.

-noresult

Must be run with the `-ds` switch.

Starts the TPE Launcher and runs the provided document specification document without opening the results window at the end of the process.

TPE Document Studio

The Document Studio component of TPE allows designing document templates.

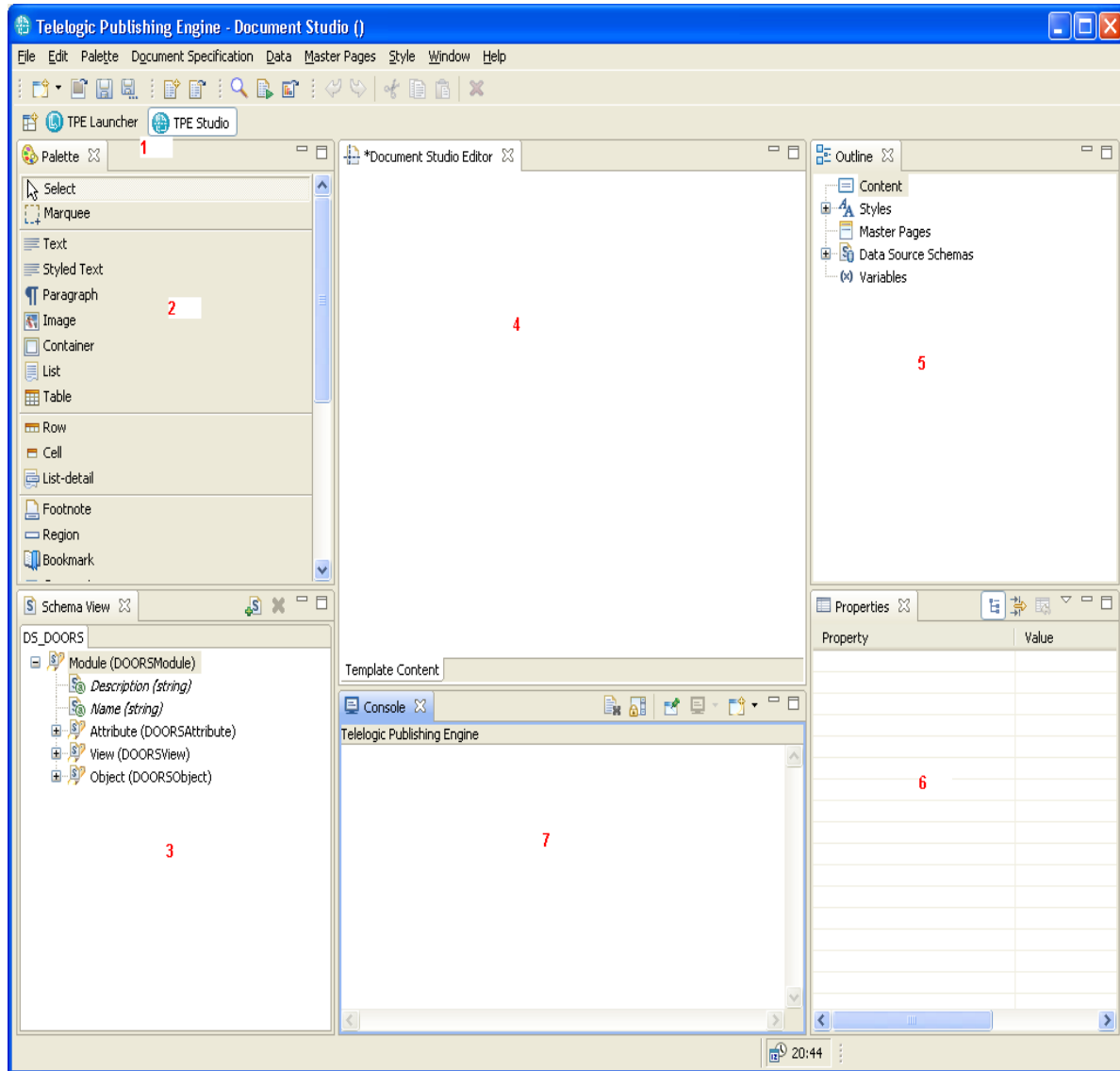


Figure 30 TPE Studio







Legend








1 – Perspective Selector








- 2 – Palette
- 3 – Data source schema view
- 4 – Editor area
- 5 – Outline
- 6 – Property page
- 7 – Console




Template elements

This section briefly describes the elements you can use when building a template. For a detailed description of elements and their properties, please consult the annex.

Icon	Name	Description	Container	Hosts data
	Text	The basic block for building templates. Renders its content with the specified formatting. The formatting is applied to the whole content. Can contain data but cannot contain other elements.	No	yes
	Styled text	A block of text that can have basic formatting at word level.	No	No
	Paragraph	A container for other elements. Cannot contain data directly. Adds a carriage return into the output then renders its child elements.	yes	no
	Image	Inserts an image into the output. The image can be taken from the template or have its path calculated at runtime.	No	yes
	Container	Similar to paragraph with 2 exceptions: <ul style="list-style-type: none"> • cannot be styled • does not add a carriage return into the output When hosted by a table can contain only row elements.	yes	No
	Table	A table can contain rows or Container elements. Creates a table in the output then renders its child elements, rows or containers.	yes	No

	Row	Creates a row in the current table (direct child or inside a container in a table), then renders its child elements (cells).	yes	No
	Cell	Creates a cell in the current row of the current table then render its child elements (any).	yes	No
	List	Creates a list in the output then renders its child elements (list items).	yes	No
	List details	Creates a list item in the current list in the output then renders its child elements (any).	yes	No
	Footnote	Creates a footnote in the current output page. The text of the footnote is the “content” of the footnote element. <i>Supported on Word output only.</i>	No	yes
	Region	Defines a region that will host elements. If an element has the “region” property specified, it will be rendered in the specified region instead of the current position in the document. You can define elements inside the region.	yes	No
	Bookmark	Defines a bookmark in the document. The name of the footnote is the “content” of the bookmark element. When generating the documents, TPE will generate a unique name for each bookmark. The name is based on the name provided at design time and a unique identifier generated at runtime.	No	Yes
	Comment	Adds a comment element into the output. The form and shape of the output comment is specific to the output format. The comment’s text is taken from the template element content. <i>Supported on Word and PDF.</i>	No	yes

	Hyperlink	Adds a hyperlink in the output. The hyperlink can point to a location inside the document or to an external location.	No	Yes
	Page Break	Adds a page break to the output.	No	No
	Table of Contents	Adds a table of contents to the output.	No	No
	Table of Tables	Adds a table of tables to the output. You need to add Table Captions in your document template for this table to contain anything. <i>Supported on Word output only.</i> <i>You need to update all fields in the Word document in order to see/update the value of the field.</i>	No	No
	Table of Figures	Adds a table of tables to the output. You need to add Figure Captions in your document template for this table to contain anything. <i>Supported on Word output only.</i> <i>You need to update all fields in the Word document in order to see/update the value of the field.</i>	No	No
	Field	Adds a generic Word field element. You can type any valid Microsoft Word code in the “field code” property. <i>Supported on Word output only.</i> <i>You need to update all fields in the Word document in order to see/update the value of the field.</i>	No	No
	Page Number	Adds a page number in the output. <i>Supported on Word and PDF output only.</i>	No	No

	Table Caption	<p>Adds a table caption. While you cannot specify dynamic content into a table caption, any subsequent text element will be concatenated to it.</p> <p><i>Supported on Word output only.</i></p> <p><i>You need to update all fields in the Word document in order to see/update the value of the field.</i></p>	No	No
	Figure Caption	<p>Adds a table caption. While you cannot specify dynamic content into a table caption, any subsequent text element will be concatenated to it.</p> <p><i>Supported on Word output only.</i></p> <p><i>You need to update all fields in the Word document in order to see/update the value of the field.</i></p>	No	No
	Include file	<p>Includes the specified file in the output as an "INCLUDETEXT" field.</p> <p><i>Supported on Word output only.</i></p> <p><i>You need to update all fields in the Word document in order to see the included file.</i></p>	No	Yes

Components

Palette

The palette displays the elements (tools) that can be used in the template. Two special tools are provided:

- Select – allows selecting elements in the editor area. A single click selects the clicked element while Ctrl + Click performs adds the clicked element to the existing selection
- Marquee – allows to select multiple elements

To add a template element, click on the appropriate entry in the palette and then click in the template in the desired location.

By default the palette is sticky, meaning that a palette element remains selected until you explicitly select a new tool. You can change the way the palette works from the preferences menu.

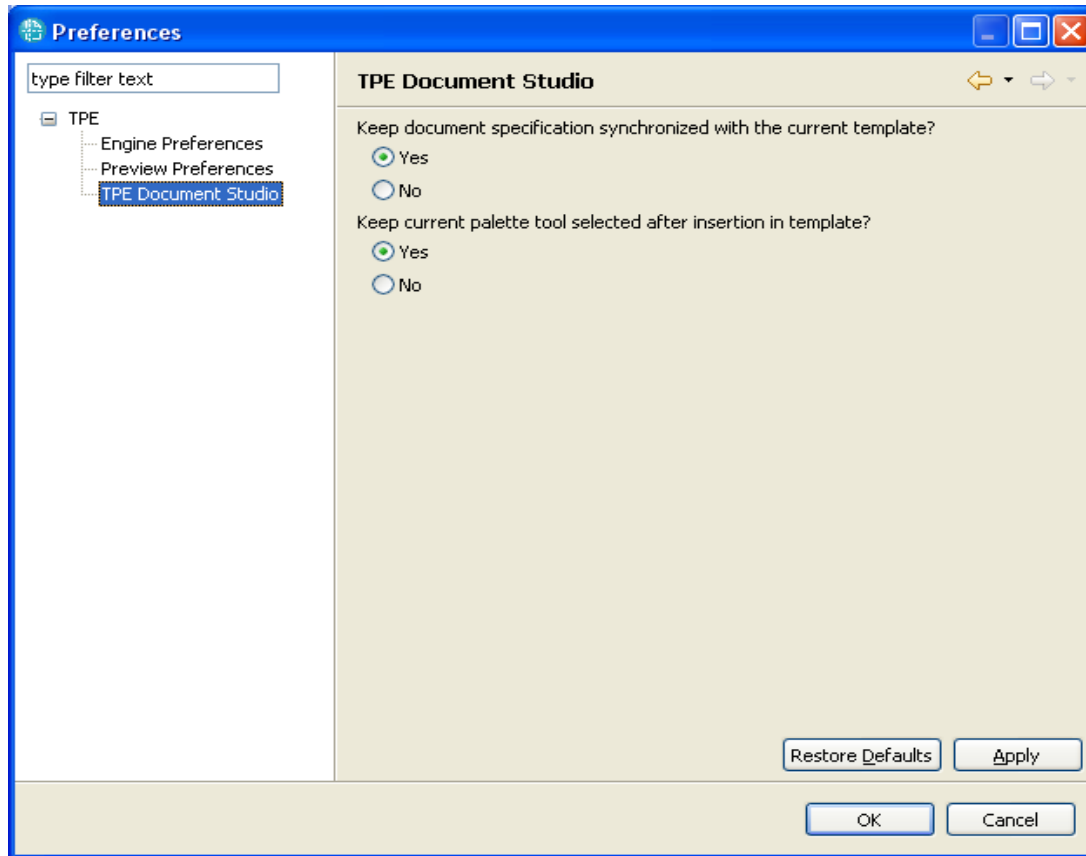


Figure 31

Schema view

This view displays all the data schemas in the template. Each schema is displayed in its own tab in this view.

Editor area

This is the core of Document Studio. The template's visual content and the master pages' content are created here. There is one tab for the template's content and one for each master page in the document template.

Outline

Displays, in a tree structure, all the visual and non visual content.

Properties

Displays and allows changing the properties of the selected element in the editor area or the outline.

Content

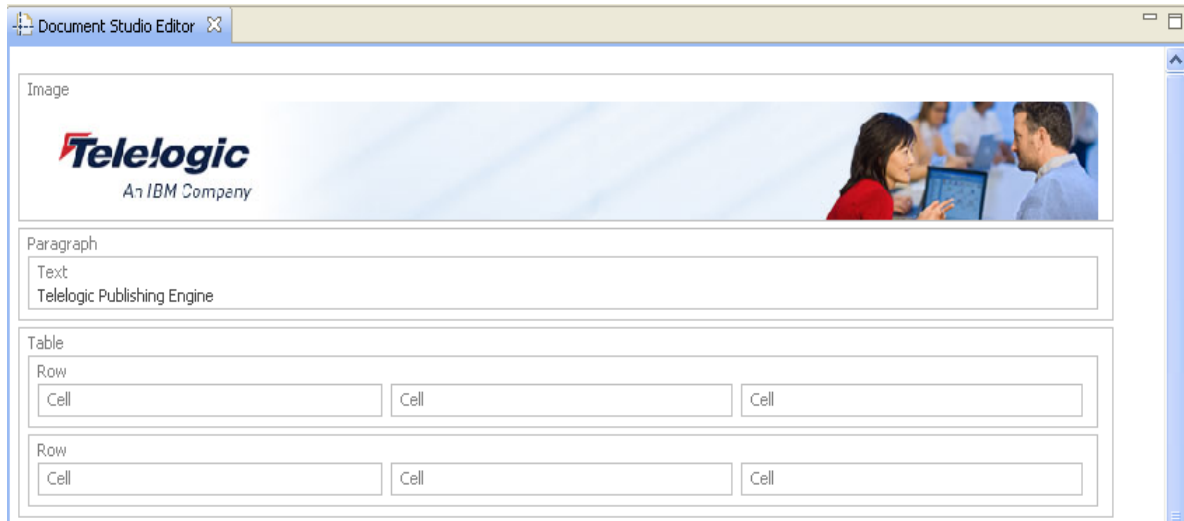


Figure 32 Editor Area

Adding elements

You add elements in the editor area by selecting the appropriate tool in the palette and clicking in the desired location in the editor area. Each element has its own set of rules when it comes to what elements it can contain and be contained by. See the annex for details.

Editing element properties

All the element's properties can be edited through the property page. There are two different types of properties:

- data related properties – related to queries set on the element, conditions etc
- formatting properties – define the way the element will be rendered in the output format

The formatting properties are grouped further as font properties (font family, color, size), alignment properties (centered), border properties etc.

NOTE The available formatting properties depend on the element type.

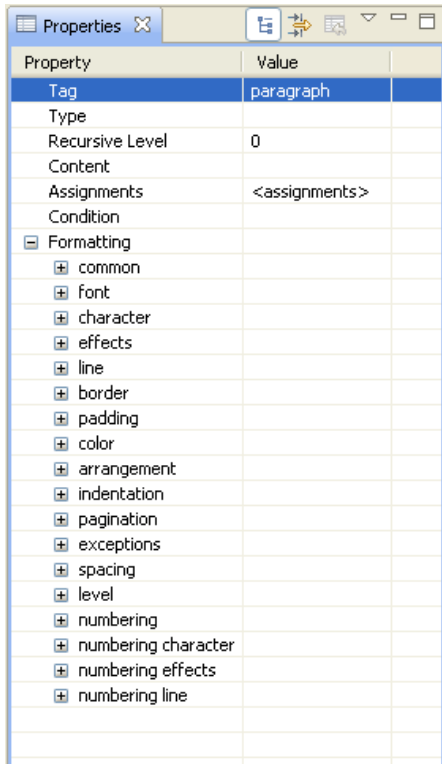


Figure 33 Properties page for a paragraph element

Formatting properties

Some of the properties (such as font family) accept any value provided by the user. Other properties (such as paragraph alignment) accept values from a list of possible options. TPE Studio assists the user in providing the right value through a tooltip describing the property value domain and through the property value editor.

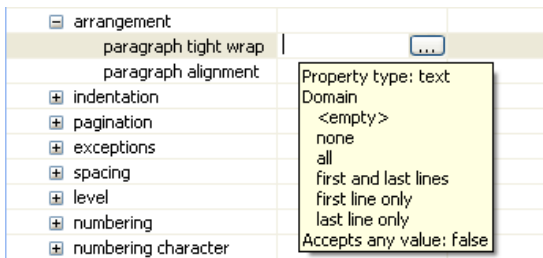


Figure 34 Tooltip for property value

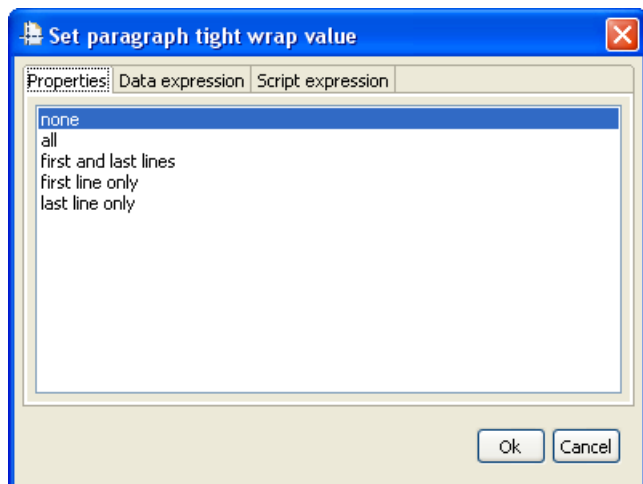


Figure 35 Value editor

NOTE You can type the value directly in the property editor or select it from the expression editor list.

NOTE Some values accept both user-provided values and values selected from the list.

NOTE Invalid properties values (typed by the user) are not reported by Studio but are ignored during the generation process.

Calculated properties

Besides the static values TPE supports dynamic values for properties. The dynamic values are either data attributes extracted from the data sources, template variables or the result of evaluating JavaScript expressions. See the data section for more details on calculated properties.

Data

As previously mentioned, a TPE document template operates only with data schemas and not with actual data sources.

Adding data source schemas

You can add new schemas in two ways:

- Through the “New data source” operation available in the main menu or the context menu in the outline view

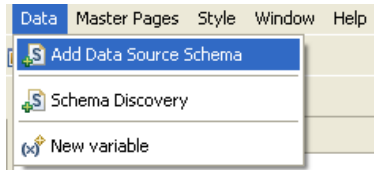


Figure 36 Add data source schema in the main menu

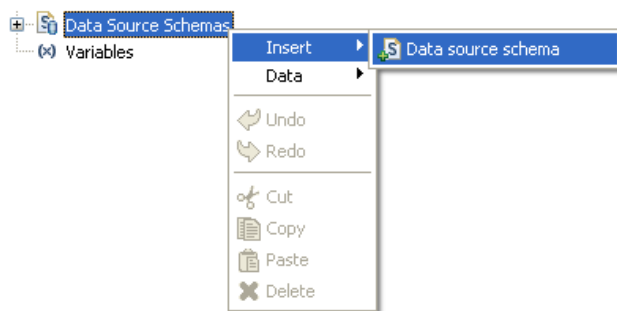


Figure 37 Insert Data Source Schema from Outline context menu

- Through the schema discovery wizard

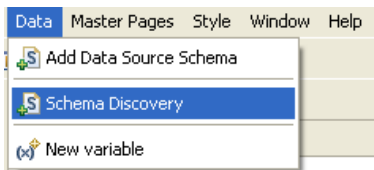


Figure 38 Schema discovery wizard

Once added, the data source schemas are visible in the outline view and in the schema view.

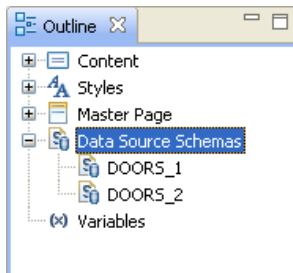


Figure 39 Data source schemas in the outline view

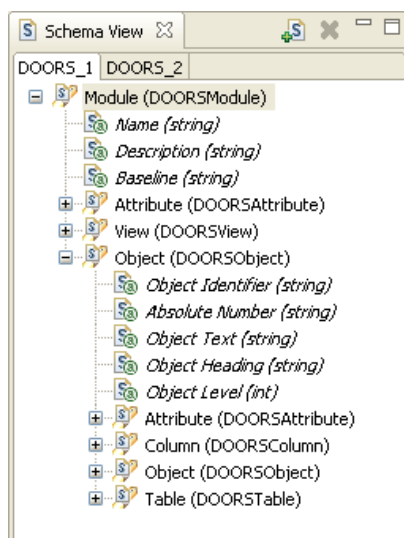


Figure 40 Data source schemas in the schema view

Queries

A query defines what data is to be extracted from the data source. You can assign a query to any template element. A template element having a query set for it will be duplicated for each data element extracted from the data source.

You assign a query to a template element by dragging an element from the schema view onto the template element.

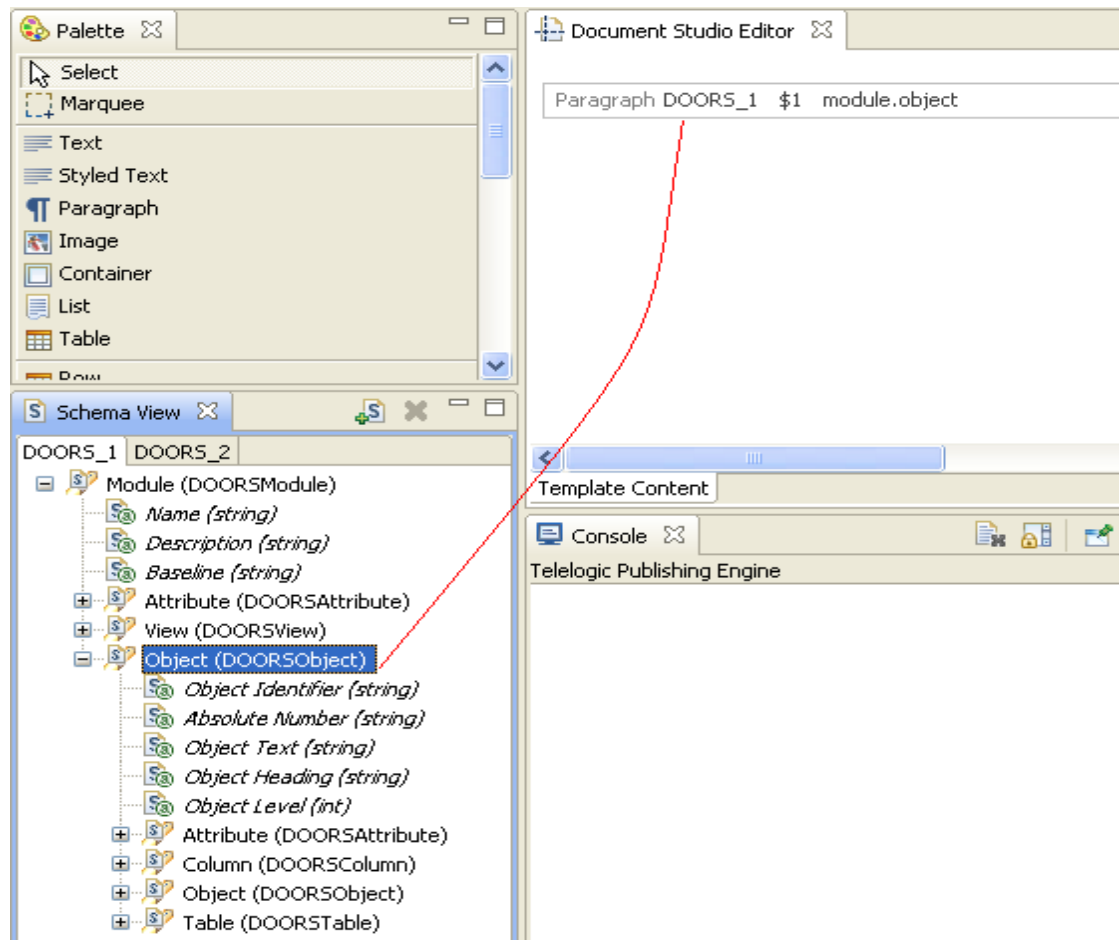


Figure 41 Assigning a query to a template element

Once a query is assigned to a template element, you can use the attributes of the query type (the element dragged from the schema view) anywhere in the template element, including its children.

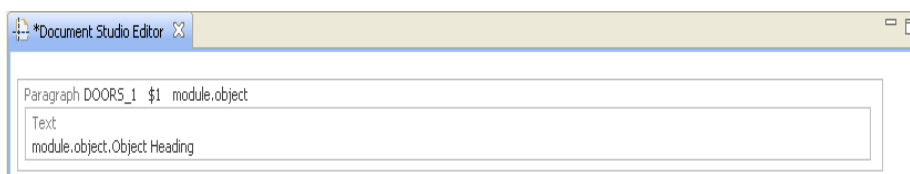


Figure 42 Using data attributes

Using data attributes in template elements

You can specify which data attributes to display in two ways:

- Drag the attribute on top of a text element

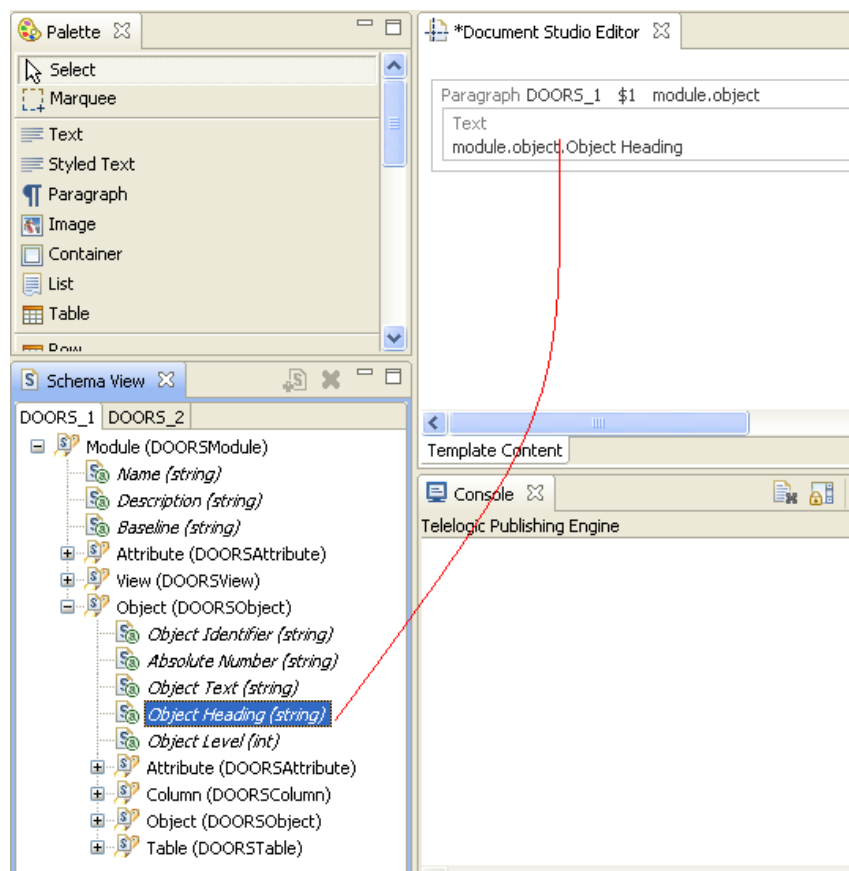


Figure 43 Dragging a data attribute

- Edit the text element's content using the expression editor

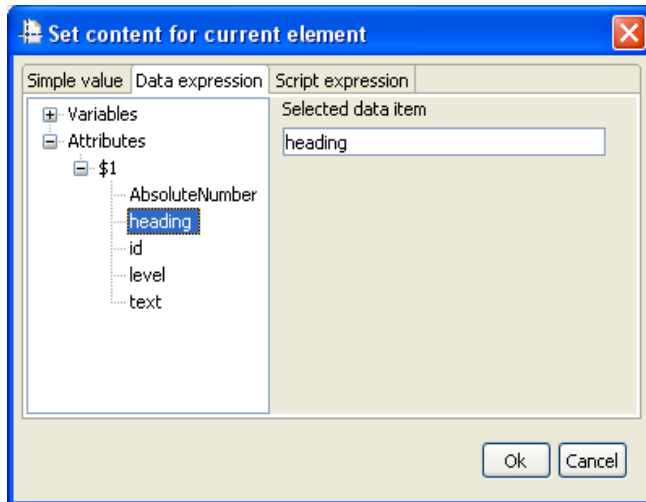


Figure 44 Expression Editor for the content of a text element

NOTE Only the **text** and **hyperlink** elements can be used to render data attributes. All the other template elements can use data attributes for conditional formatting only.

Filter

A query can have a filter defined for it. A filter can be expressed using the native data source means (if available and textual) or via a TPE Filter. A native filter is interpreted by the data source. A TPE filter is processed by TPE as it extracts data from the data source.

See Data section for details on using filters.

NOTE Whenever possible it is recommended to use native filter as it should yield better document generation times than when a TPE filter is used.

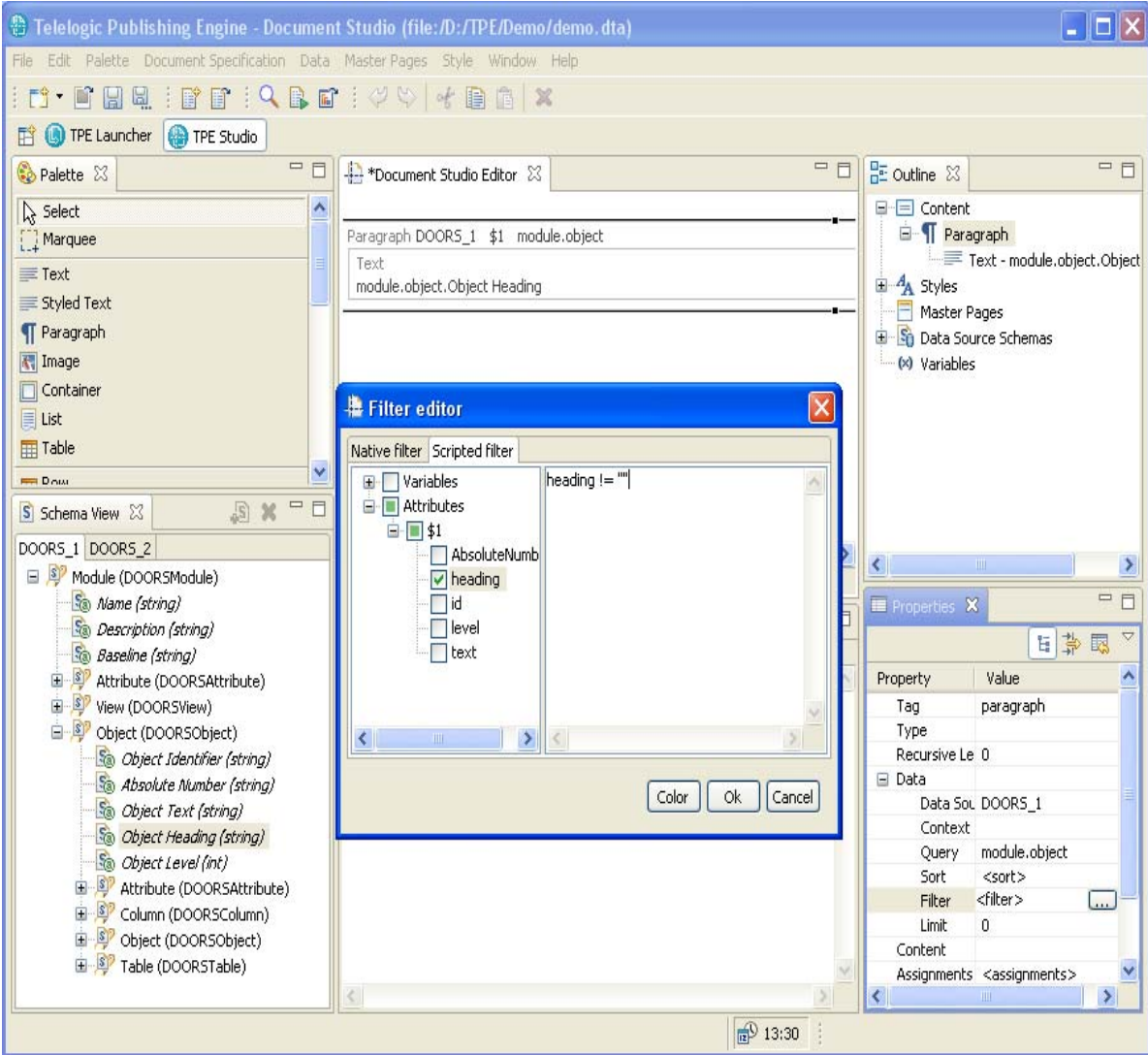


Figure 45 TPE Filter

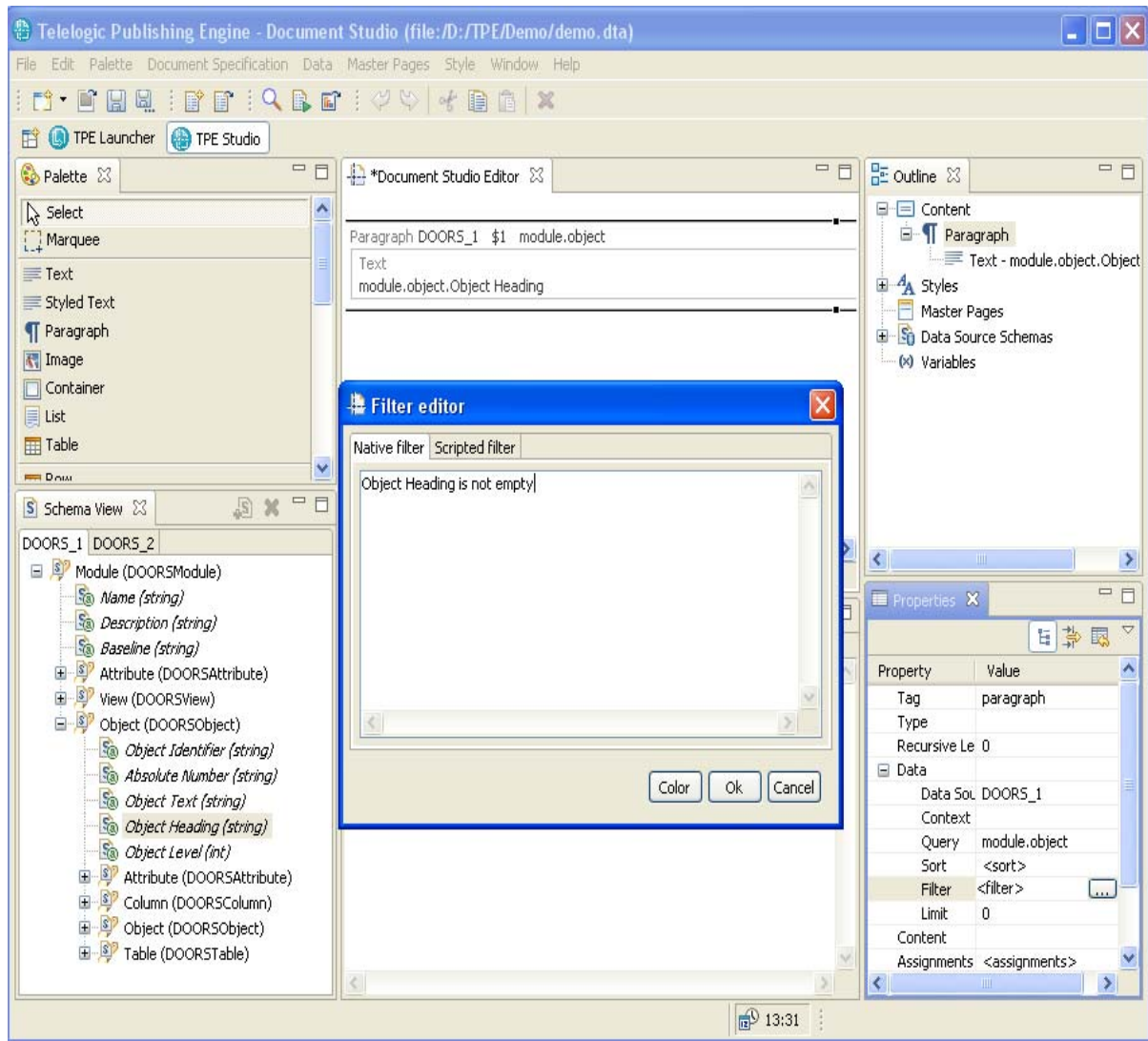


Figure 46 DOORS Native filter

NOTE TPE cannot check the native filter correctness. It is the user's responsibility to ensure the native filter is correct. For DOORS Data Sources the native filter should be first tested in DOORS then copied from the "Description" window of the Filter properties.

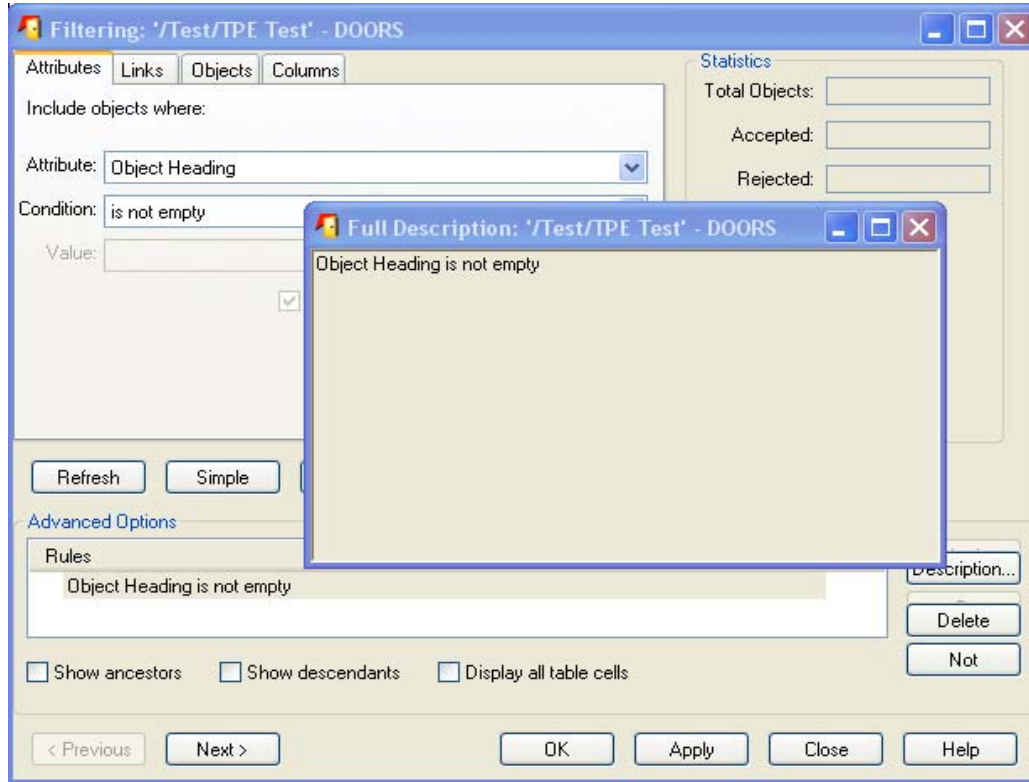


Figure 47 DOORS Filter text

Sort

A query can have a sort defined for it. A sort can be expressed using the data source's native sort (if available and textual) or as TPE Sort. A native sort is interpreted by the data source. A TPE sort is processed by TPE as it extracts data from the data source.

See Data section for details on using sort.

NOTE Whenever possible it is recommended to use a native sort as it should yield better document generation times than when a TPE sort is used.

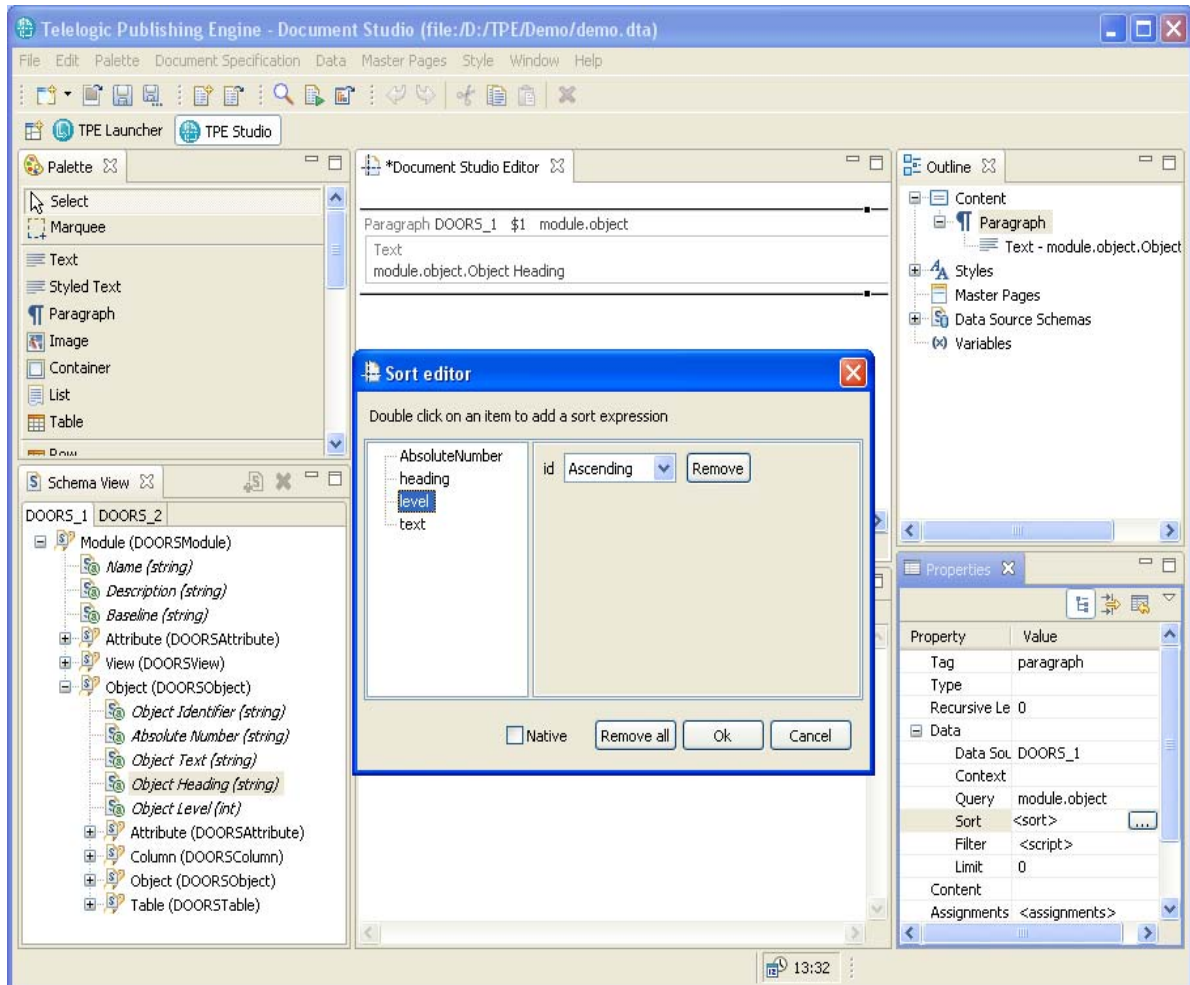


Figure 48 TPE Sort

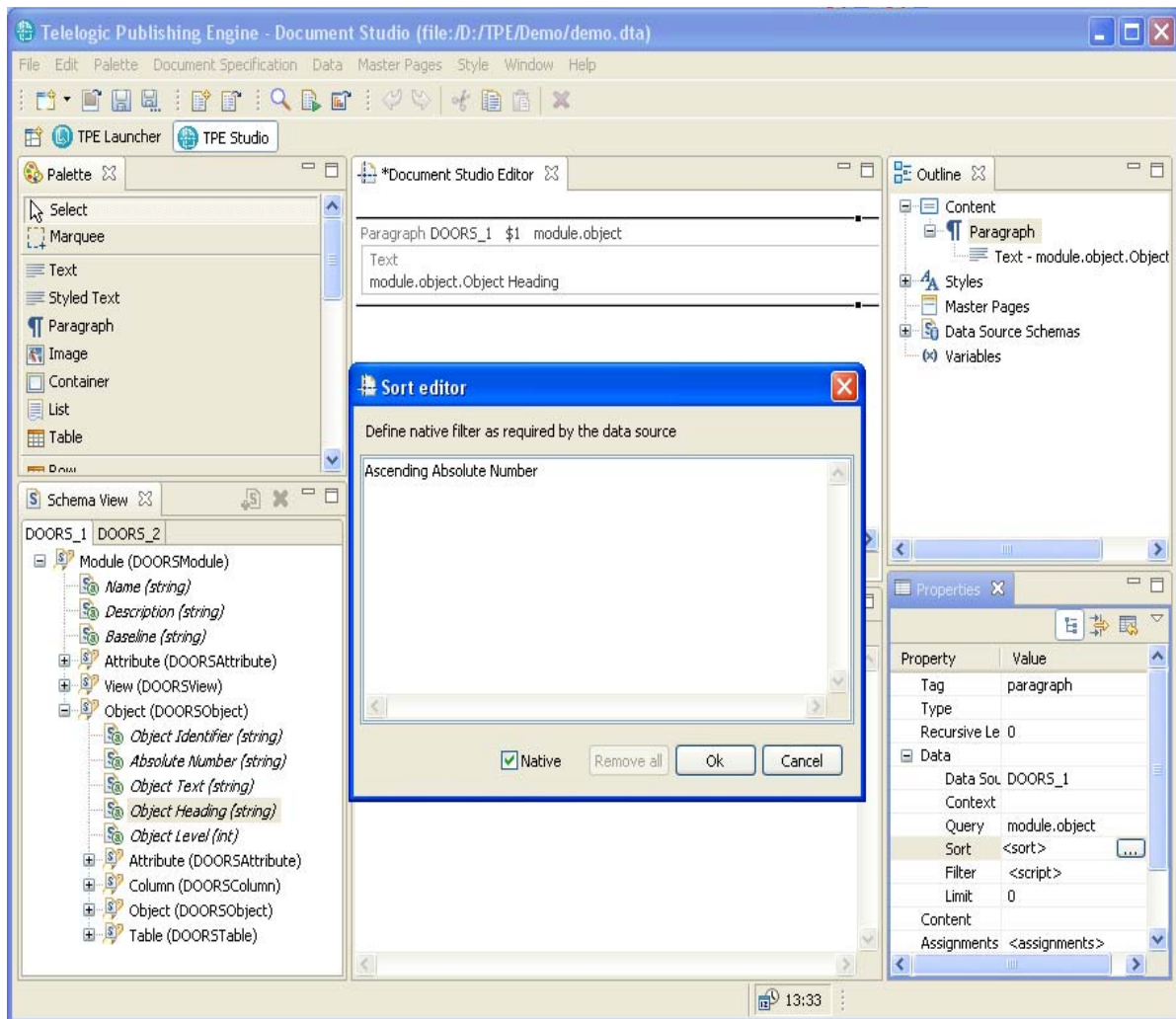


Figure 49 DOORS Native Sort

NOTE TPE cannot check the correctness of a native sort. It is the user's responsibility to ensure the native sort is correct. For DOORS Data Sources the native sort should be first tested in DOORS.

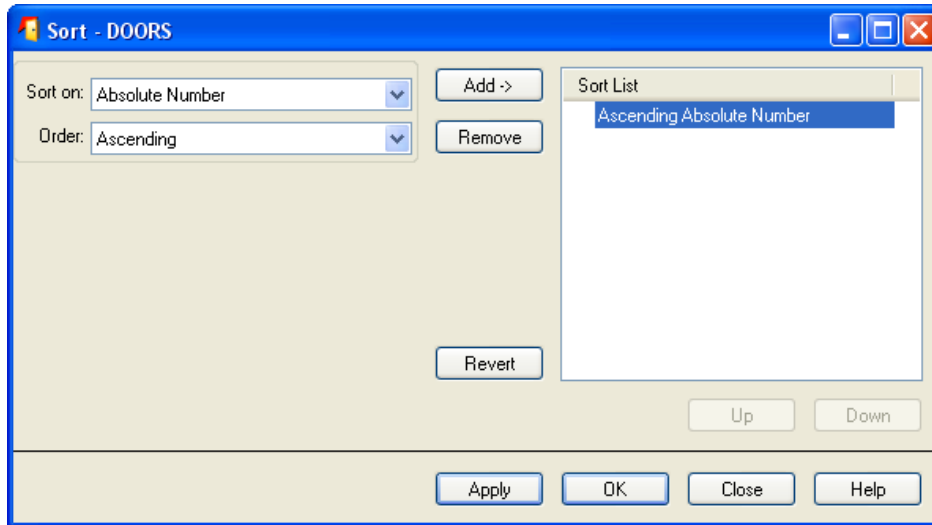


Figure 50 DOORS Sort Window

Contexts

You can use only the attributes valid in the current context (see the Data/Queries and contexts for details). TPE studio will prevent you from dragging attributes not visible in the current context.

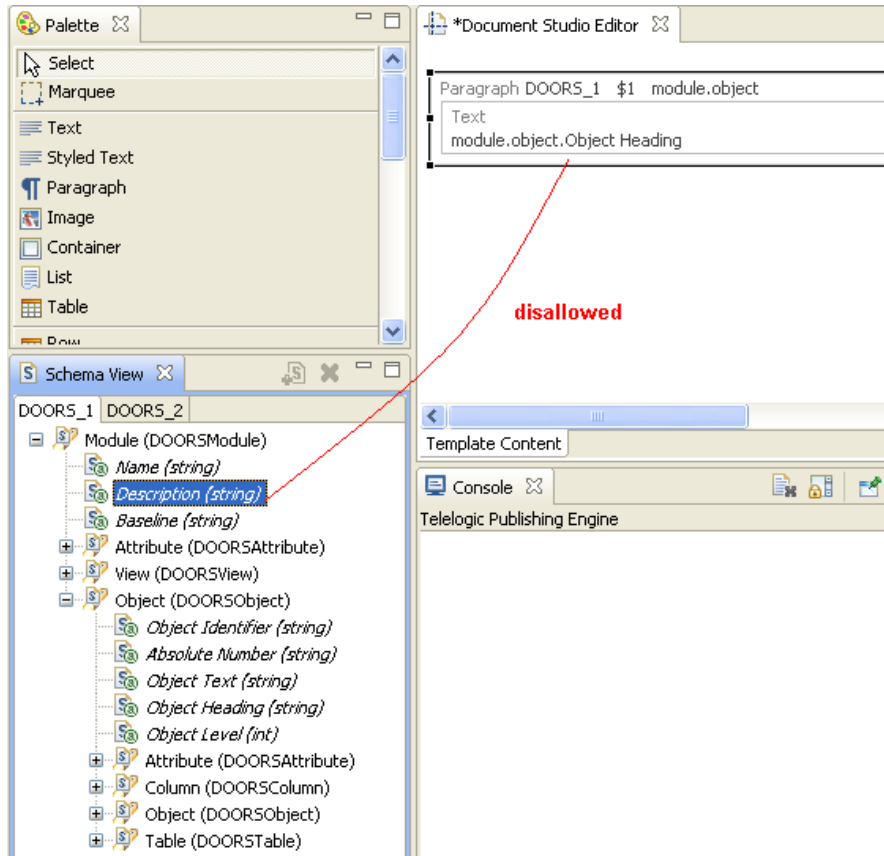


Figure 51 Disallowed drag attribute operation

In the above example the “Description” attribute is not part of the “Object” type in the schema hence TPE will not allow dragging it in the selected text element from the editor area.

Nested queries

You can nest queries by assigning queries to children template elements.

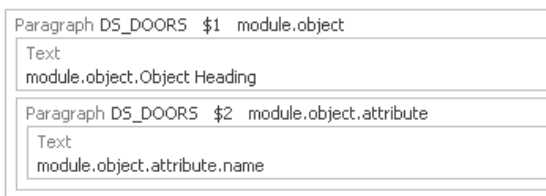


Figure 52 nested queries

When a query is dragged onto an element that has at least 1 parent element with a query, TPE will calculate if the dragged query can be executed in the context parent's element query. All the queries that can serve as context are displayed for selection. The list displays the id of the query as well as its textual representation.

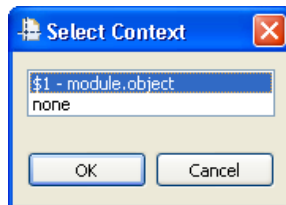


Figure 53 Context selection

In the example above, `module.object` can serve as context for `module.object.attribute` hence the identified of the query is displayed in the list.

Selecting **none** for the context will result in two nested but unrelated queries.

In the above example, setting the second query's context to `$1` will produce the following output:

- a set of paragraphs containing the heading of each object in the module
 - a list of paragraphs with the attribute names for the *current object* in query `$1`

In the above example, setting the second query's context to **none** will produce the following output:

- a set of paragraphs containing the heading of each object in the module
 - a list of paragraphs with the attribute names for *all the objects*

Advanced usage of data attributes

The data attributes can be used for more than displaying the raw data extracted from the data source.

Calculated values

Using the expression editor you can create JavaScript snippets to process the data attributes as needed. Examples of such processing is combining data attributes, trimming white spaces, transforming numeric values into textual descriptions etc.

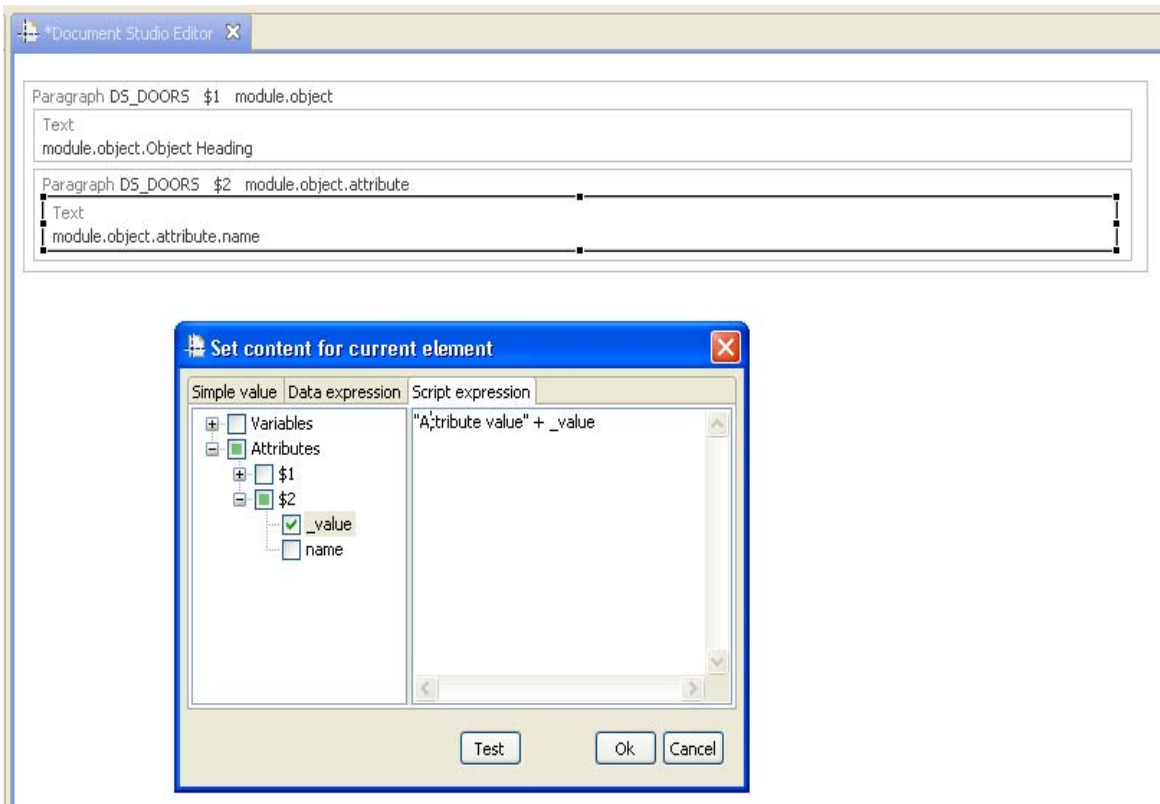


Figure 54 adding some text to a data attribute

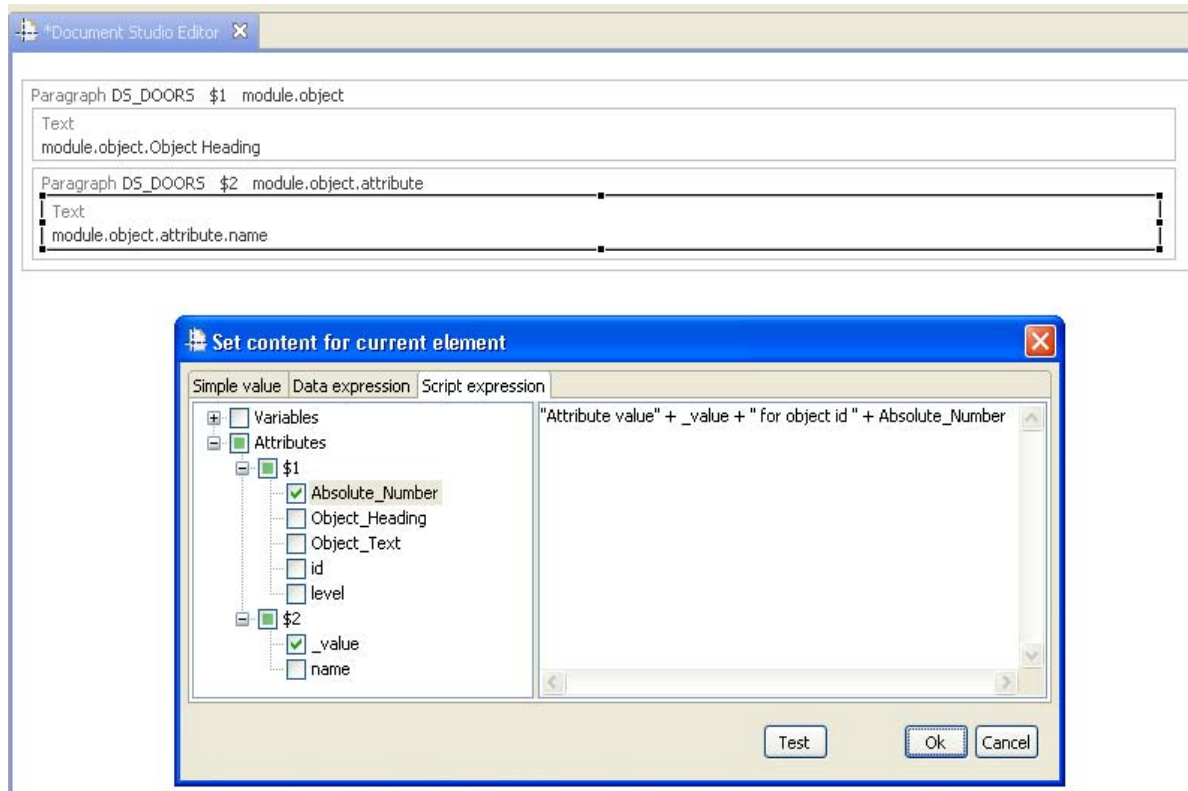


Figure 55 combining attribute values

NOTE When using an attribute in a script expression you need to mark it as used in the attributes tree in the Expression Editor. The expression editor will report errors for attributes used but not declared as used.

NOTE You can use any data attribute from the current context (the element's query attributes and the attributes from all the parent elements' queries).

Conditions

Using expressions based on data attributes or template variables you can define conditions for when an element should be rendered. The condition is a JavaScript expression that returns a *boolean* value.

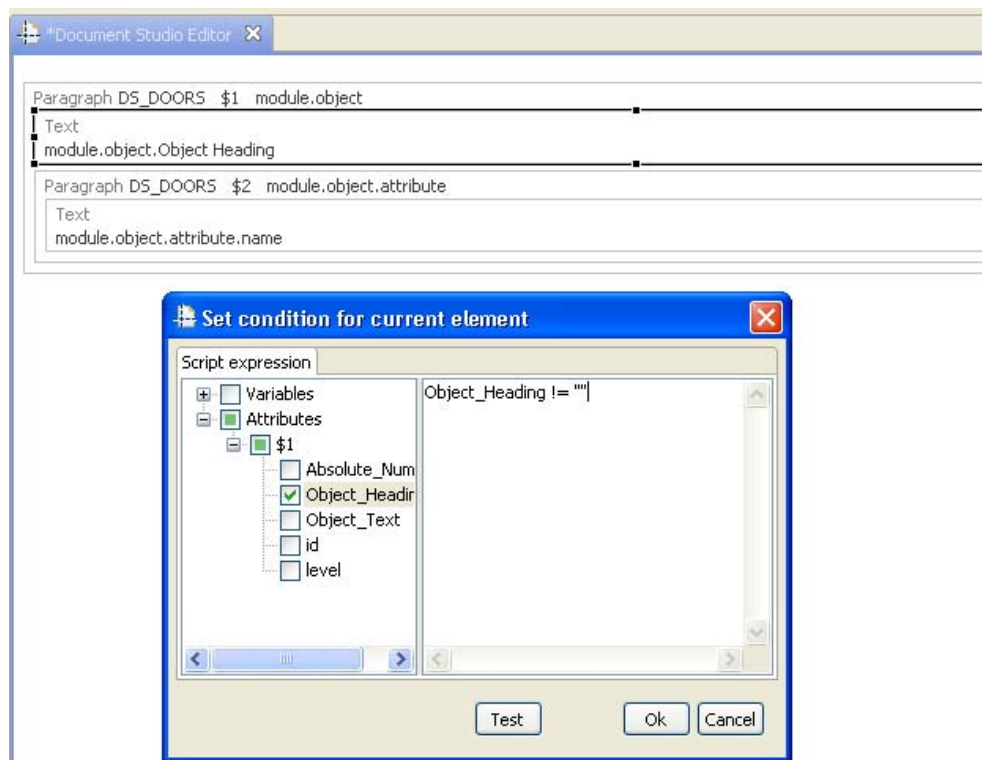


Figure 56 Setting a condition for a template element

In the above example the “heading” attribute is displayed only if not empty to prevent empty paragraphs appearing in the output document.

Filter vs. Conditions

While looking similar in purpose the Filters and Conditions are two different mechanisms that serve different purposes.

A **filter** can be evaluated as data is retrieved from the data source.

A **condition** is evaluated only after the data has been extracted from the data source. So while you can use conditions instead of filters, using filters yields better performance as TPE will only process a subset of the data.

A **condition** is evaluated only once for an element even if that element is a query. This is why you cannot use the current query as a context for a condition.

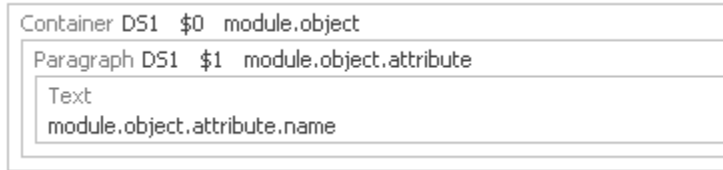


Figure 57

In the above example the 3 elements can use the following contexts in their conditions:

- **container** – none
- **paragraph** – attributes of the *module.object* query
- **text** - attributes of the *module.object* query and attributes of the *module.object.attribute* query

As you can see, the container element cannot use attributes of the *module.object* to define its condition. If you want this kind of behavior you need to apply a filter (scripted or native) onto the container element.

Conditional formatting

You can use expressions to define formatting properties based on data attribute values. The conditional formatting are similar to element conditions but their return values depend on the property type.

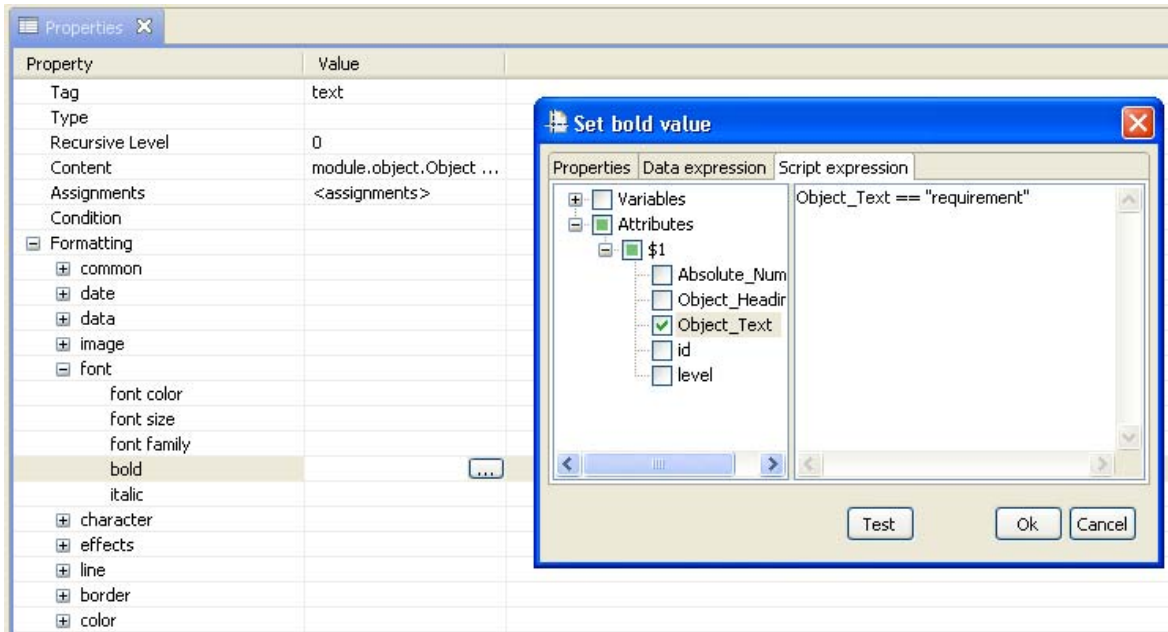


Figure 58 Conditional "bold" property

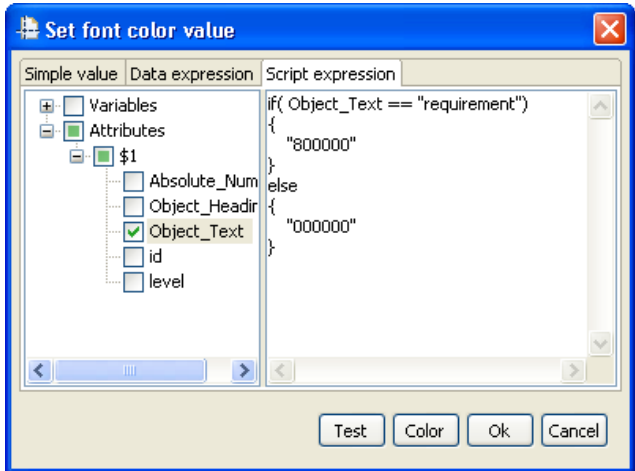


Figure 59 conditional "color" property

NOTE The condition verifies if the object text equals requirement and not if it contains the object requirement.

NOTE Assuming the condition is changed to check if the object text contains “requirement”, the formatting will still be applied to the whole object text and not only to the requirement word.

NOTE For color properties you can use the color picker to get the color code.



Figure 60

NOTE The color picker is available in the Simple value and Script value tabs of the expression editor.

Styles

TPE allows creating styles and applying them to template elements. Unlike the styles available in a word processor application such as Microsoft Word, the TPE styles are entirely defined by the user. When creating a style you need to define the list of properties that make the style and then provide values for them.

Creating a style

You create styles using the style wizard.

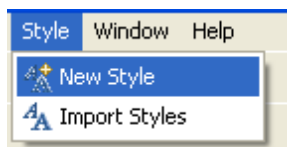


Figure 61 Style menu

Define the style name

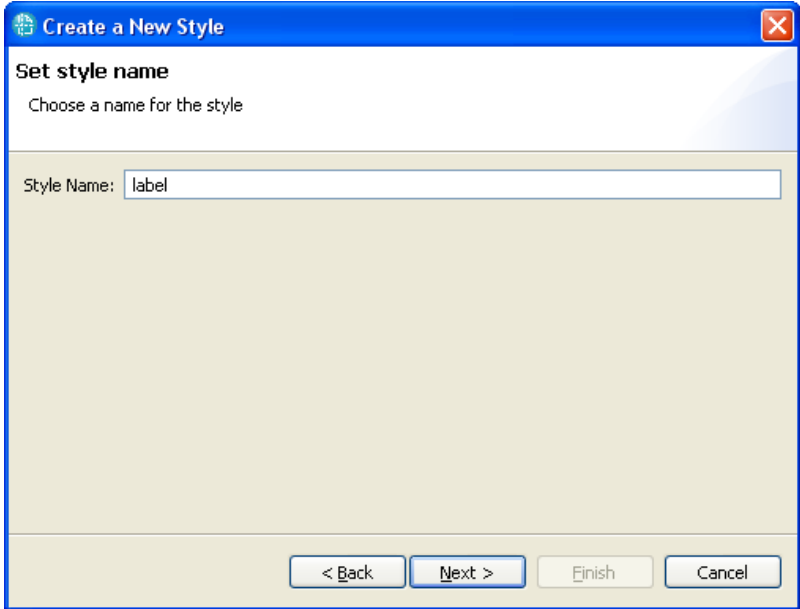


Figure 62

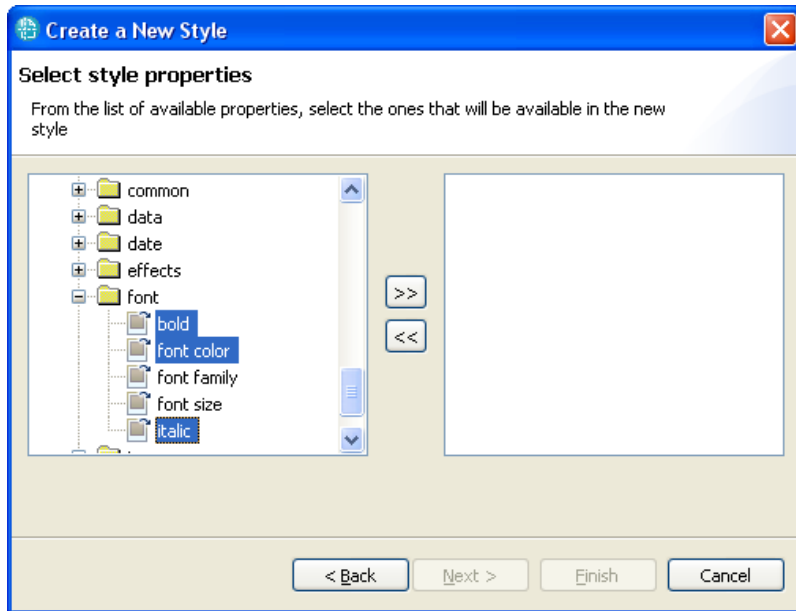
Select the style's properties

Figure 63

Add the properties to the style

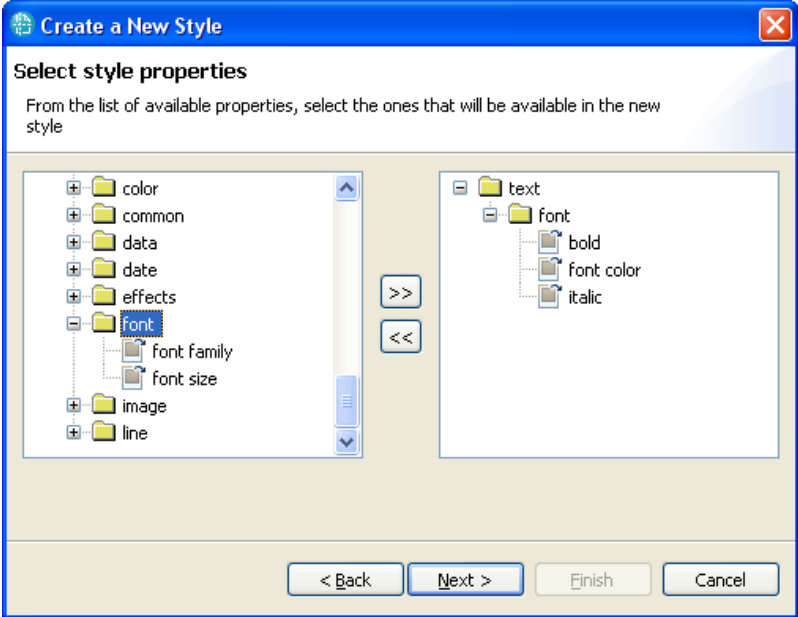


Figure 64

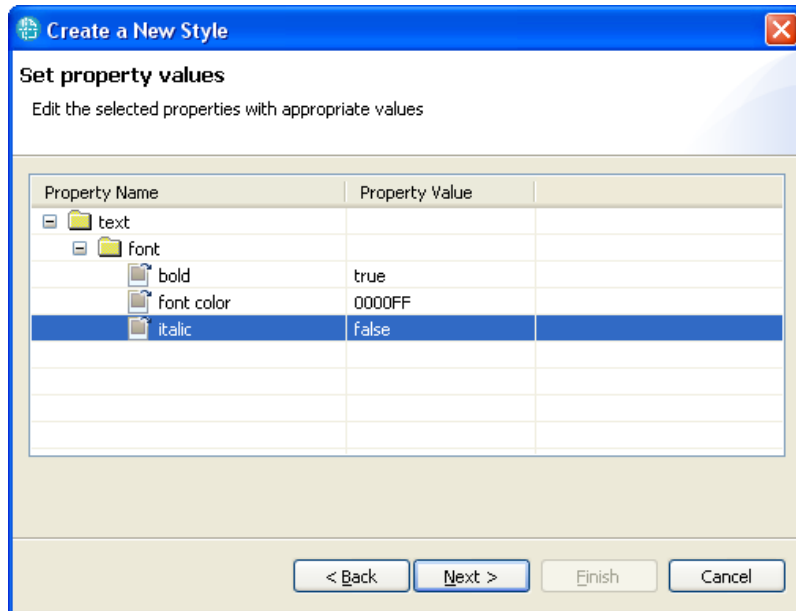
Define the values for the style's properties

Figure 65

Style summary

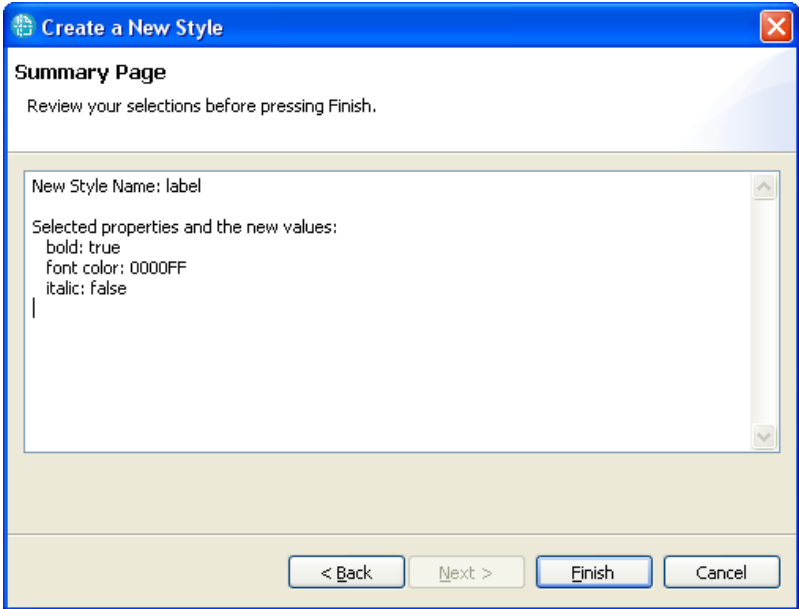


Figure 66

Editing a style

Once you have created a style you can add new properties to it at any time using the "Edit style" function from the context menu of the Outline view. The same wizard will open once again allowing you to add/remove properties from the style.

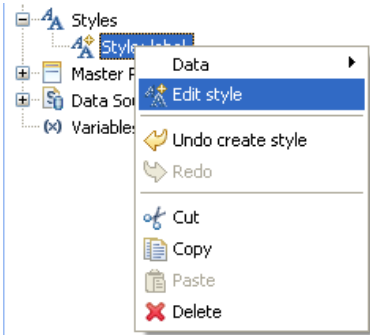


Figure 67

Changing style properties

You can change the values for the style's properties using the wizard or through the property pages. The second option should be more straightforward.

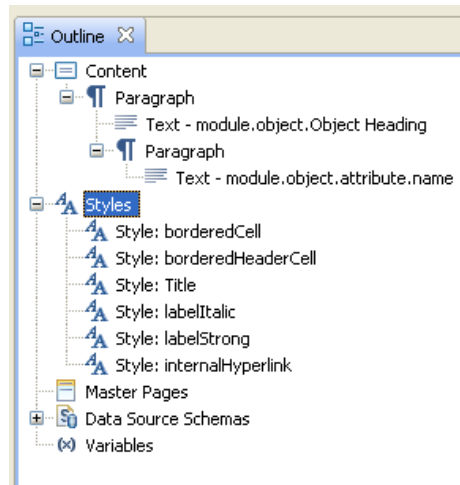


Figure 68 Editing style property values

Applying a style

To apply a style you can drag it from the outline page and drop it on the desired element.

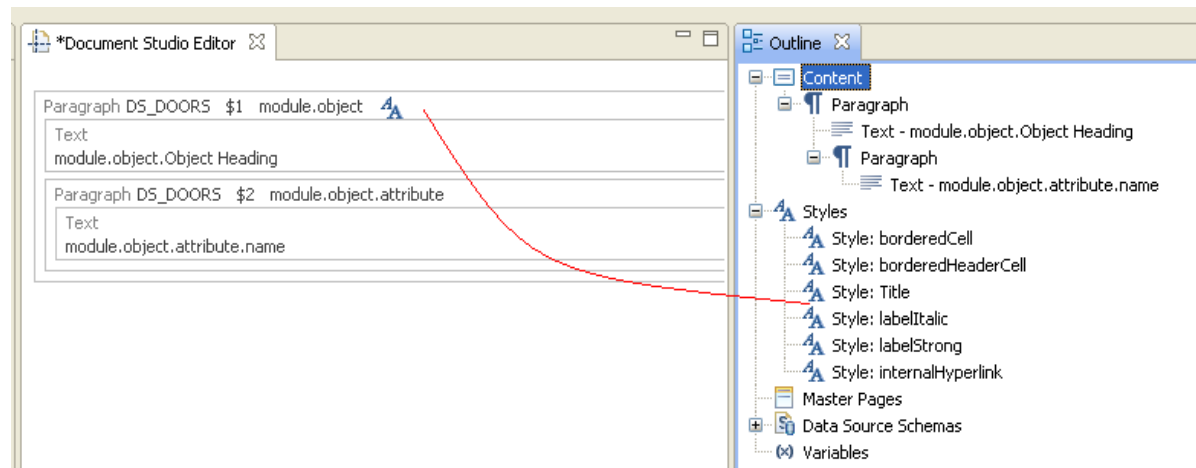


Figure 69 dragging a style on a template element

You can also type the style's name in the "style name" property of the desired element.

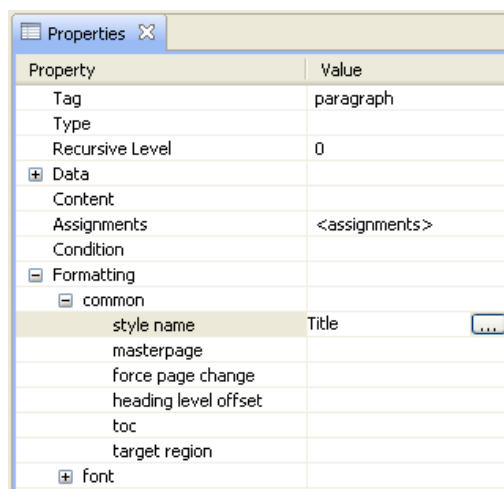


Figure 70 Set the style name through the property pages

Using external styles

TPE can use styles that are not defined in the document template as long as they are provided in the stylesheet. All you need to do is type the name of the style in the element's property page as you would do with a TPE style. When the output is generated TPE will try to use that style if present in the stylesheet. If a style with that name is not found, no style will be applied to that element.

Style precedence

At document generation time, if TPE finds a style with the same name as a style defined in the document template, the stylesheet style is used. This applies to Word output only.

For HTML output you can manually remove the TPE style from the generated stylesheet so that the provided stylesheet is used.

Style inheritance

If a style is applied to a template element that also has values set for individual formatting properties, the element's changes will override the style ones if such an overlap exists.

Reserved style names

You can use any name for the style you define in TPE excepting styles named 1,2,3,4,5,6,7,8,9. These are reserved style names in TPE that are matched to heading styles in the output formats. 1 is translated to "Heading 1" for word output and "H1" for HTML.

Master pages

The content of a document template has no concept of pages; it is a single continuous flow. The reason is simple: the document template defines data placeholders so it is not possible to know at design time when a page starts.

However that does not mean TPE doesn't support the page concept. TPE handles pagination through "Master Pages". A master page is defined by a header element, a footer element and the page's properties such as page orientation, page borders etc.

A master page is edited in the same way as you edit the template's content with two major restrictions:

- You cannot put queries or data attributes in the master page (though the same result is possible via alternate means)
- You can add elements only inside the Header/Footer elements

Adding a master page

You can add a masterpage from the main menu or from the outline view's context menu.

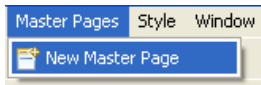


Figure 71 Adding a master page from the main menu

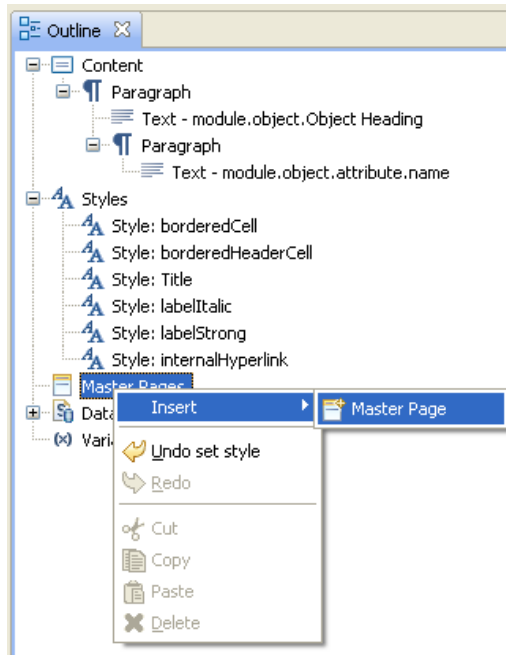


Figure 72 Adding a master page from outline view

Once added the master page is visible in the outline view. Its properties can be edited through the property page and a new tab is added in the editor area.

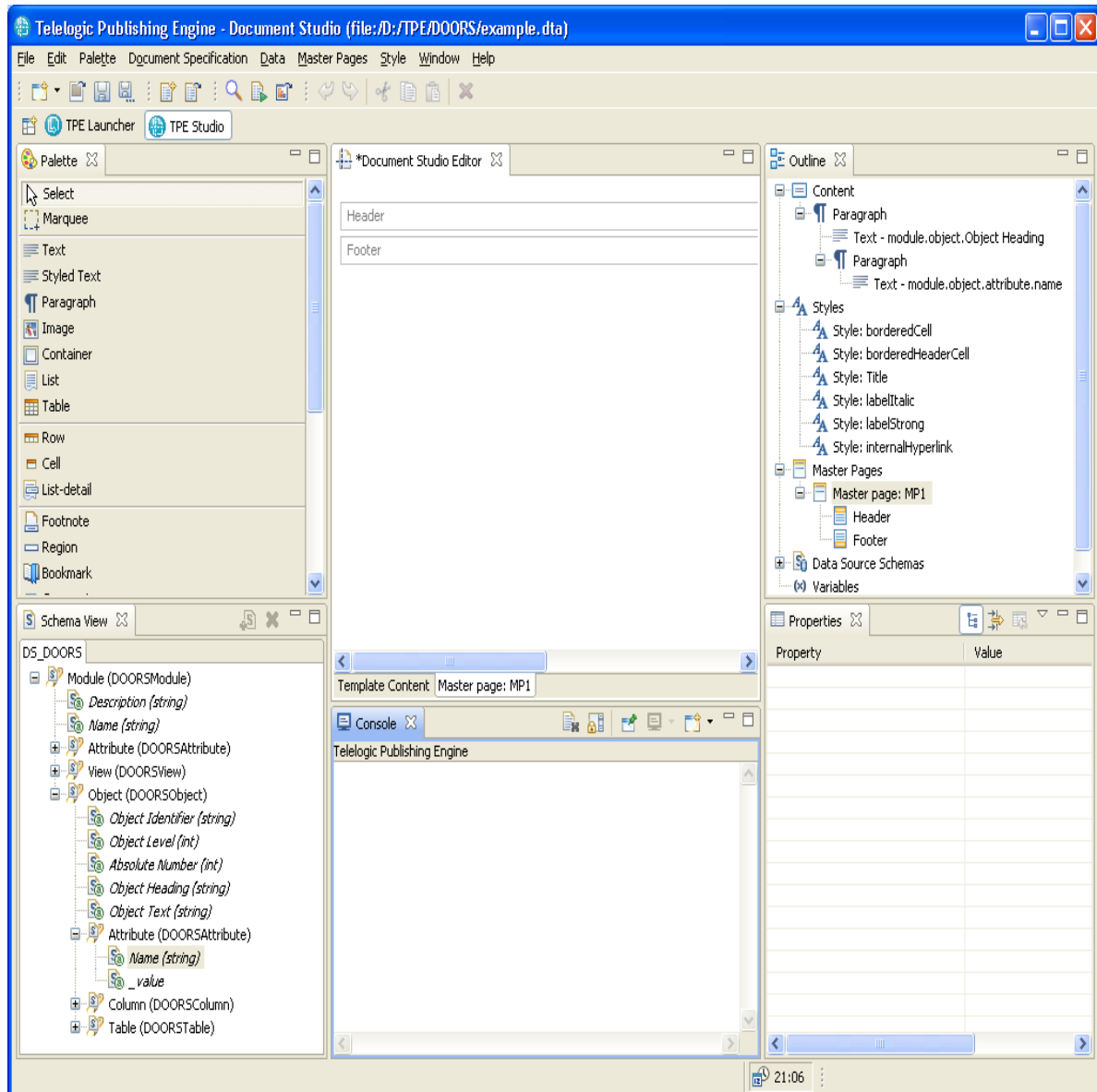


Figure 73 Master page

Using master pages

A master page is used in same way as a style: you assign the master page to an element. When the document is generated, TPE will evaluate the “master page” property of the element.

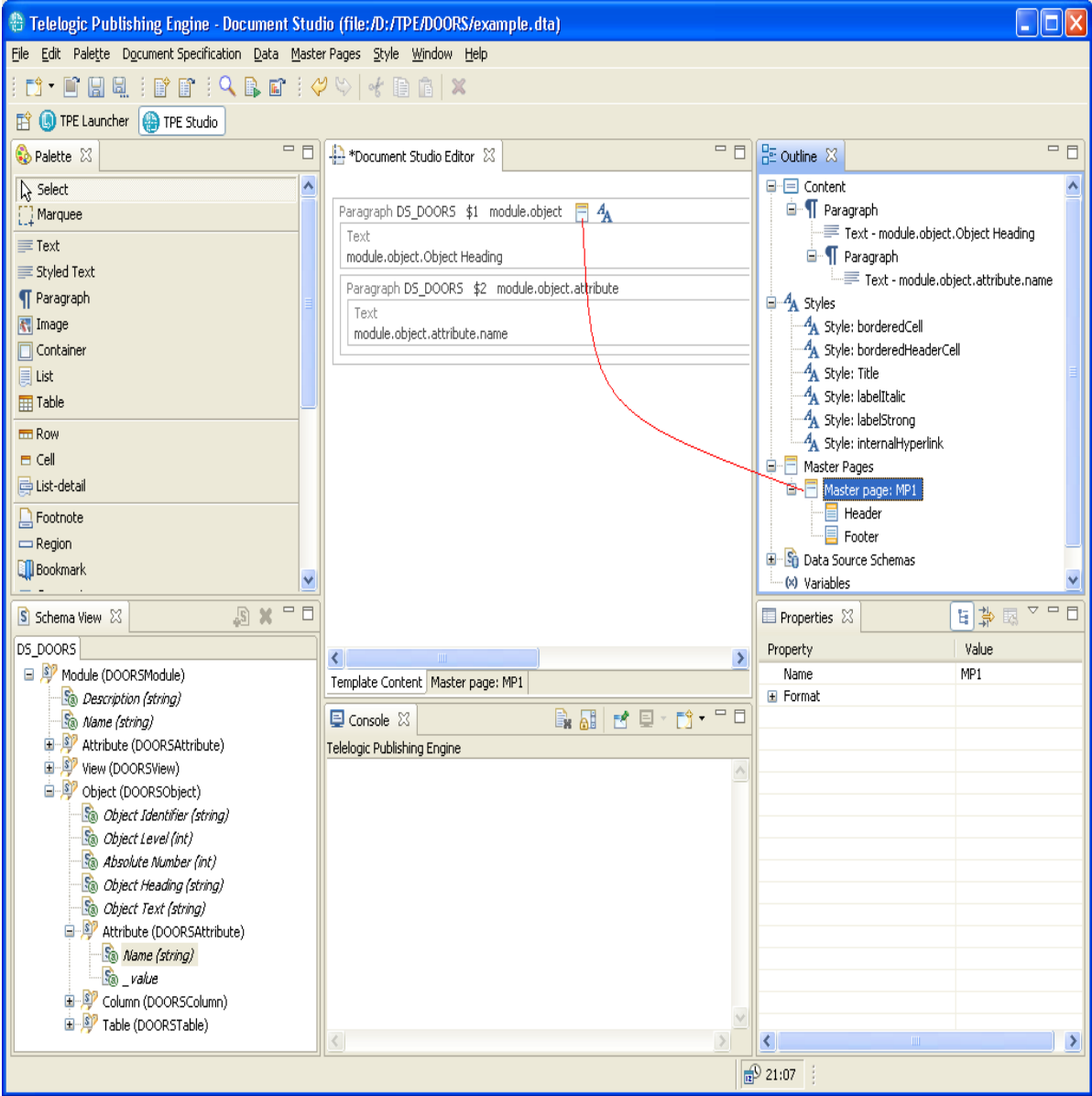
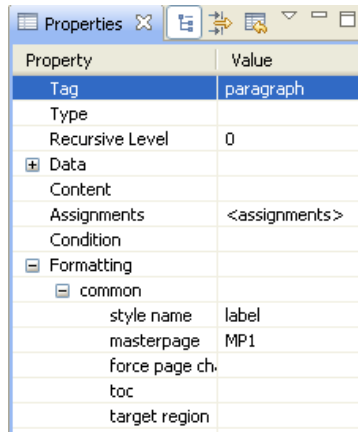


Figure 74 Assigning a master page using drag&drop



Property	Value
Tag	paragraph
Type	
Recursive Level	0
Data	
Content	
Assignments	<assignments>
Condition	
Formatting	
common	
style name	label
masterpage	MP1
force page ch.	
toc	
target region	

Figure 75 Assigning a master page from the properties page

NOTE When the output is generated, TPE will apply a master page only if different from the last used master page. This is to prevent having a master page for each element in a query. If you want this behavior you need to set the “force page change” property to true for the element having assigned the master page.

Data attributes in master pages

While you cannot use data attributes directly in master pages, it is possible to have data rendered in there. This is accomplished by:

- Creating a variable in the template
- Using the variable in the master page content
- Assigning a data value to the variable for each element that changes a page

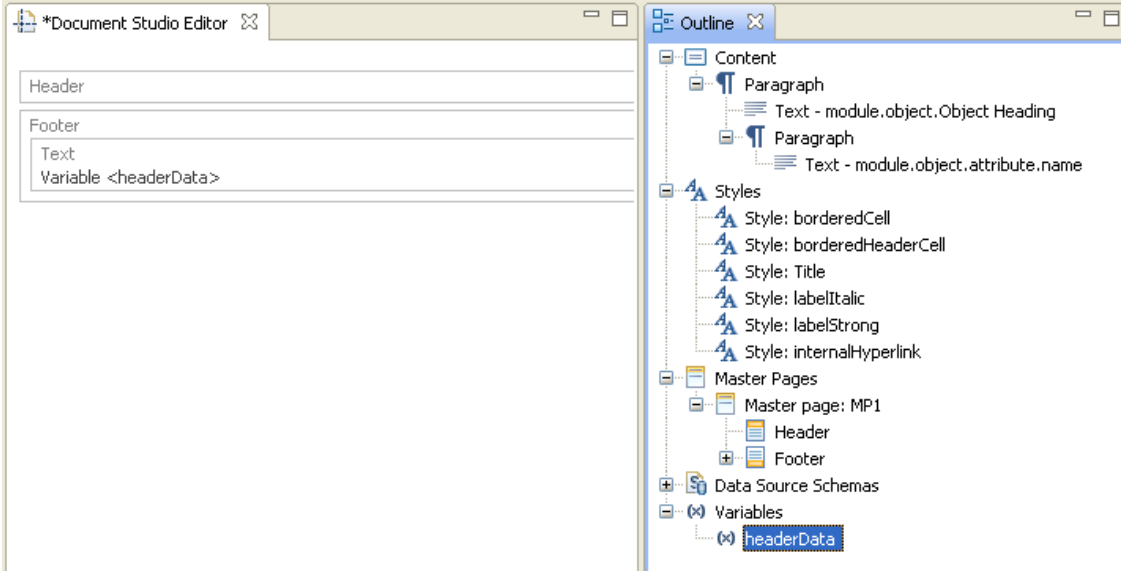


Figure 76 Variable used in the master page content

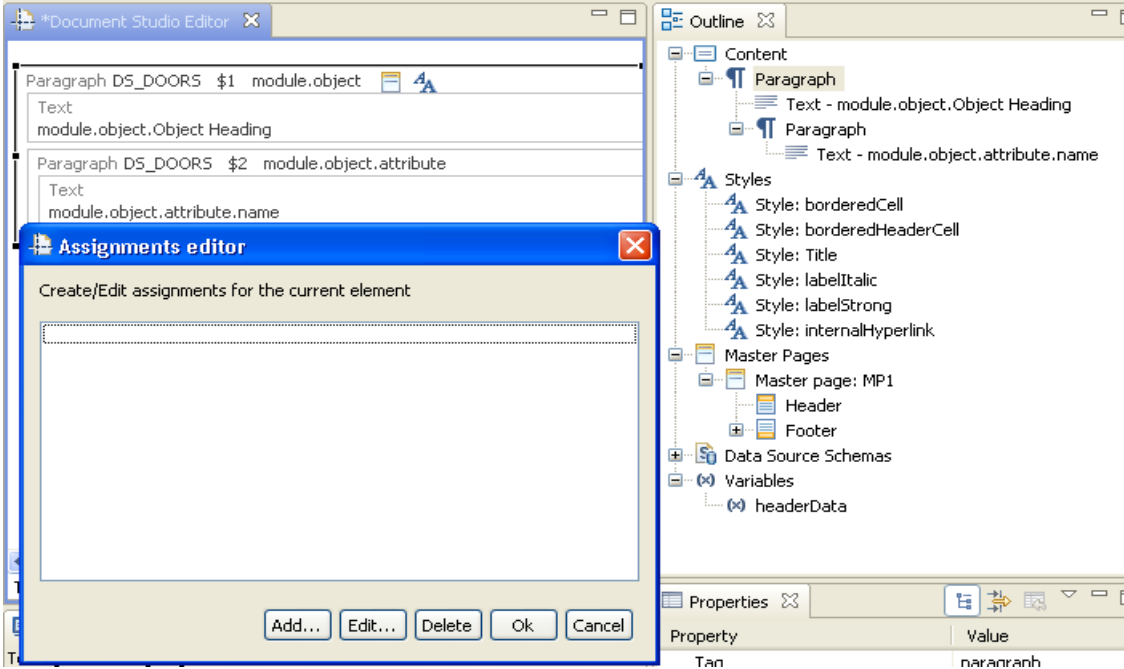


Figure 77 Creating an assignment for the element changing the master page

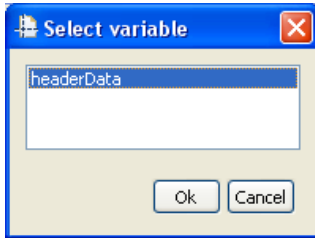


Figure 78 Select the variable to make the assignment for

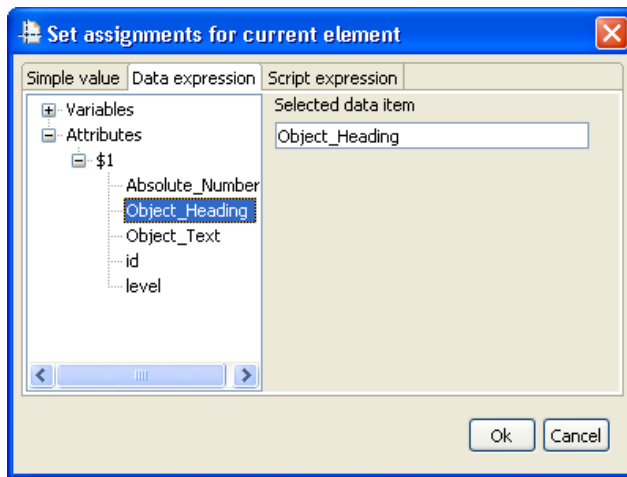


Figure 79 Define the assignment's value

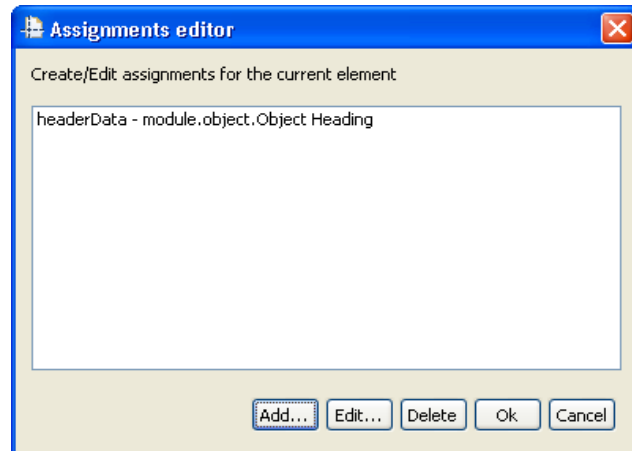


Figure 80 Newly created variable assignment

With the assignment in place the header of the master page will be rendered using the data content.

NOTE The variable assignment is calculated only when the master page is changed. If a header/footer contains a variable, the header and footer will display the value calculated when a page change occurred. In the above example, if the “force page change” is set to false then the header will display the text for the first object heading, regardless of how many objects with heading are in the module. This is caused by TPE attempting to minimize the number of page changes.

Advanced template design

Bookmarks

The TPE bookmark concept is identical to the bookmark concept in Microsoft Word documents or HTML documents.

NOTE_TPE will generate a unique name for each bookmark defined in the template. The unique name is based on the name defined for the bookmark and a random string generated by TPE. TPE will ensure internal hyperlinks use the unique name.

NOTE_You can use data/script expressions to define the name of a bookmark.

Hyperlinks

The hyperlinks defined in TPE Document Studio are of two types:

- *internal* – to a bookmark defined in the document template
- *external* – to an external URL (<http://www.telelogic.com>)

The type of a hyperlink is defined by the *internal* property.

The link location is defined in the *content* of the hyperlink.

The text displayed by the hyperlink is defined by the *display* property and has the default value of “link”.

NOTE You can use data/script expressions to define the content and display of a hyperlink.

Internal hyperlinks

When using internal hyperlinks it is mandatory that the value (content) of the link is identical to the name (content) of the bookmark. This holds true even if the two values are data/calculated values. If the two values are not identical the internal hyperlink will not take you to the expected bookmark.

Variables

TPE template variables are available to be used as placeholders for data calculated at runtime (variable assignments) or provided in the document specification. Unlike data attributes you can also use variables in master pages.

A variable can be added from the outline context menu or from TPE's main menu.

NOTE_The variable name needs to respect Java Script naming conventions. With the current TPE version you have to enforce this manually but future TPE builds will handle this.

TPE Predefined Template

TPE Document Studio comes with a predefined template named “template.dta” locate in the “utils” folder. You can see this template the equivalent for Microsoft Word’s normal.dot. TPE Document Studio uses “template.dta” each time it creates a new document.

NOTE_You can customize template.dta to match your organization’s needs. You can define styles, add header content (company logo, legal info etc), define master pages etc.

TPE launcher integration

The TPE launcher comes integrated in TPE Document Studio. There are two run modes for the launcher in this case:

- Synchronized with studio
- Not synchronized with studio

The run mode can be specified in the preference page of TPE Document Studio.

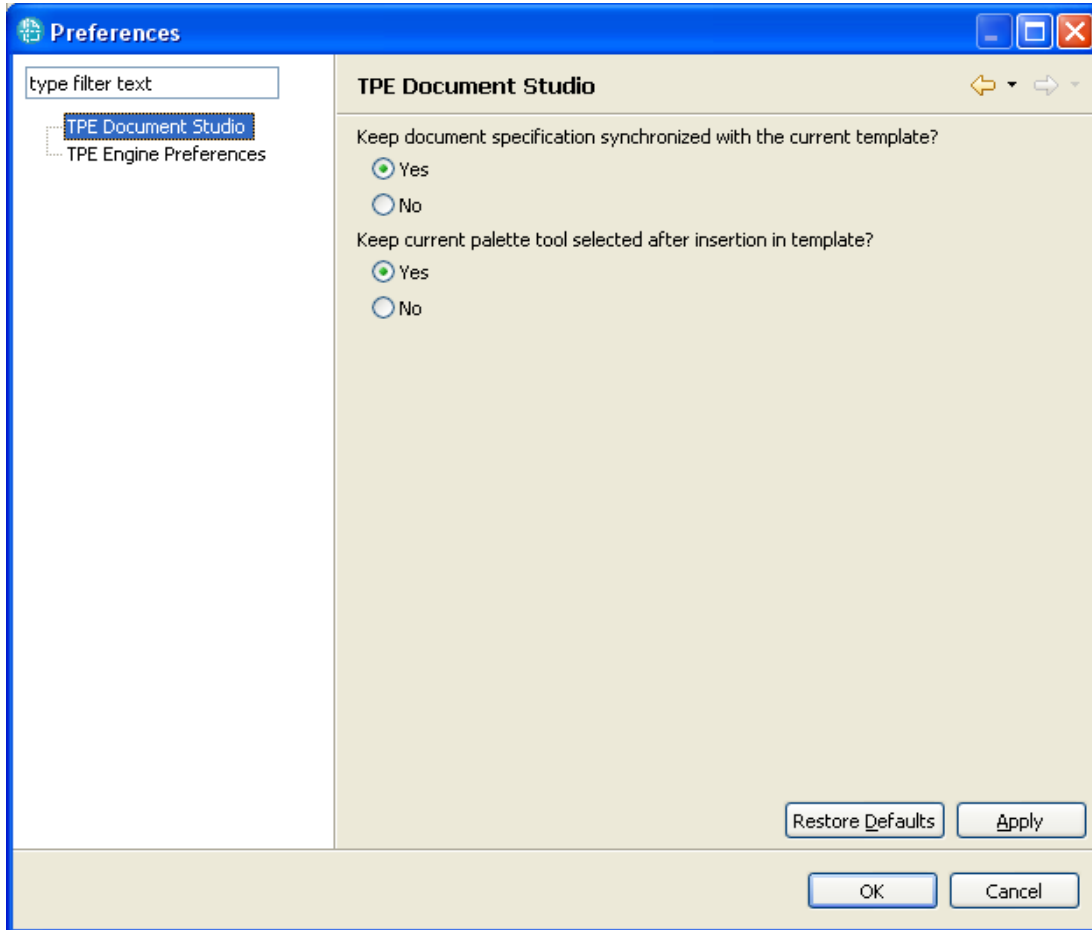


Figure 81 TPE preference page

Synchronized mode

This run mode is meant to support testing of the current template in TPE Studio. While running in this mode it is not recommended to manually load document specifications as TPE Studio actions will modify the document template.

TPE Studio Action	Effect in TPE launcher
New template	The existing document specification is closed and a new one is created

Open template	The existing document specification is closed and a new one is created. The newly loaded template is assigned to the document specification
Save template	If this was a newly created template, the template is assigned to the document specification. The document specification is synchronized with the template.
Save as	If this was a newly created template, the template is assigned to the document specification. Otherwise the current template in the document specification is replaced by the new one (with the new path).

Generate & preview

In synchronized run mode TPE will detect if the current template is unsaved and will prompt you to save it before generating the output or the preview. Cancelling the save operation will also cancel the document generation.

Unsynchronized mode

In this run mode, the TPE Launcher operates as it would do as a standalone application.

Schema Discovery

TPE provides tools to assist the user to build data source schemas for some data sources. In the current build the only data source type supported is DOORS.

DOORS Schema Discovery

The wizard guides you in building a schema for a specific DOORS module. This will greatly simplify authoring document templates for modules with the same structure (same or similar attribute list)

Start the wizard

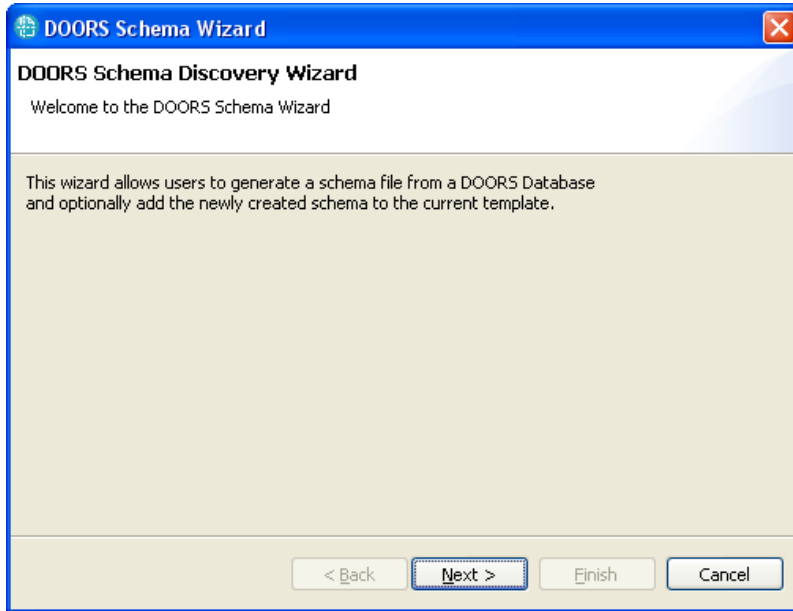
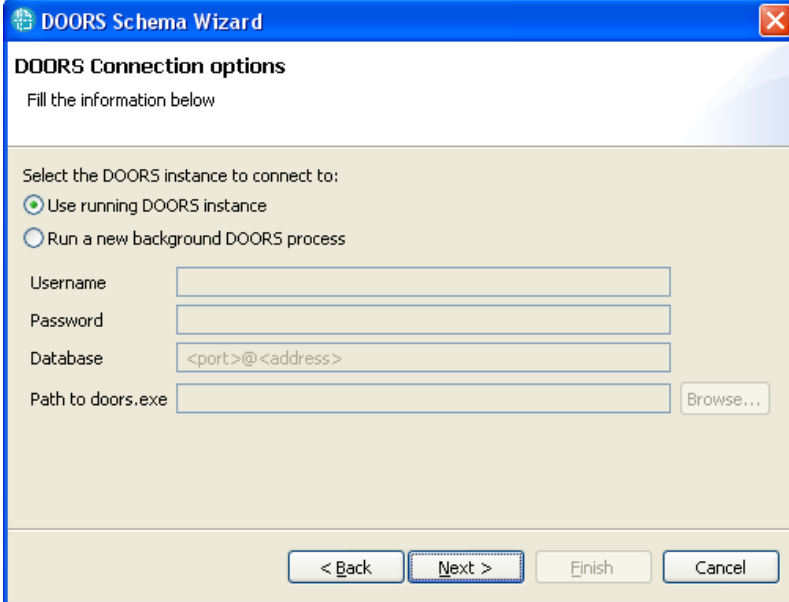


Figure 82

Configure DOORS Connection

Specify how you want TPE to connect to DOORS in order to discover the schema.



The screenshot shows a Windows-style dialog box titled "DOORS Schema Wizard". The main heading is "DOORS Connection options" with the instruction "Fill the information below". Below this, there are two radio button options: "Use running DOORS instance" (which is selected) and "Run a new background DOORS process". Underneath are four input fields: "Username", "Password", "Database" (with a placeholder "<port>@<address>"), and "Path to doors.exe" (with a "Browse..." button to its right). At the bottom of the dialog are four buttons: "< Back", "Next >", "Finish", and "Cancel".

Figure 83

Select for which module you want to discover its schema

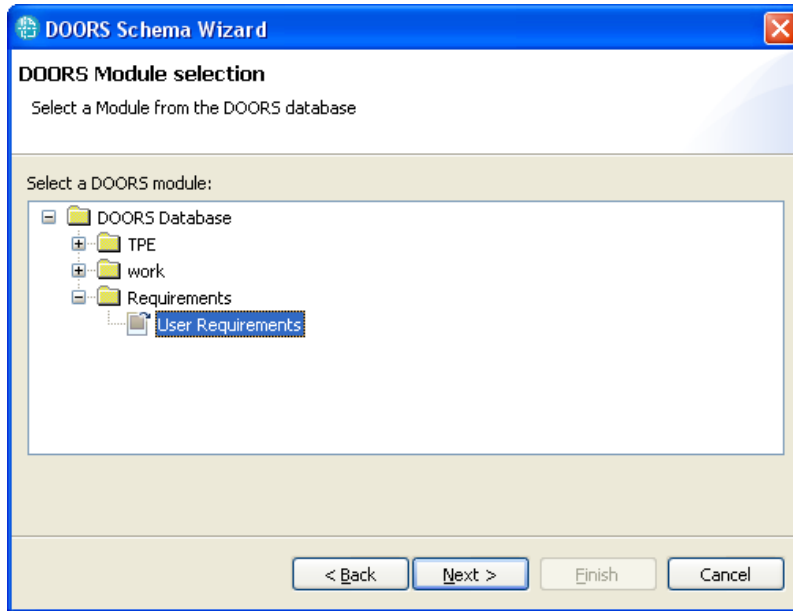


Figure 84

Select the baseline from where to read the attributes

Once you've selected the module you need to specify the module's baseline you want to use. The selected baseline will determine which attribute set to use.

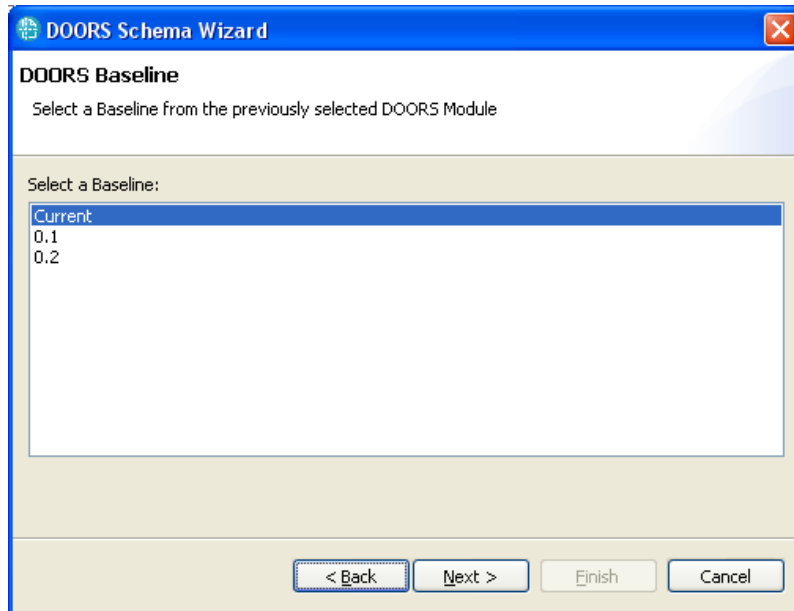


Figure 85 Selecting the baseline

Select the attributes to elevate

This screen allows you to select the attributes to use. The attribute set is taken from the baseline selected in the previous screen.

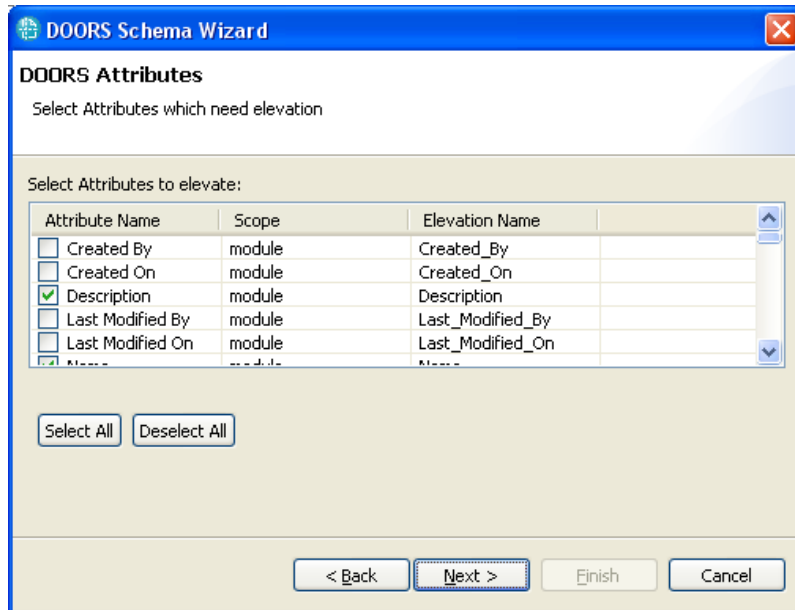


Figure 86 Selecting attributes

An elevated attribute allows direct access to that attribute's value from a *module.object* context. Non elevated attribute values are available only from a *module.object.attribute* context.

NOTE You can select the attribute set from any baseline of the module but you cannot have attributes selected from 2 different baselines.

NOTE The elevated name is the name used in script expressions. Hence it must be a valid JavaScript identifier. TPE will generate a valid name out of the DOORS attribute name and will prevent you from changing it to an invalid name.

NOTE The selected baseline is used for the sole purpose of defining the attribute shown to the user. This information (the baseline used to elevate the attributes) is not used at document generation time. If one attribute does not exist in the baseline used for document generation nothing will be rendered for it in the output.

Select the columns you want to elevate

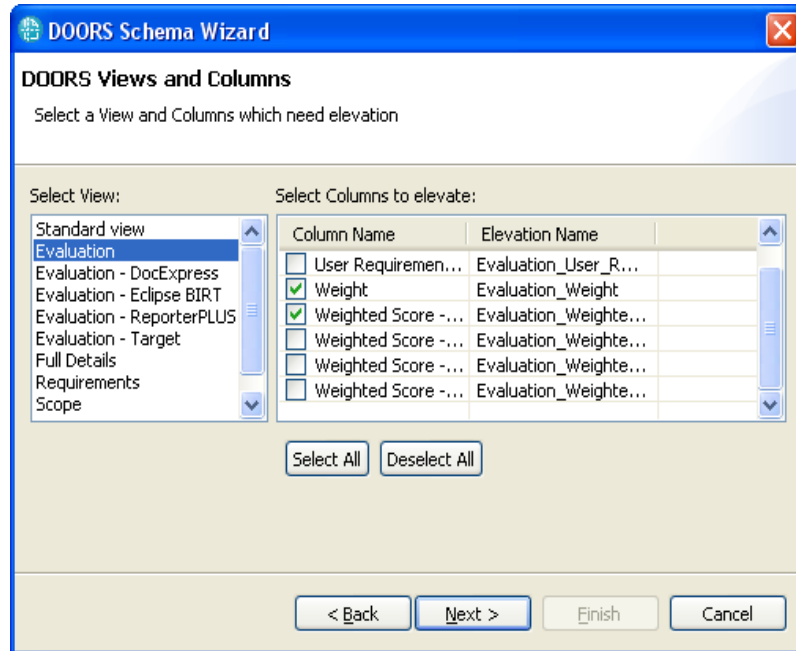


Figure 87

An elevated attribute allows direct access to that attribute's value from a *module.object* context. Non elevated attribute values are available only from a *module.object.column* context.

NOTE You can elevate columns from any number of views.

NOTE With the current TPE version you can elevate only columns that do not contain < or >.

NOTE The elevated name is the name used in script expressions. Hence it must be a valid JavaScript identifier. TPE will generate a valid name out of the DOORS column name and will prevent you from changing it to an invalid name.

Save schema and add it to the current template

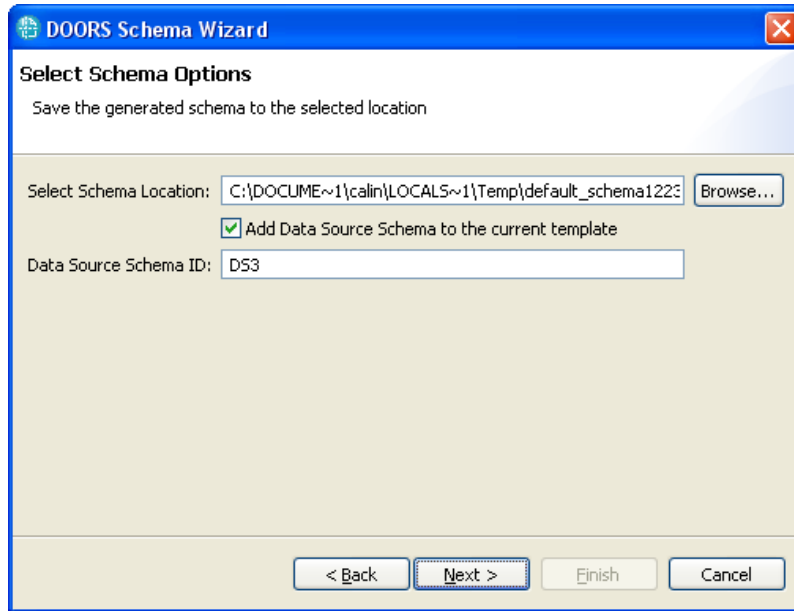


Figure 88

NOTE You should save discovered schemas so you can reuse them in other templates without having to run the schema discovery wizard again.

Review changes and finalize the wizard

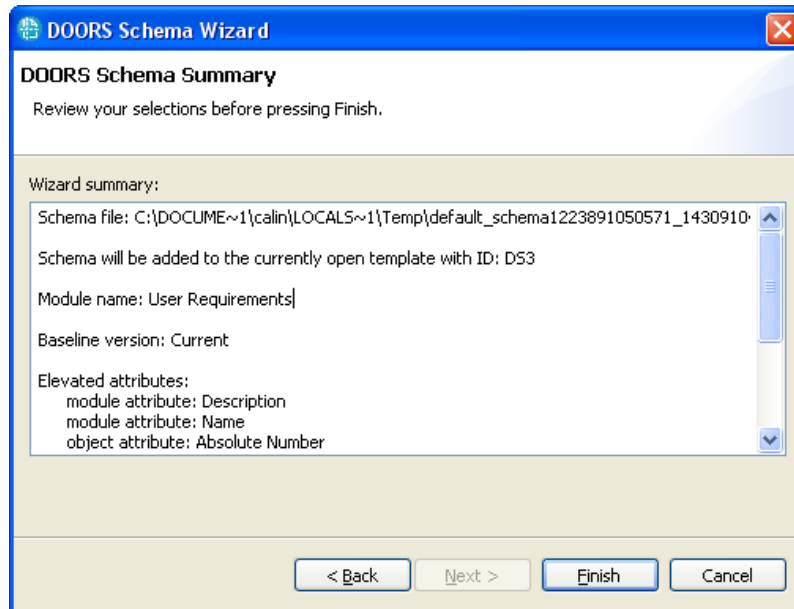


Figure 89

Tau Schema Discovery

TPE 1.0 does not offer an integrated schema discovery wizard for Tau. To generate a Tau schema for your model you need to do the following actions:

Start Tau and open your model

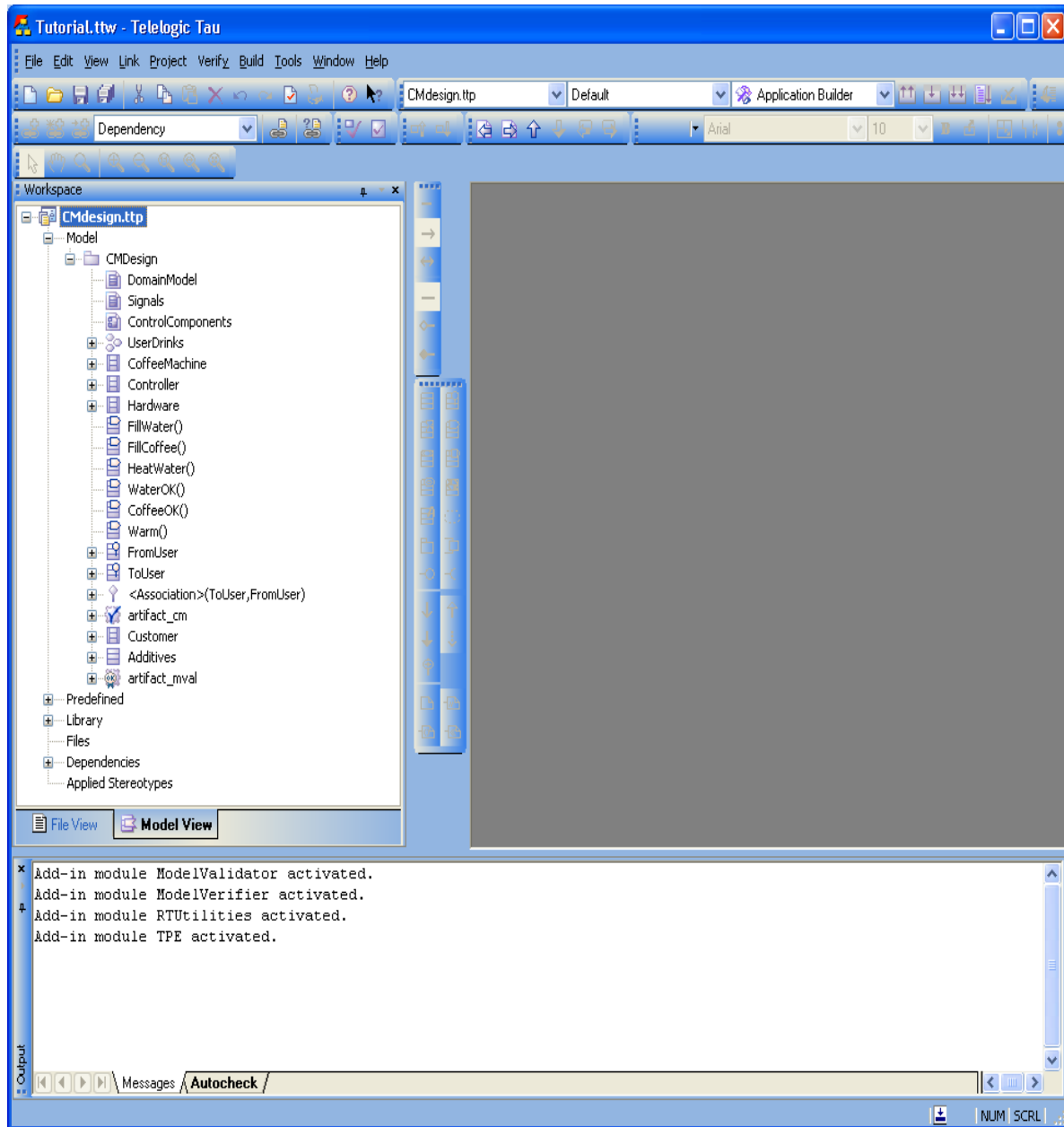
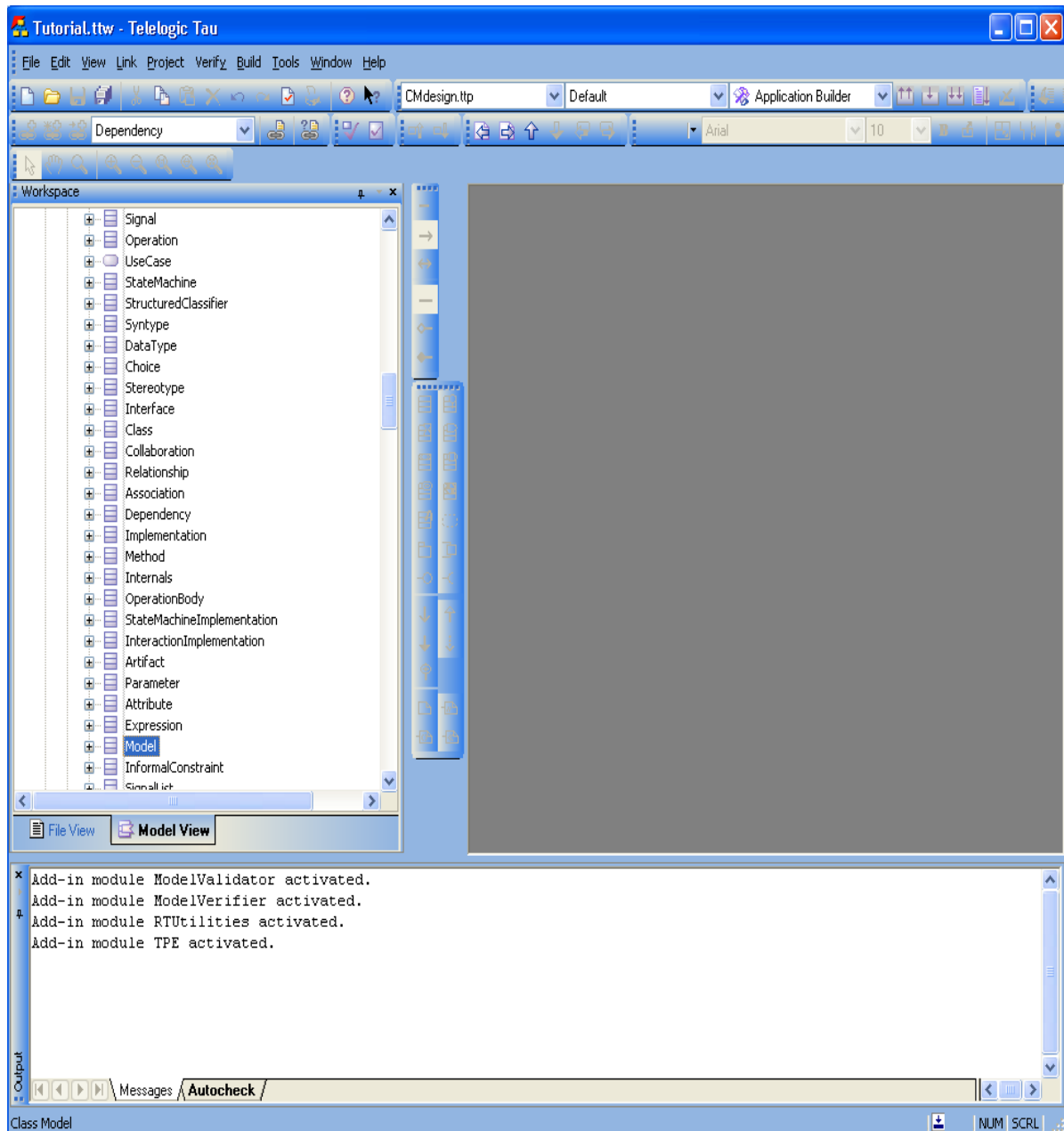


Figure 90

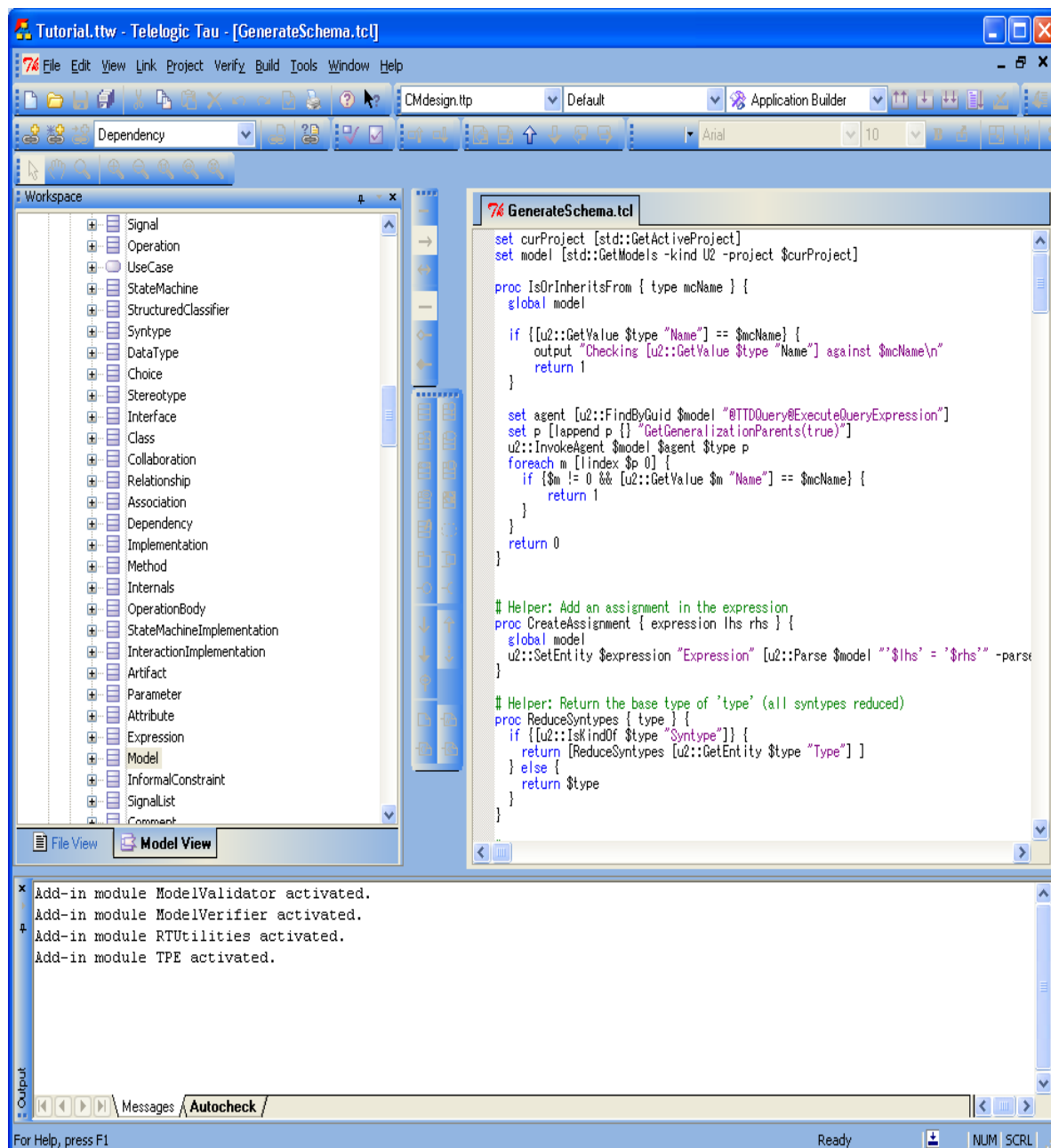
Select the metamodel's root

In the explorer select the model element that serves as your model root.

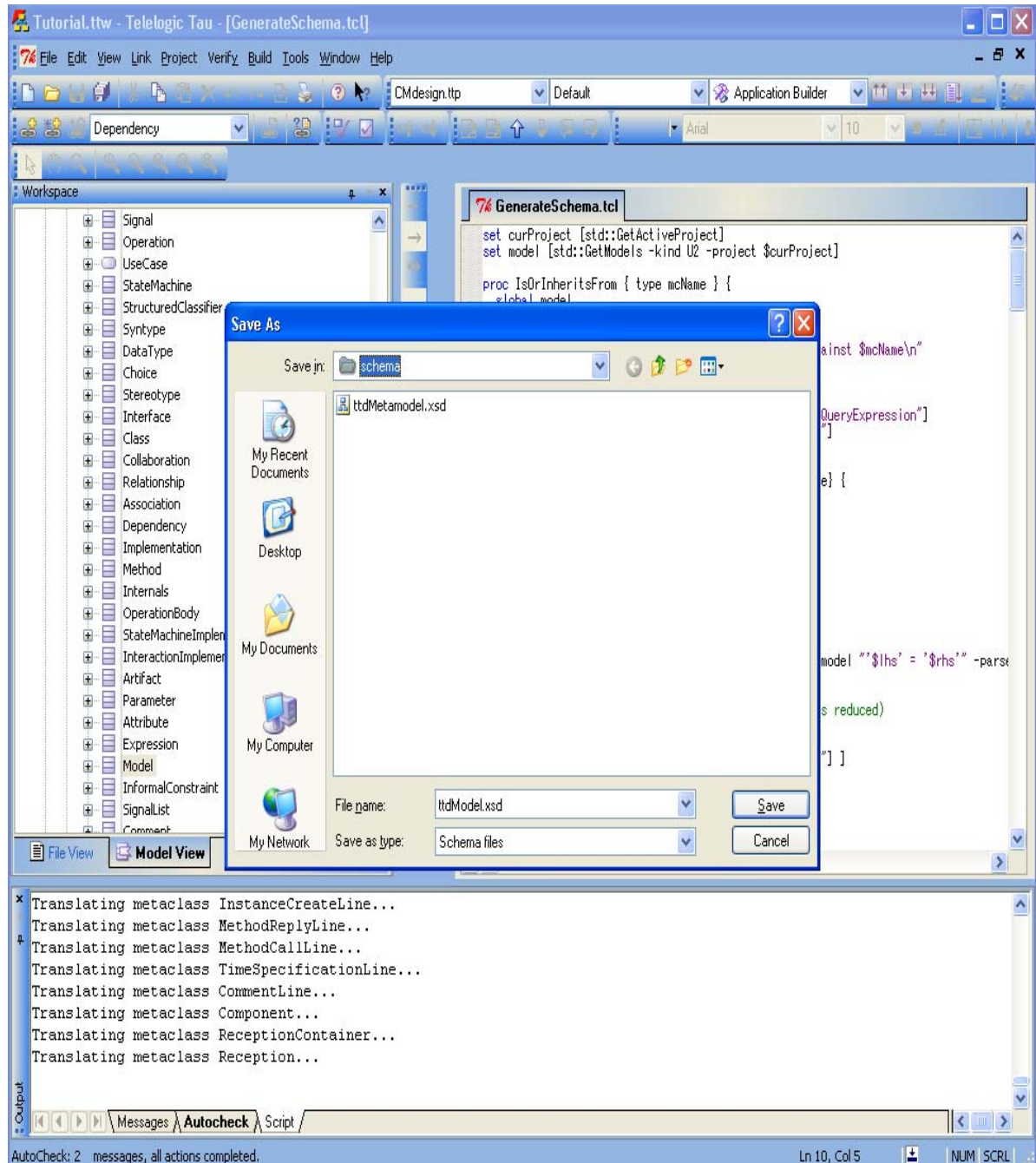


Load the GenerateSchema.tcl

Load the TCL script file provided in the TPE installation.



Run the script



Once the script execution is complete you will be prompted with the location for saving the schema

DOORS Addin

TPE integrates with DOORS to allow DOORS user to generate documents from within their familiar environment. TPE allows a DOORS user to generate documents based on predefined document templates or specifications.

The integration is available both within a module as well as from the database explorer

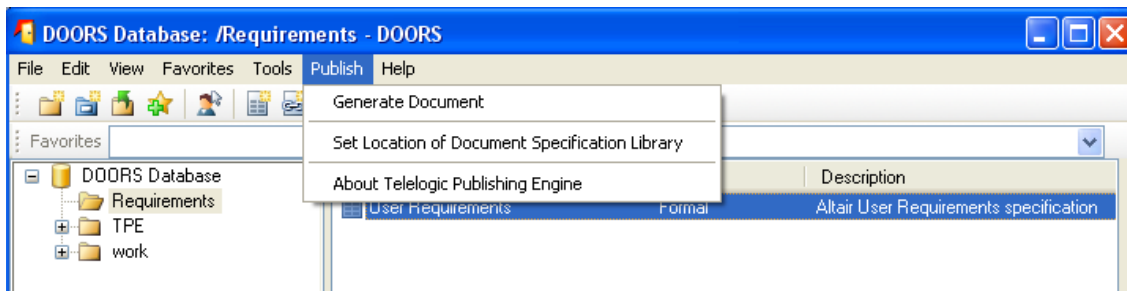


Figure 91 TPE Addin in database menu



Figure 92 TPE Addin in module menu

Before using the integration a DOORS Administrator needs to set the location for the “Document Specification Library”. The Document Specification Library is a folder on the local or network file system used to host document templates and specifications to be shared by all users. The library can contain subfolders to structure the templates as needed, and standard file system access rights can be used to control which templates and document specifications are visible to which users.

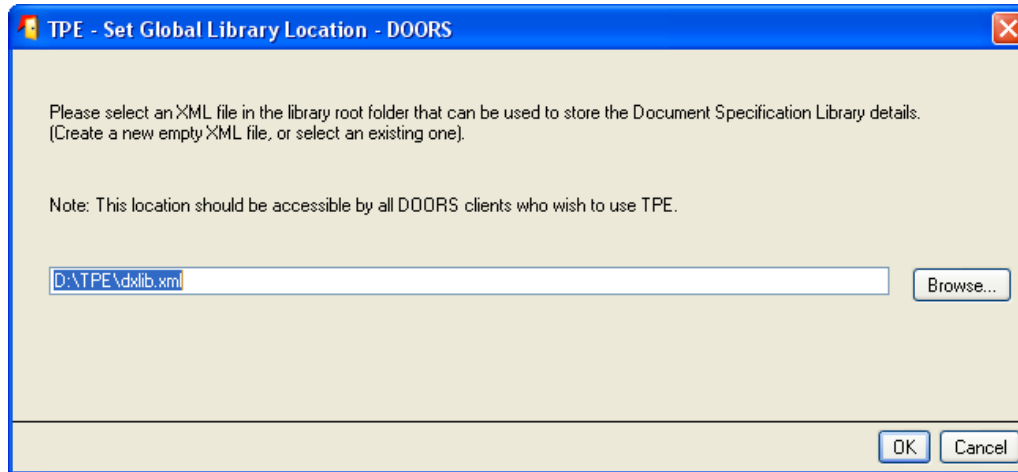
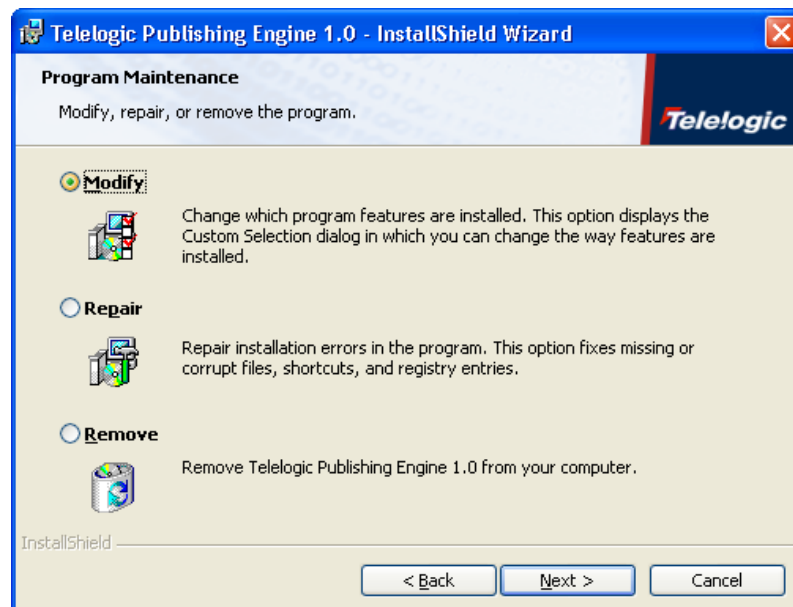


Figure 93 Setting the Document Specification Library

Installation

The TPE addin for DOORS is automatically installed by TPE if a valid DOORS installation is found. If you install DOOR after you have installed TPE you can install addin by running the TPE installer again with the option to modify the existing installation.



Usage

To publish a document from within DOORS you need to follow a set of simple steps.

Select the document template/specification.

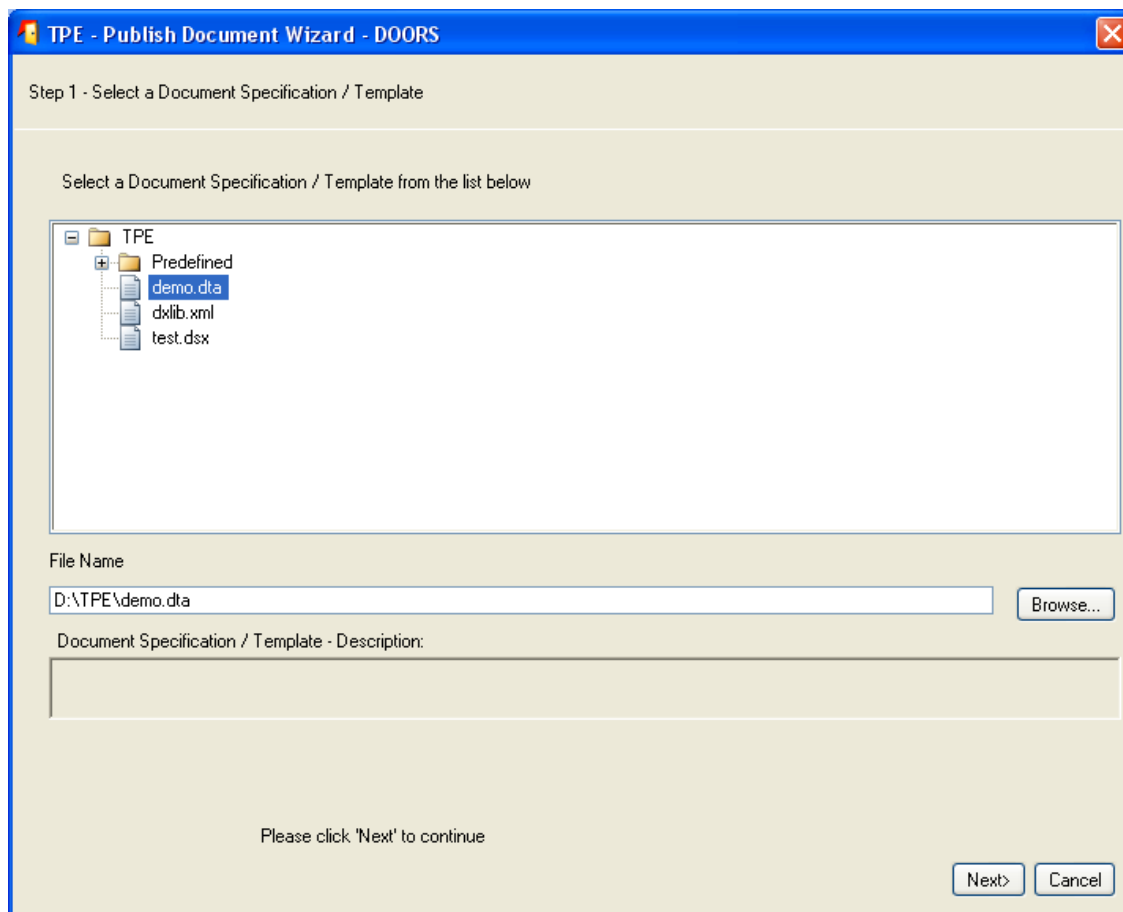


Figure 94

If the document template/specification is not in the library use the browse button to select it.

NOTE The description field is not filled in the current TPE version.

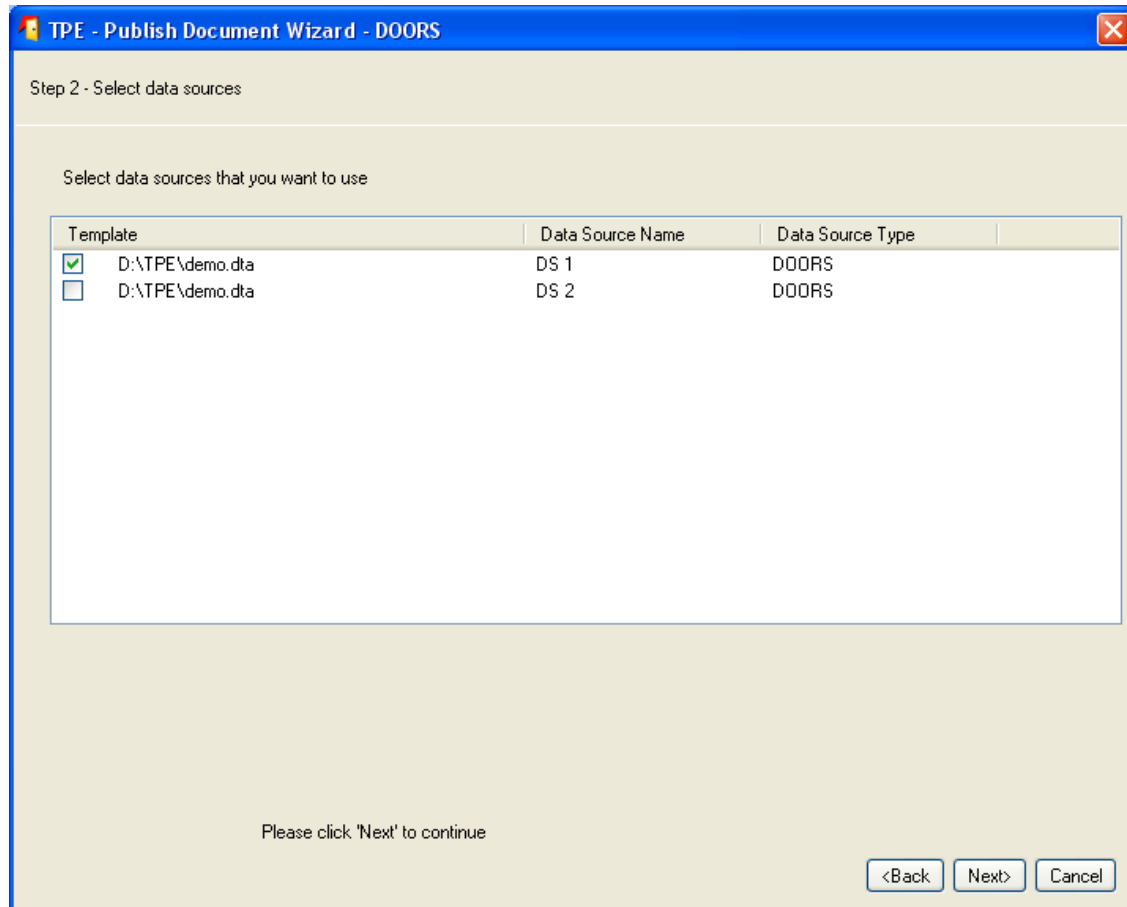
Select the data sources to be used

Figure 95

NOTE A data source that is not selected in this screen will be ignored during the document generation process.

NOTE If no DOORS data source is selected, the wizard will not continue.

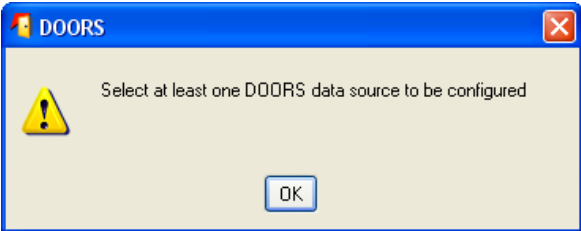


Figure 96 Warning message

Configure the data sources

Applicable when running from the Publish menu in the DOORS database explorer

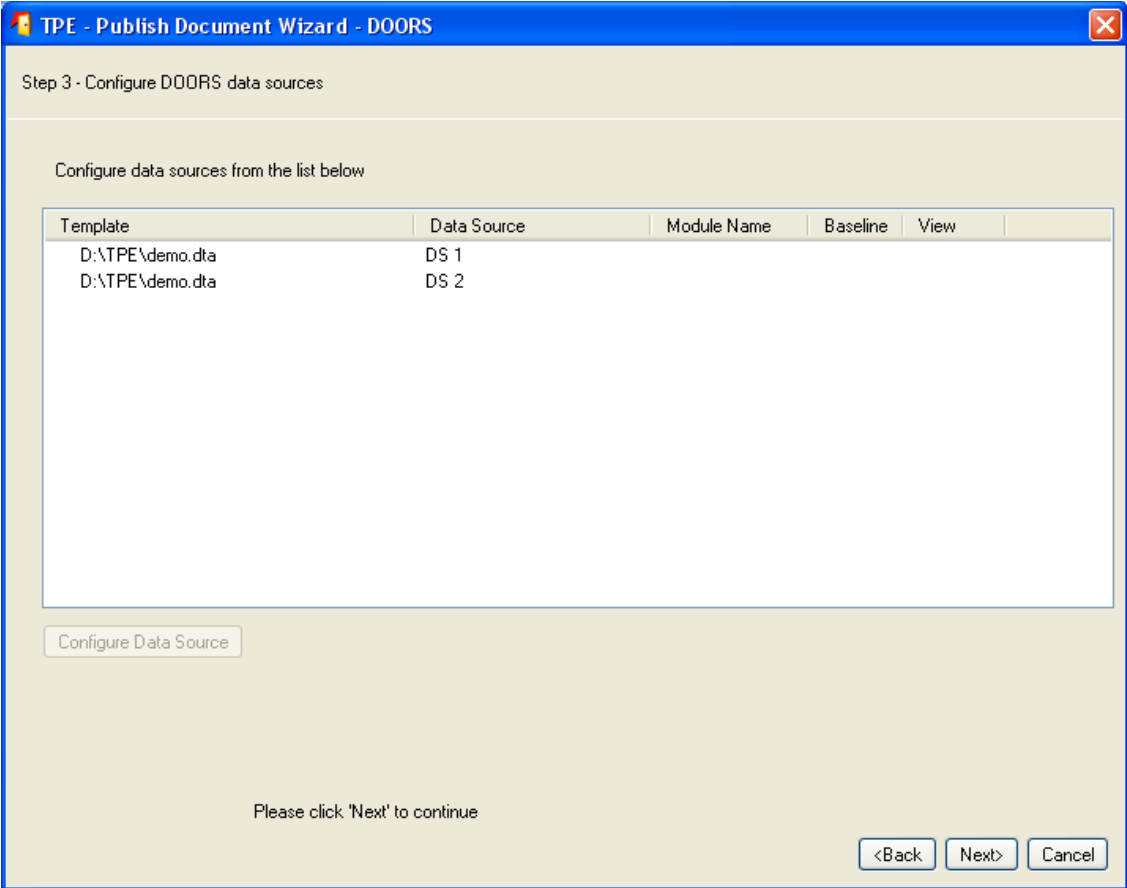


Figure 97 Configure data sources screen

NOTE If you are publishing a document template the DOORS data sources are left empty.

NOTE If you are publishing a document specification, the DOORS data sources are initialized with the configuration set in the document specification.

NOTE You can only configure DOORS data sources. If you are using a document specification, any existing configuration for non-DOORS data sources will be preserved.

Configure the data sources

Applicable when running from the Publish menu in a DOORS Module.

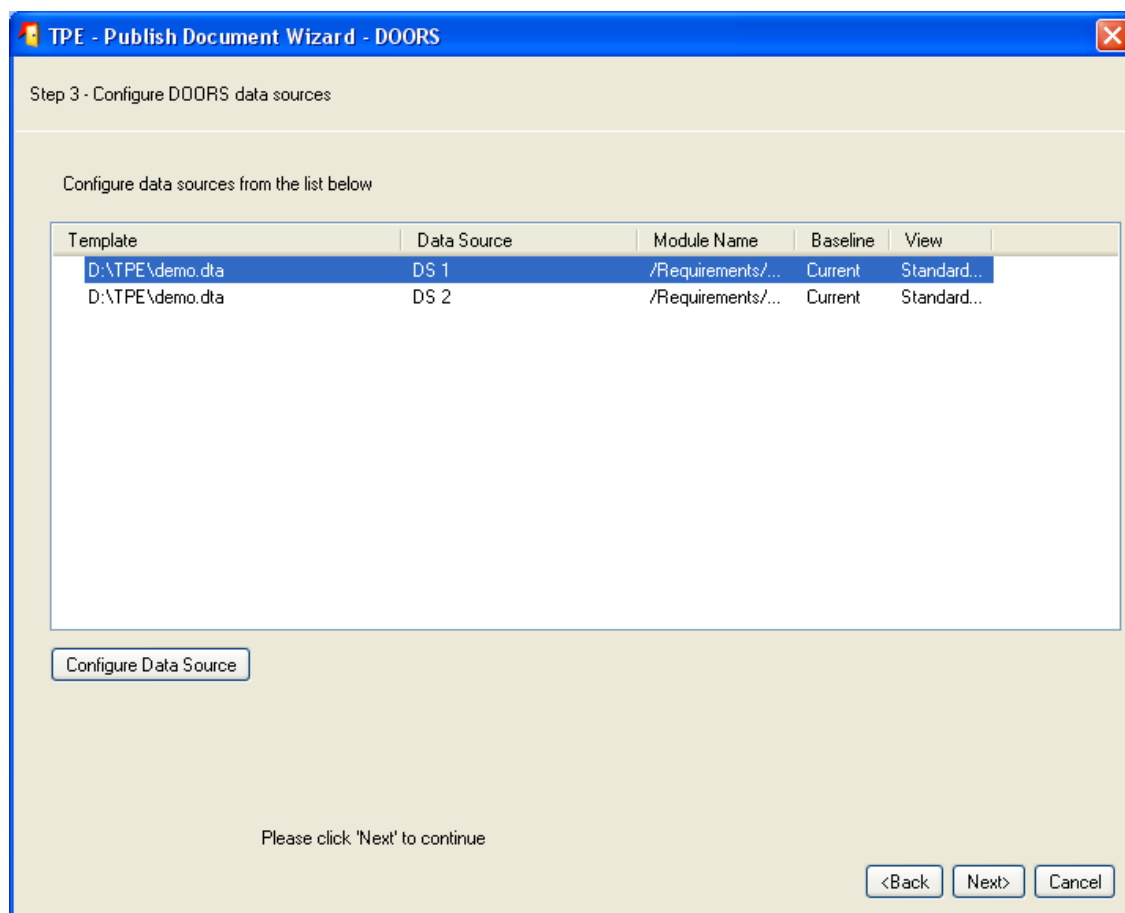


Figure 98

NOTE The DOORS Data sources are initialized with the current module, view and baseline. If you are publishing a Document Specification any existing DOORS Data Source configuration is automatically overridden (though you can still manually change each)

NOTE You can only configure DOORS data sources. If you are using a document specification, any existent configuration for non-DOORS data sources will be preserved.

NOTE If not all selected DOORS sources are configured TPE will issue a warning but will allow you to continue.

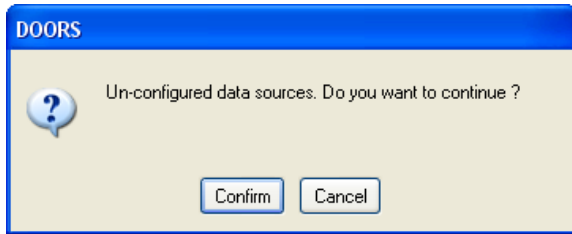


Figure 99

Configuring a data source

If you want to change module/baseline/view assigned to a data source you need to select the data source in the list and click the “Configure Data Source” button. The structure of the database will be displayed in a new window allowing you to select the desired module/baseline/view.

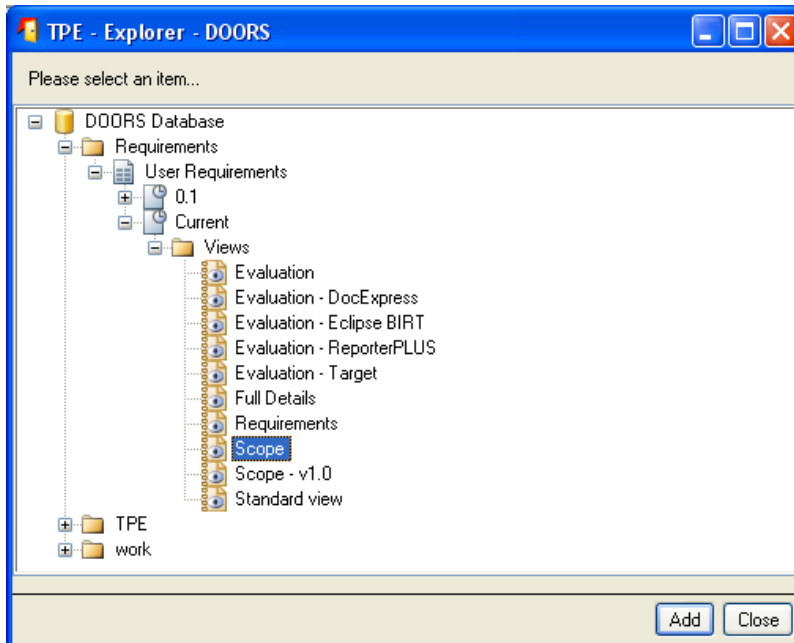


Figure 100

Configure the output

Once all data sources are configured you can define the document outputs that you want to use.

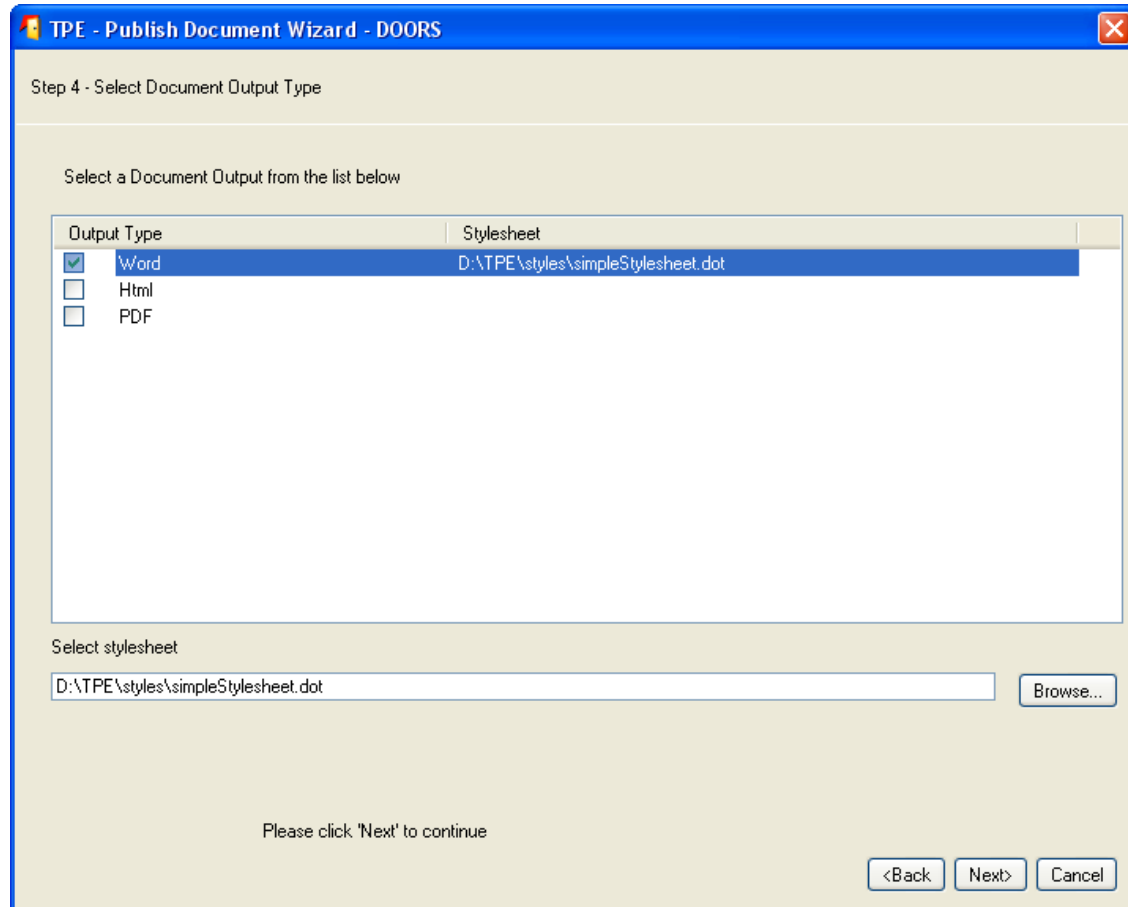


Figure 101

For each document output you can specify the stylesheet to be used (where applicable). To assign a stylesheet to an output format you need to do the following:

- select the output format in the list
- browse for the desired stylesheet

Document generation options

Once the data sources and output formats are configured you are prompted with options for running the document generation. You can:

- publish the document immediately

- create a document specification based on the options made in the wizard
- both

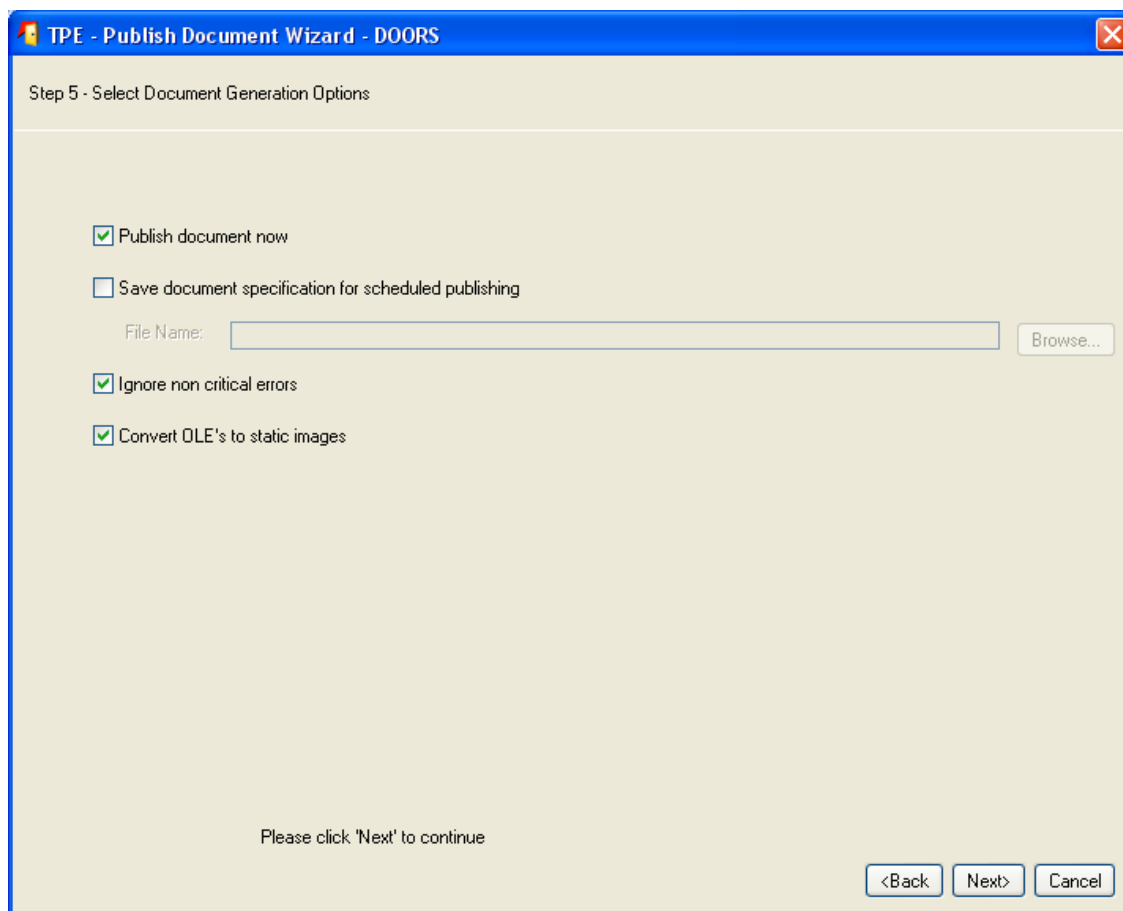


Figure 102

NOTE If “Convert OLE’s to static images” is checked, the Word output document, if selected, will contain images instead of OLEs.

NOTE If “Convert OLE’s to static images” is unchecked, you need to run the “insertOLEs” macro to get the OLEs embedded in the Word output.

Summary page

Before closing the TPE Publish wizard will display a summary page with all the choices made.

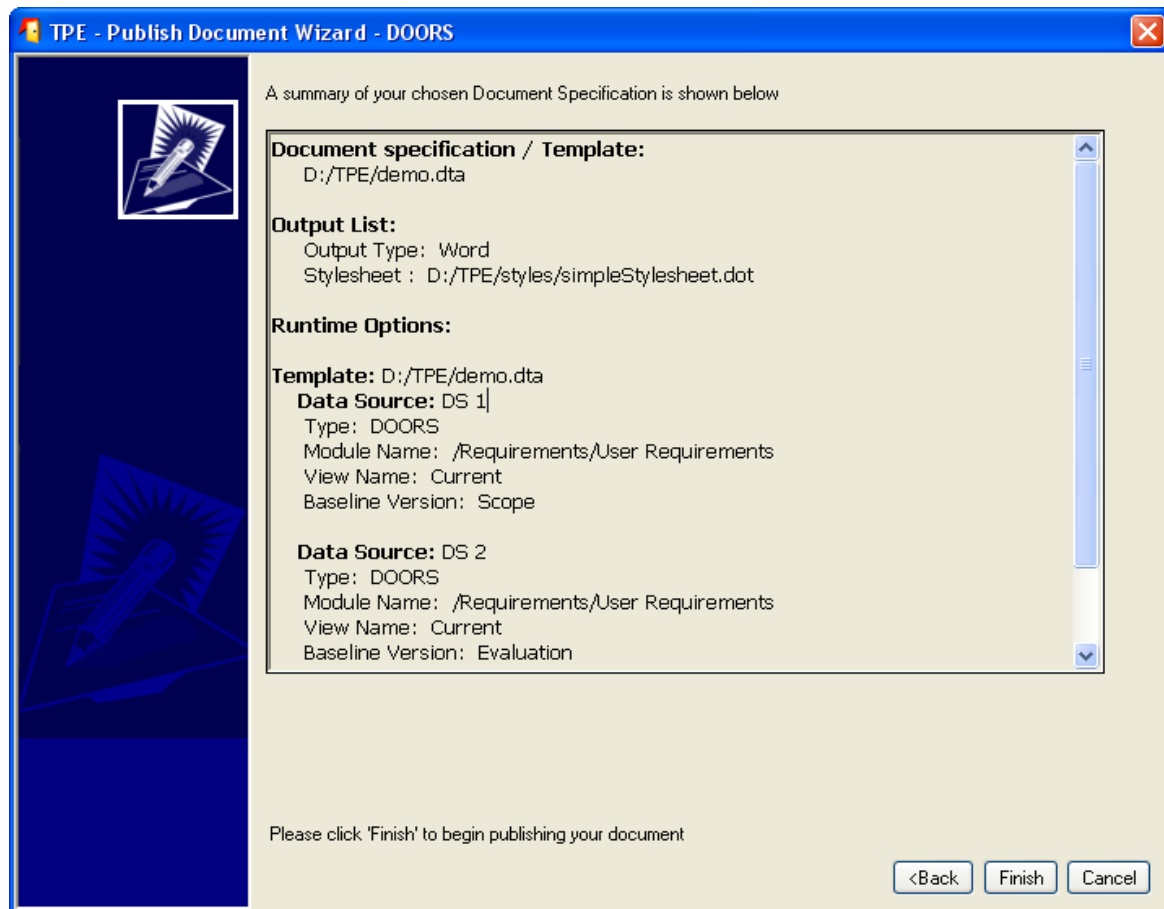


Figure 103

Press Finish to complete the wizard and start the document generation (if option selected).

NOTE The document generation is started as a separate process from DOORS but TPE will use the existing DOORS instance to extract data.

Tau Addin

Installation

The TPE addin for Tau is automatically installed by TPE if a valid Tau installation is found. If you install Tau after you have installed TPE you can install addin by running the TPE installer again with the option to modify the existing installation.

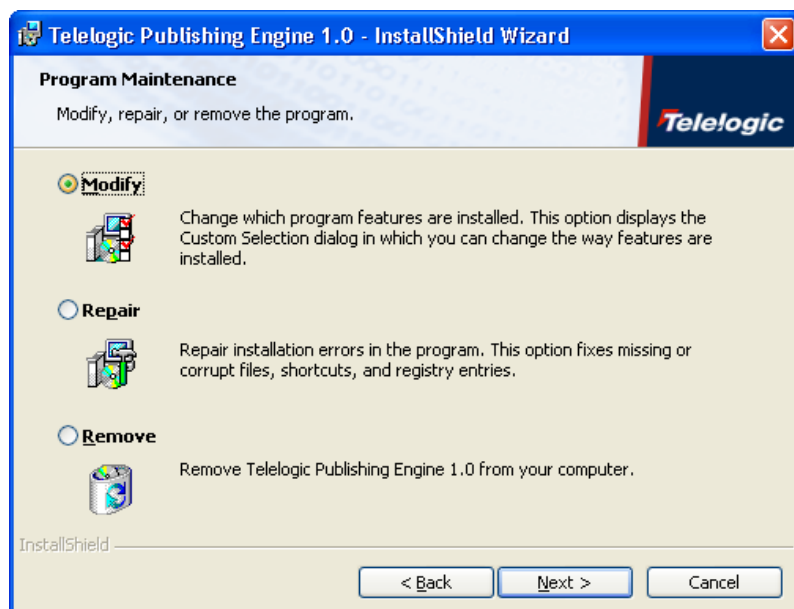


Figure 104

Once installed you need to activate the addin in Tau. Activating the addin can be done from Tau's addins page of the Customize function.

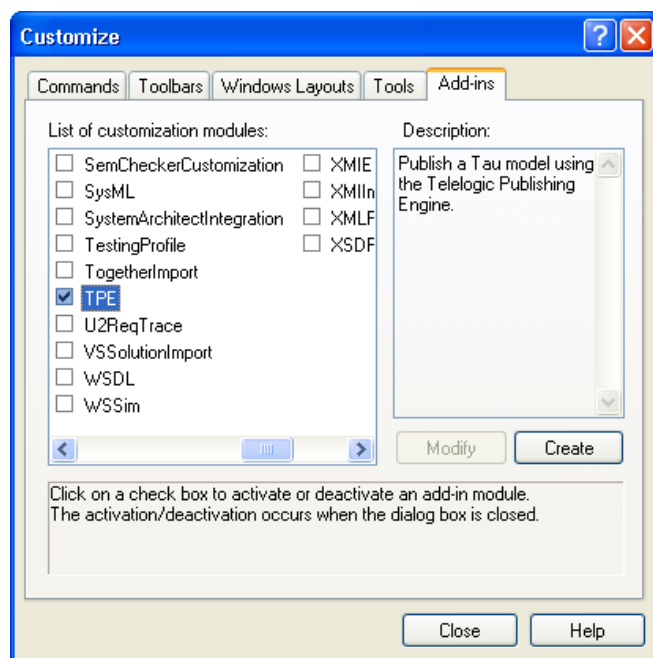


Figure 105

NOTE You need to have a Tau model loaded for the TPE addin to be listed.

Shared Document Library

The TPE Tau addin, much like the DOORS, makes use of the concept of a shared document template library. The content of this shared library (which is a folder on the file system, usually on a shared network drive) is displayed to the user in the Publish wizard.

NOTE As Tau does not have a server side component, the shared document library path is stored in the TPE_DOCUMENT_LIBRARY system variable. The variable is setup by the TPE installer to point to the Tau examples folder in the TPE installation. You can change this value to the desired location once TPE is installed.

Usage

Start the TPE addin from the Tools->Publish... menu.

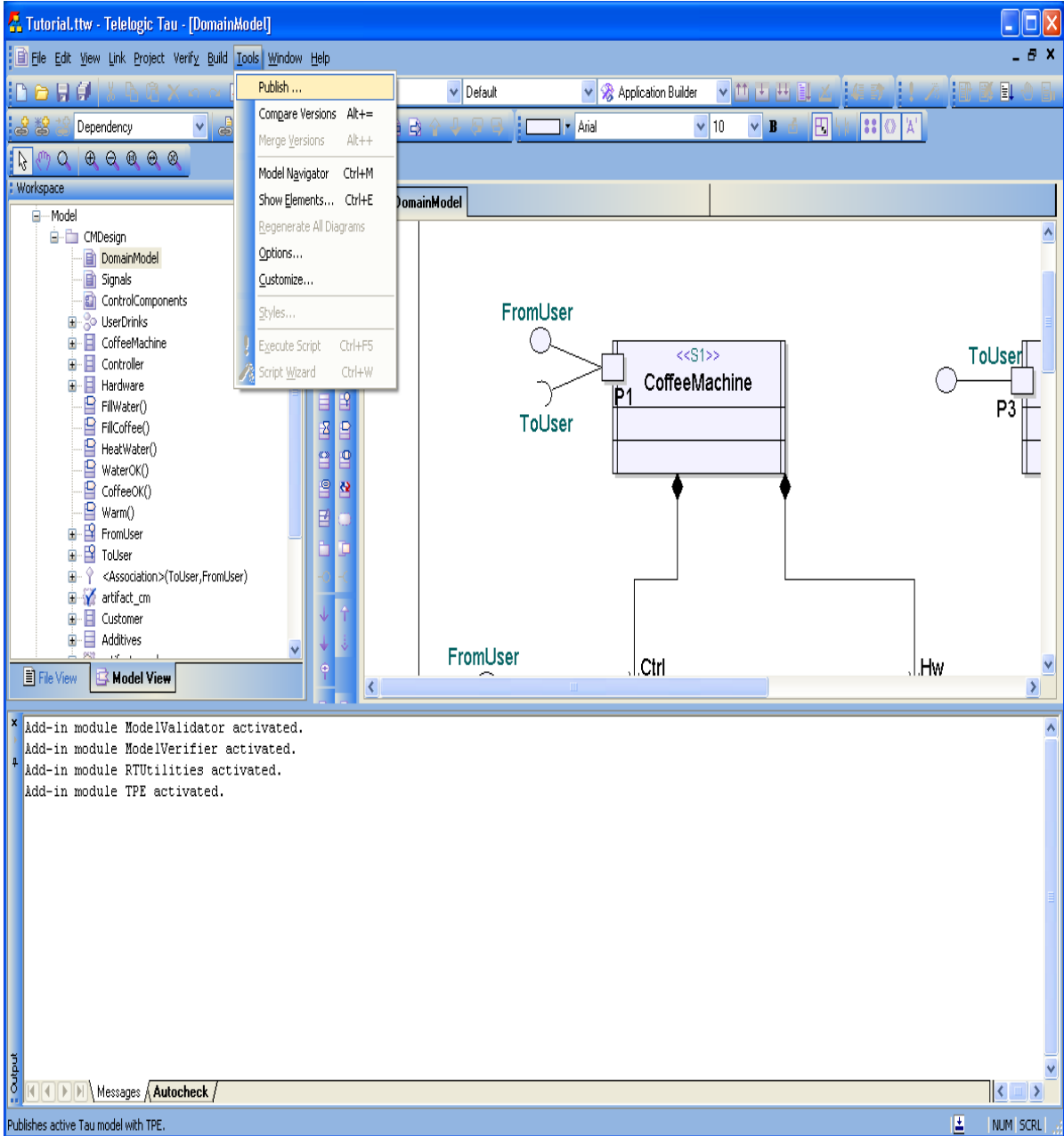


Figure 106

The TPE Publish Wizard is started.

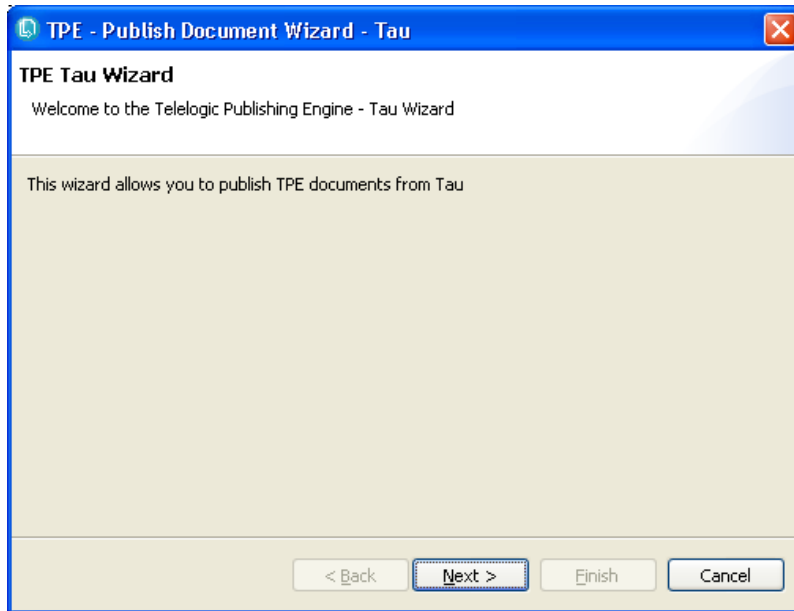


Figure 107

Select Next.

The content of the TPE Document Library is listed

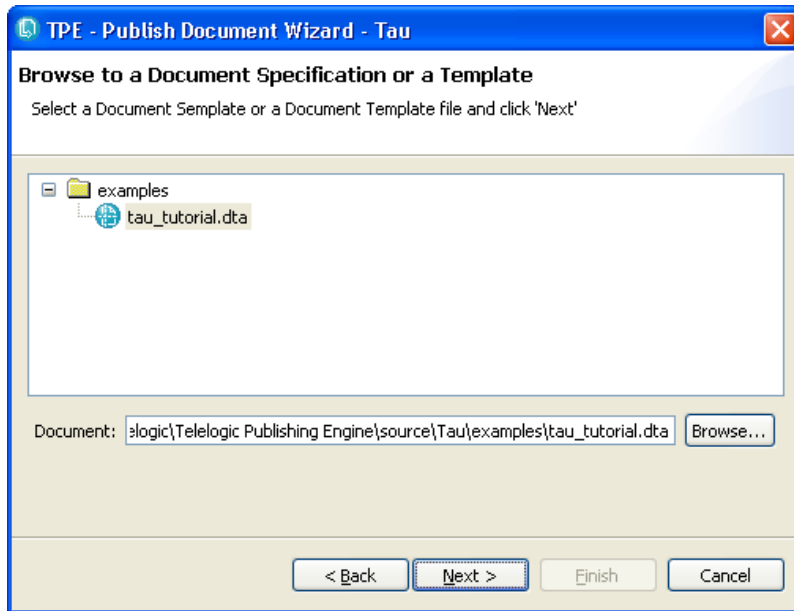


Figure 108

Select the desired document template/specification

NOTE The tree is populated with all the document templates and specifications found in the Shared Document Library.

The list of data sources for the current template is displayed

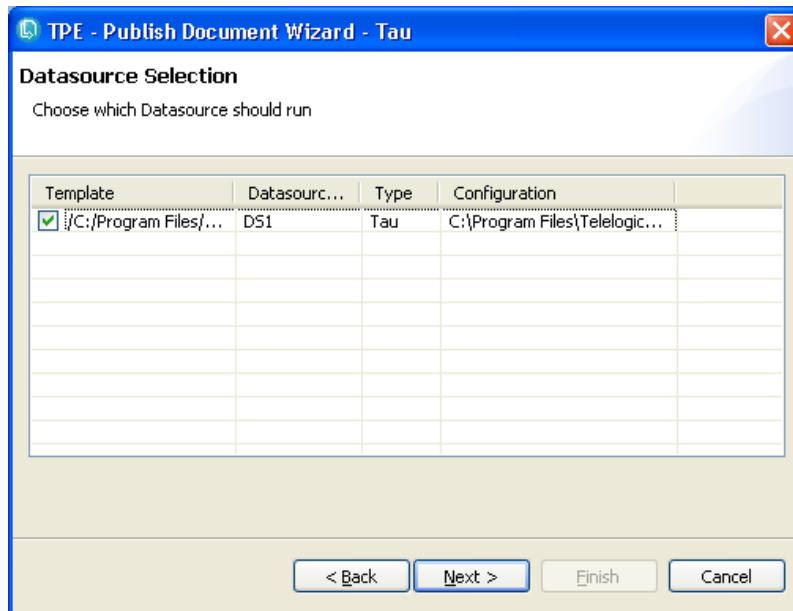


Figure 109

NOTE Unchecking a data source will mark it as ignored. All the queries belonging to ignored data sources are discarded when producing the output document.

The current Tau project is assigned by default to all the Tau data sources in the template. You can change the association as needed using the browse button available in the Configuration column.

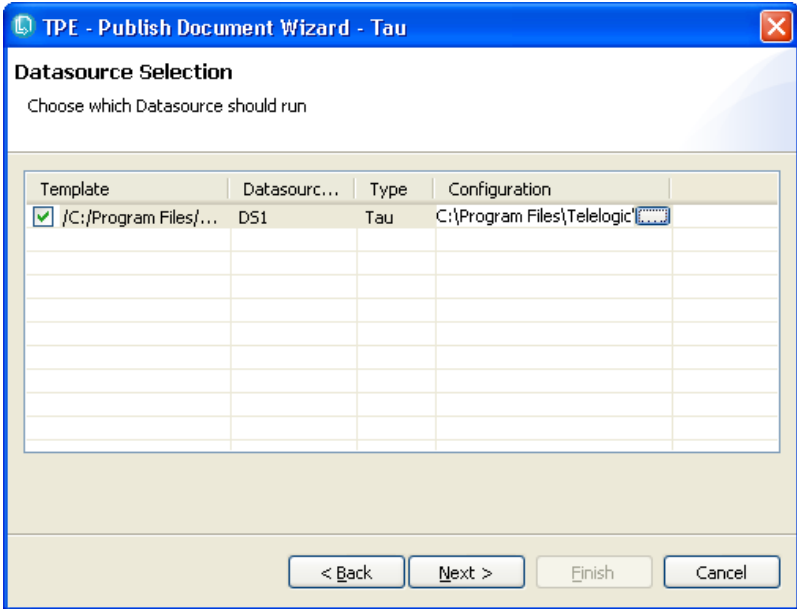


Figure 110

Select next.

You can now select the output types you are interested in.

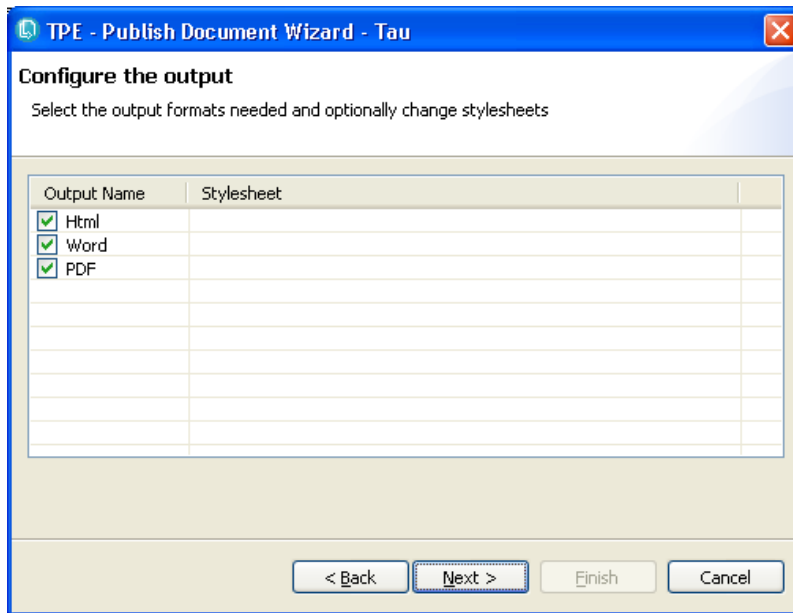


Figure 111

For any output type you can select the stylesheet to use.

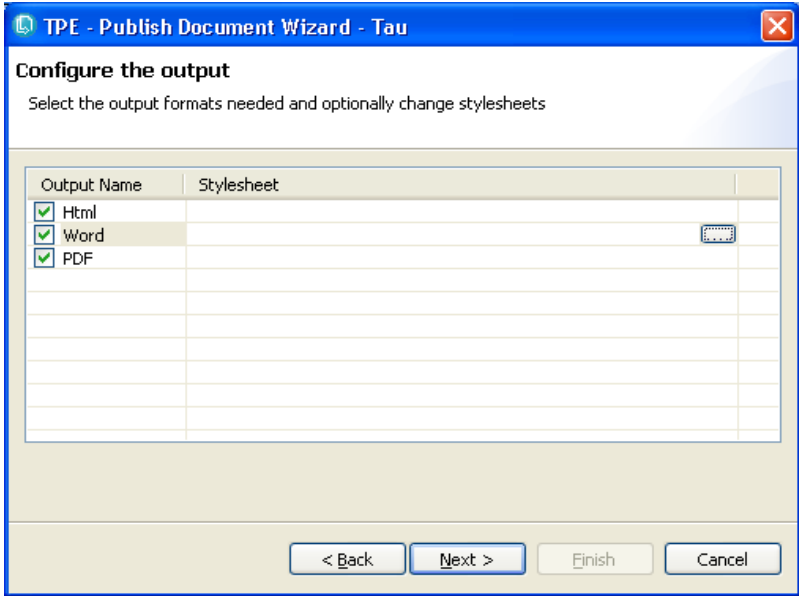


Figure 112

Select the Word stylesheet provided with TPE.

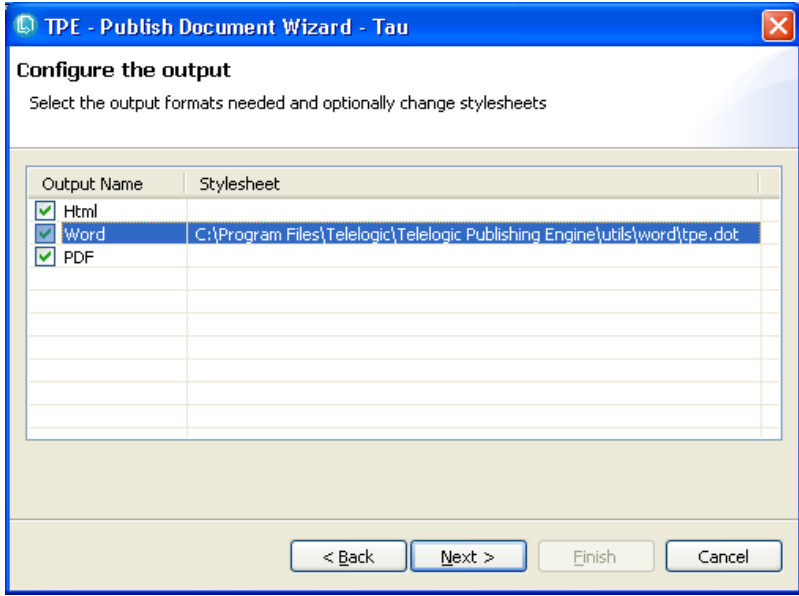


Figure 113

Choose the runtime options.

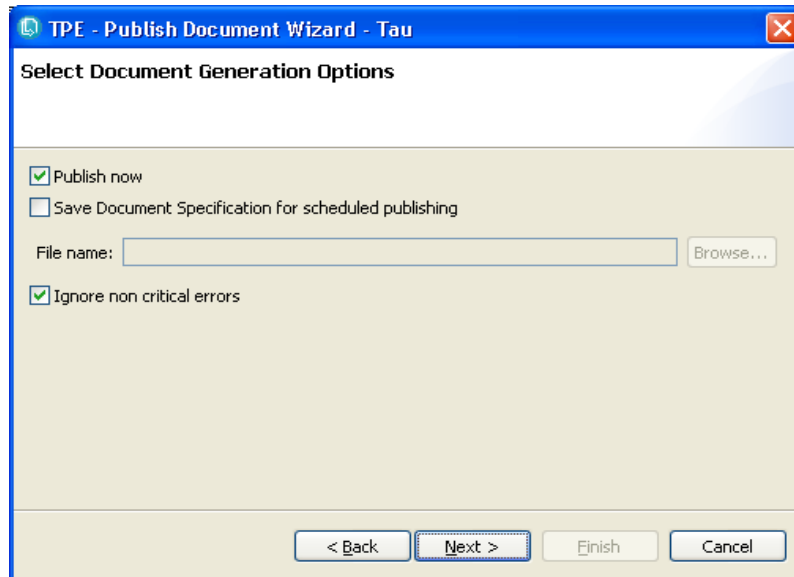


Figure 114

Property	Description
Publish now	If unchecked the document generation is not started
Save document specification for scheduled publishing	Checking this flag will allow you to select a location where the document specification will be saved. The document specification reflects all the options you've made in the wizard.
Ignore non-critical errors	If checked it will instruct TPE to proceed with the document generation if not every data source is properly configured.

Deployment scenarios

Local engine

This is the most common usage scenario for TPE. In this deployment scenario TPE is fully installed on the local machine and all document generation tasks occur are performed on the local machine as well.

Remote engine

This usage scenario is meant to support document generation on a server machine in order to minimize the resource consumption on the client machine.

TPE 1.0 ships with the web service implementation. In addition, TPE Launcher can use a remote installation in addition to the local one.

NOTE TPE 1.0 does not offer a separate installation package for this deployment scenario. You do need to install the minimal TPE configuration.

Deploying the web service

The setup example is given for Apache Tomcat but TPE should work with any application containers supporting Axis2.

Server Setup

Install and configure the latest version of Apache Tomcat on the server machine. A TPE compatible JRE is required along with DOORS 9.1 client software and Tau 4.2 if generating documents from these sources is to be supported.

Still on the server machine, after starting Tomcat, type the following URL in the browser:

<http://localhost:8080/manager/html/list>.

The admin username/password is specified in "\conf\tomcat-users.xml" in Tomcat's installation folder.

In the *WAR file to deploy* section, browse to *tpe.war* from the TPE installation and deploy it. You can check if the web service was deployed successfully by accessing the following URL:

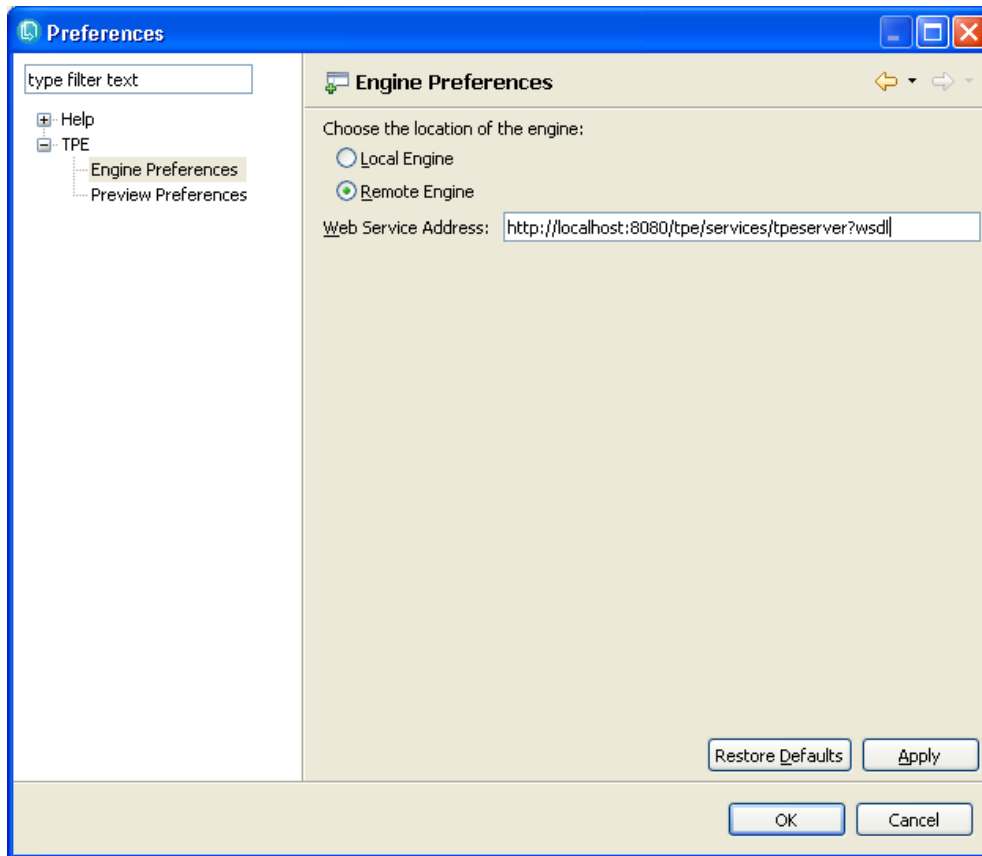
<http://localhost:8080/tpe/services/tpeserver?wsdl>

TPE currently requires that the *tl_lic.dll* is in a system path folder. Either append the "%TPE_HOME%\lib" folder to the system path, or copy the "%TPE_HOME%\lib\tl_lic.dll" to a folder in specified in %PATH%.

Client setup

Install TPE on a different client machine, open the TPE Launcher application and edit the Engine Preference. Set the Remote Engine Address property to match the URL of the published webservice.

Error! Hyperlink reference not valid.



NOTE Please consult your system administrator for the URL of the published webservice.

How to

Tips & tricks

Editable elements

You can edit the content for: text, styled text, image, include file, bookmark and hyperlink. Double clicking any of the mentioned elements will open the content editor dialog.

Table of Contents

If you want your output to contain a table of contents you can define a Table Of Contents in your template or you can define the table of contents in the stylesheet.

If you define the Table of Contents (or Table of Figures, Table of Tables) in the template, the table will not show in Microsoft Word until you update the document fields. This can be done using the “Update Fields”/“Update Table” feature in Microsoft Word or using the macros provided by TPE.

Figure captions

Just as the table of contents, the figure and table captions are not automatically updated. The resolution applied for Table of Contents can be applied here.

Included files

TPE handles Include File elements differently, depending on the output format. For PDF and HTML the included file is presented as a hyperlink while for Microsoft Word output an “INCLUDE TEXT” field is generated. This means that TPE will delegate the task of importing the file to Microsoft Word.

An included file is not visible in the output document until all fields are updated. The resolution applied for Table of Contents can be applied here.

NOTE_A Word document linking other files is not self contained. Moving the document on other machines will prevent you from visualizing the content of the linked documents. If this is needed you have to include the content of the linked files using the “Break links” feature (Alt+E+K in Office 2007) from Microsoft Word or the “includeLinkedFiles” macro provided with TPE.

Heading styles

If you want to use the predefined heading styles for Microsoft Word (Heading 1, Heading 2 etc) and HTML (H1, H2 etc) the style name to use in TPE is 1,2, ..., 9. While there is no concept of heading style for PDF, TPE will use internally defined heading styles.

Formatting properties vs. Styles

You should favor defining styles in TPE instead of changing individual formatting properties for template elements.

TPE Styles vs. External Styles

While TPE offers a great deal of flexibility in defining formatting, if your main output is Microsoft Word or HTML, you can and should use external styles (defined in stylesheet) as much as possible. This approach allows changing the appearance of the output document on the fly and enforces a uniform look company wide.

Numbering headings for Microsoft Word

For headings to be numbered as a hierarchical list the easiest way is to use a stylesheet that has the headings numbered.

Dynamic images

TPE can include images whose path is not known at design time. To achieve this you need to:

- Add an image element to the template
- Double click the image to edit its content
- Define the path as a static, data or scripted expression

Dynamic included files

TPE can include files whose path is not known at design time. To achieve this you need to:

- Add an include file element to the template
- Double click the element to edit its content
- Define the path as a static, data or scripted expression

Unicode data in output

All Unicode data will be rendered as long as the font used supports it.

For PDF additional configuration is necessary (see the PDF output section for details).

For Word output, if you used a non-Unicode font you can change that font in the output document after the document generation.

For HTML output, if you used a non-Unicode font you can change the font family in the stylesheet after the document generation.

For PDF output, if you used a non-Unicode font you need to generate the document again using true type Unicode fonts and setting the appropriate output properties (see the PDF output section for details)

Moving an element outside the visible editor region

TPE Document Studio is an Eclipse RCP application. One trait of Eclipse applications is how auto scrolling is handled. If you are trying to move an element into a template location not visible without scrolling, you need to do the following:

- Select the element
- Drag it near the top/lower margin of the editor so that a part of the figure goes beyond the margin, but not the whole figure
- Pause for a second or two. Scrolling should begin at this time

Table fit to window

When auto fit to window is specified for a table, TPE will resize the table's columns using the following algorithm:

- Columns (cells) with a specified width are not changed
- The highest index column with no width specified is resized to occupy all the remaining space between the table's margins and the document's margins

If all the columns have their width specified, no changes are made to the table's width.

NOTE this behavior applies only to Word output.

PDF Tables

Tables in PDF output will always occupy the full available width unless the cell's width is set. There is no equivalent for Word's "auto fit to contents".

NOTE this is a technical limitation that we look to address in future versions.

Table merge

On all output formats consecutive tables will have the aspect of a single table. However unless cell widths are explicitly set to the same size the columns of the resulting table will not have the same width for all rows.

Once per table

When tables are merged it is desirable for some rows to appear only once, a good example being the header rows. This behavior can be specified through the "once per table" property of the row element.

Formatting	
+ common	
- specific	
row index	
row break across pages	
row repeat at page beginning	false
once per table	false

Figure 115

NOTE The property does not for rows in tables coming from different template elements. See below:

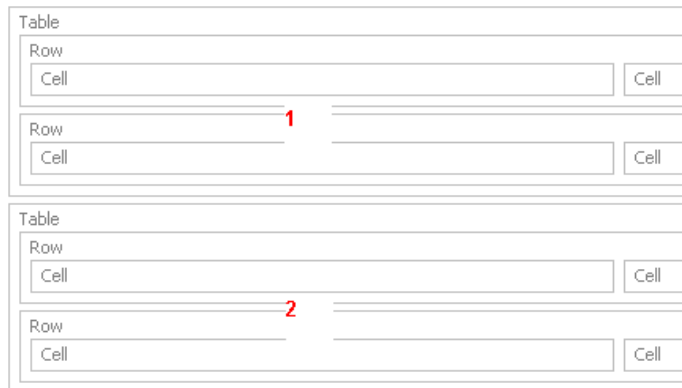


Figure 116

Frequently asked questions

Q: When Document Studio/Launcher started for the first time the menu actions do not work.

A: All the menu commands work but they are not visible until the welcome screen is closed.

Q: When inserting text with more than one line, only the first line is displayed

A: This behavior is by design. TPE Document Studio is meant to allow for the template designer to focus on the structure of the document.

Q: The forbidden mouse icon is displayed while expanding the branches in the Outline panel

A: The outline pane can be used to add/delete/copy/paste elements just as the main editor window. Hence, if a palette element is selected the mouse cursor will show the locations where that element can be dropped in the outline pane.

Q: DOORS headings are not numbered

A: The easiest way to achieve heading numbering is to have the Heading styles numbered in the stylesheet that you use.

Q: Opening a Document Specification does not load the template in the studio. You have to open the template first, then the Document Specification. If you open the Document specification first and then the template a new document specification is created.

A: This behavior is by design. This allows testing different document specifications without changing the current document template.

Q: I would expect that the tree in the right pane is already expanded

A: We are evaluating the possibility of this being an option

Q: TPE allows providing incorrect values to formatting properties.

A: TPE validates all properties values and will warn the user if the value is not correct. However TPE will not discard the value. All incorrect values are handled at generation time either by being ignored or by being replaced with valid values.

Troubleshooting TPE

Common problems

Problem	Resolution
TPE won't start	<ul style="list-style-type: none">• Java 1.6 or later needs to be installed on the machine
Cannot generate documents from DOORS Data Sources	<ul style="list-style-type: none">• A DOORS 9.1 or later client must be installed• Check that the data source is properly configured (the module path, view name and baseline are case sensitive)
Cannot generate documents from Tau Data Sources	<ul style="list-style-type: none">• A Tau 4.2 or later installation must be available• The path to Tau's bin folder must exist in the PATH system variable• Check that the data source is properly configured
Cannot run Word macros	<ul style="list-style-type: none">• Microsoft Word needs to be installed• The macro must be defined in the stylesheet
No OLEs in the output document.	<ul style="list-style-type: none">• OLEs are support for Word output only. In order to see get the OLEs in your Word document please follow the indications from the Output chapter.

NOTE For any problems you encounter during TPE usage we recommend sending the log files mentioned bellow to the support team.

TPE Core Logging

TPE logs all the activity to the console and to a file on the file system. The console will display human readable information while the file contains detailed information usable by TPE support. The logging behavior is controlled through the *log4j.properties* file located in %TPE_HOME%.

NOTE The system variable TPE_HOME must be defined for TPE to be able to read the logging configuration.

NOTE The default output of the log file is Telelogic\tpc.log in the user home folder, usually: C:\Documents and Settings\\Telelogic\tpc.log

TPE UI Logging

TPE Document Studio and TPE Launcher, like any other Eclipse RCP applications, log all user interface related problems in their own log files. These log files can be found in the workspace for Document Studio and Launcher. The location of the workspaces, unless changed by the user are:

C:\Documents and Settings\\Application
Data\Telelogic\TPE_20081024\Studio\workspace

C:\Documents and Settings\\Application
Data\Telelogic\TPE_20081024\Launcher\workspace

The log files are located inside the workspace in the .metadata folder.

NOTE These log files may not exist

Examples

A template for DOORS data

Basic setup

Create a new Document Template

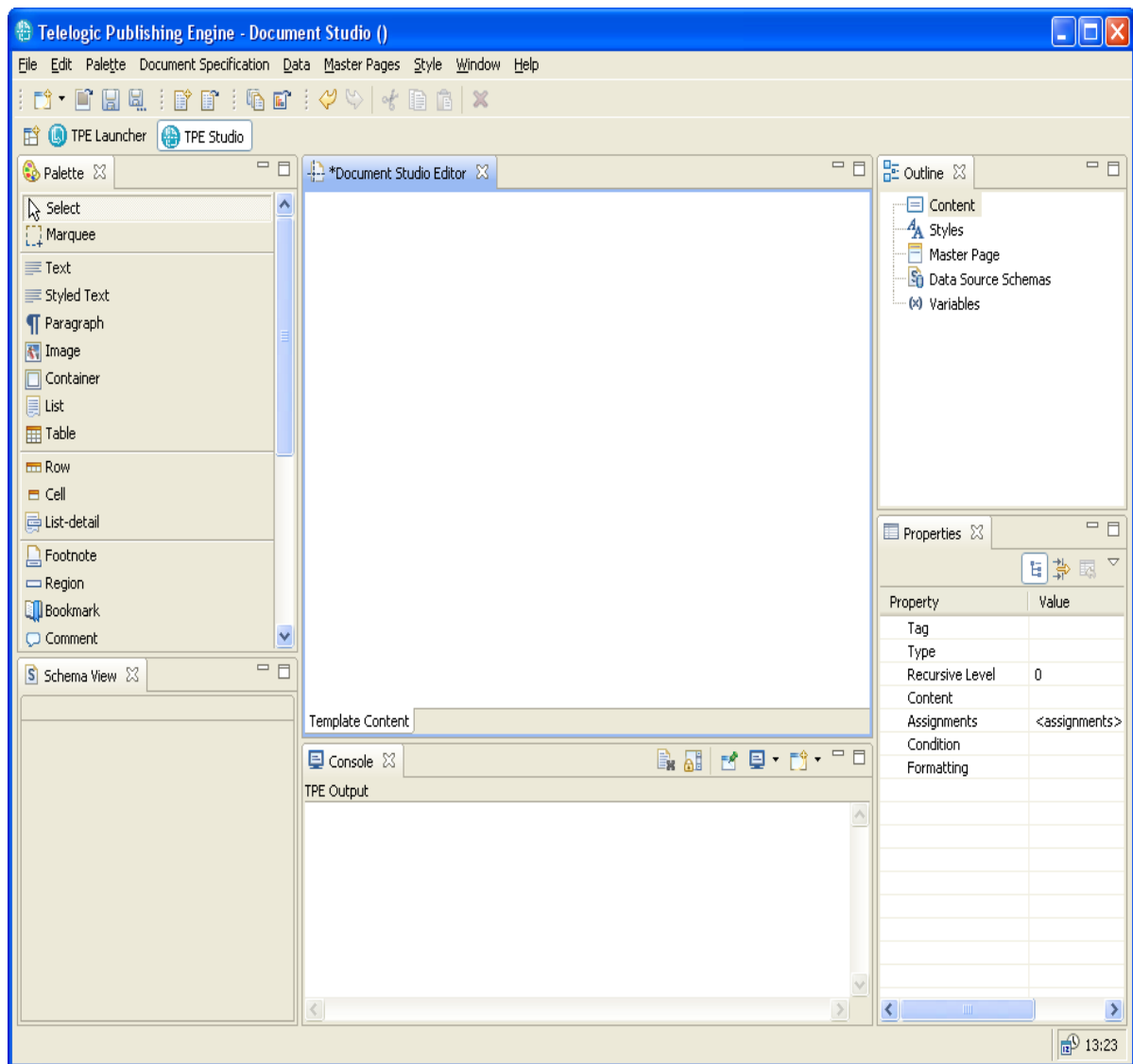


Figure 117

Add a data source schema using the Add Data Source Wizard

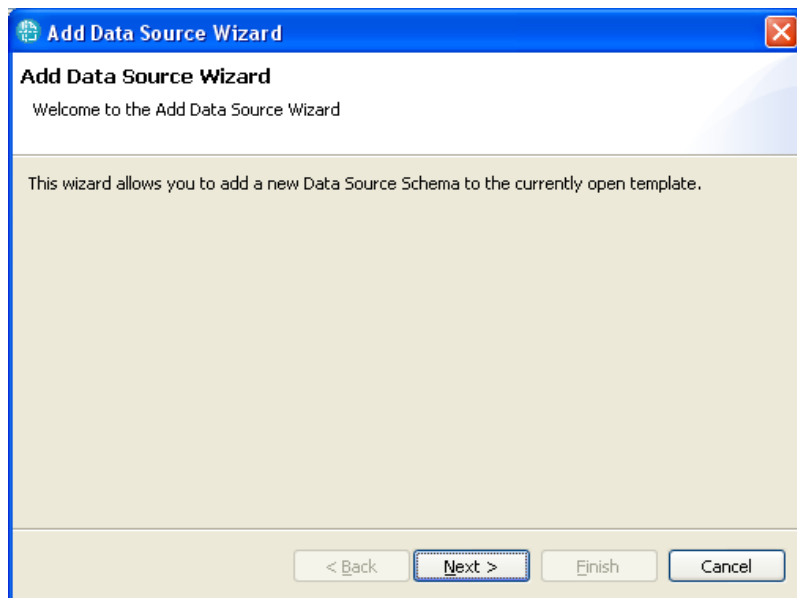


Figure 118

Select the schema provided in the TPE installation

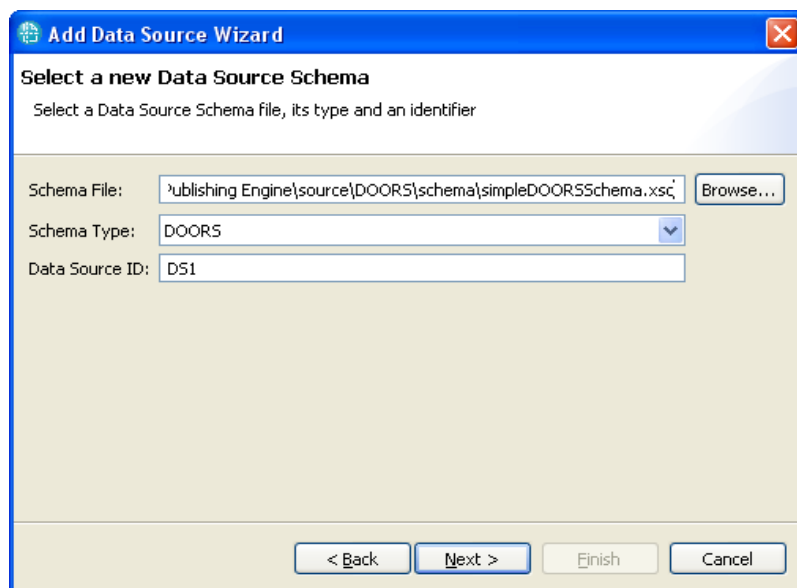


Figure 119

Review selection

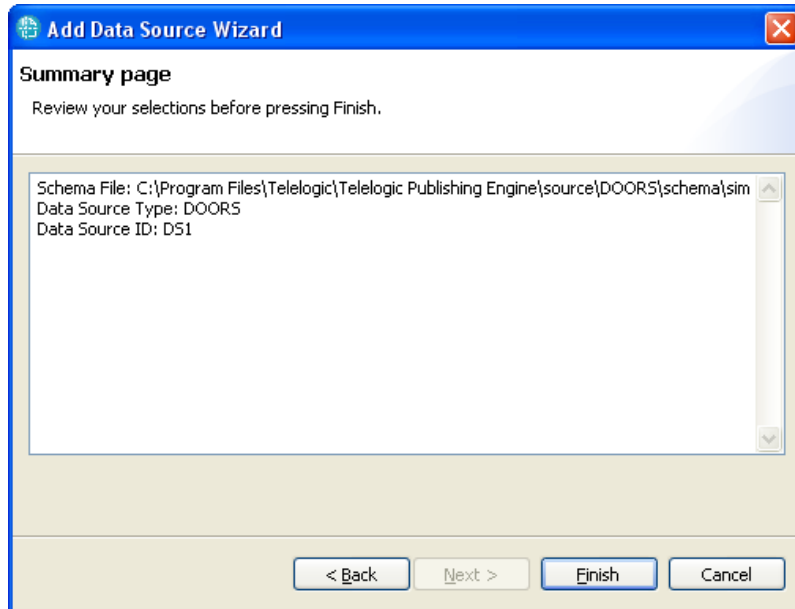


Figure 120

Add a **container** element in the template

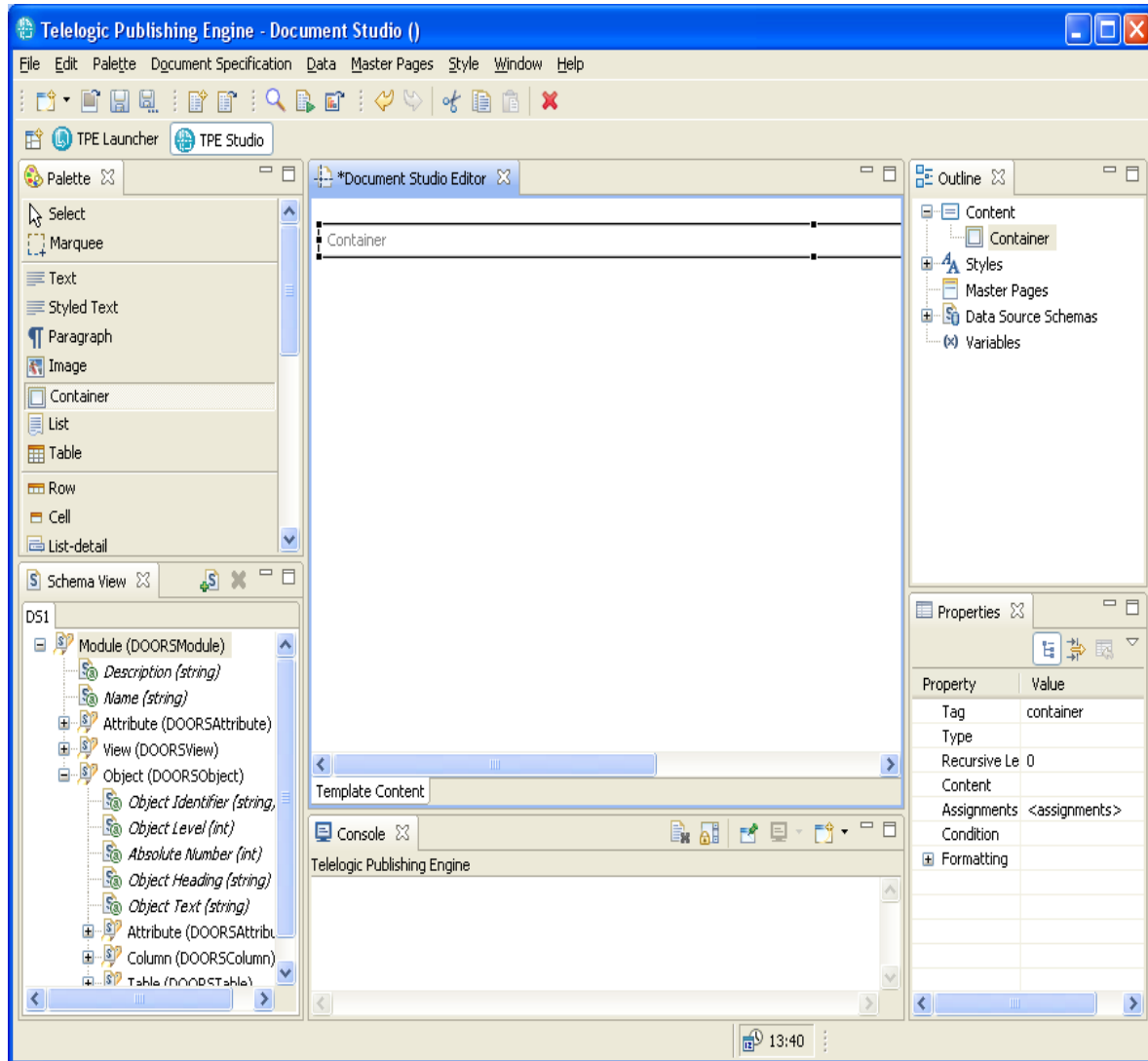


Figure 121

Assign a query to the element

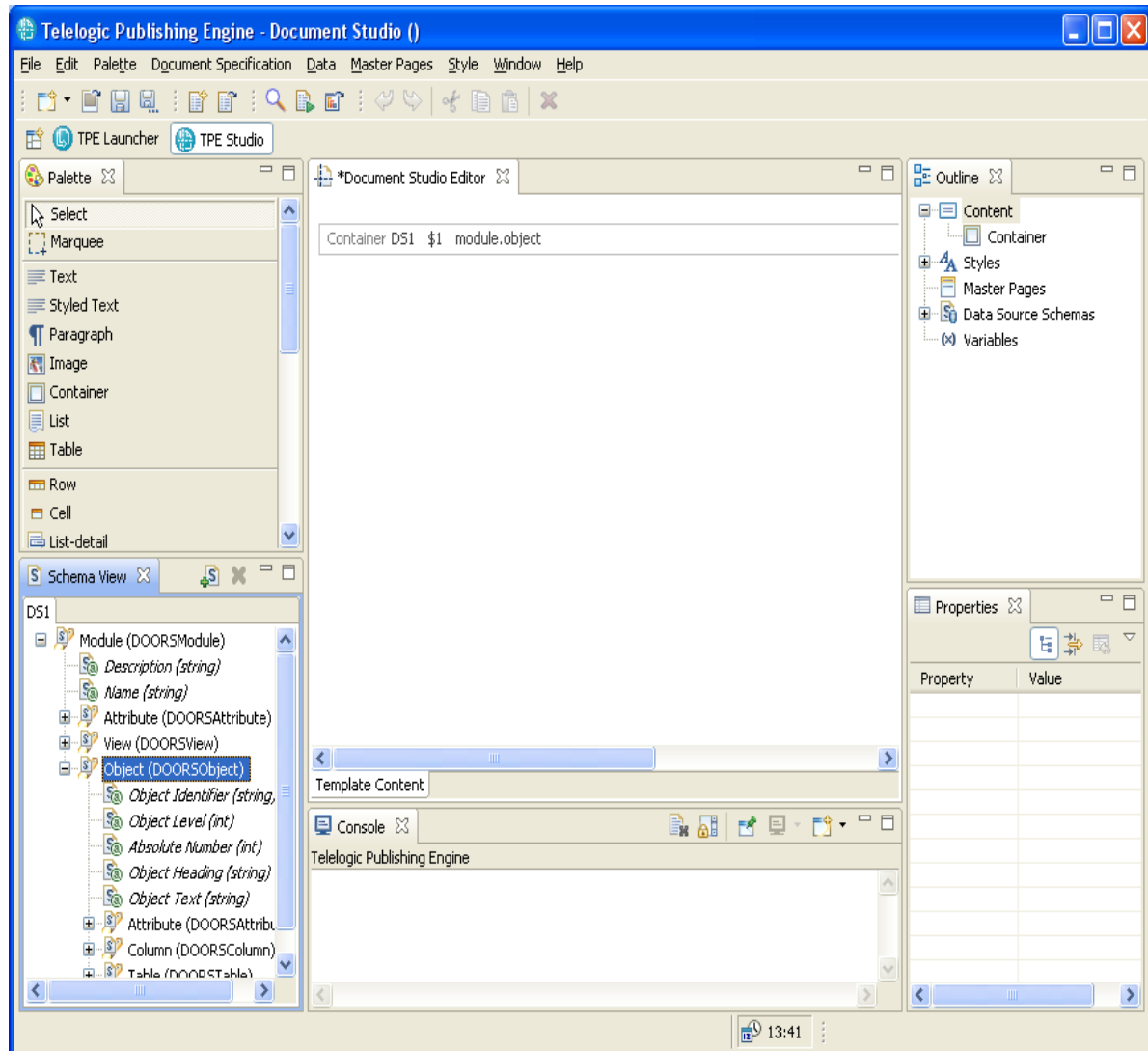


Figure 122

Add the "Object Heading" attribute

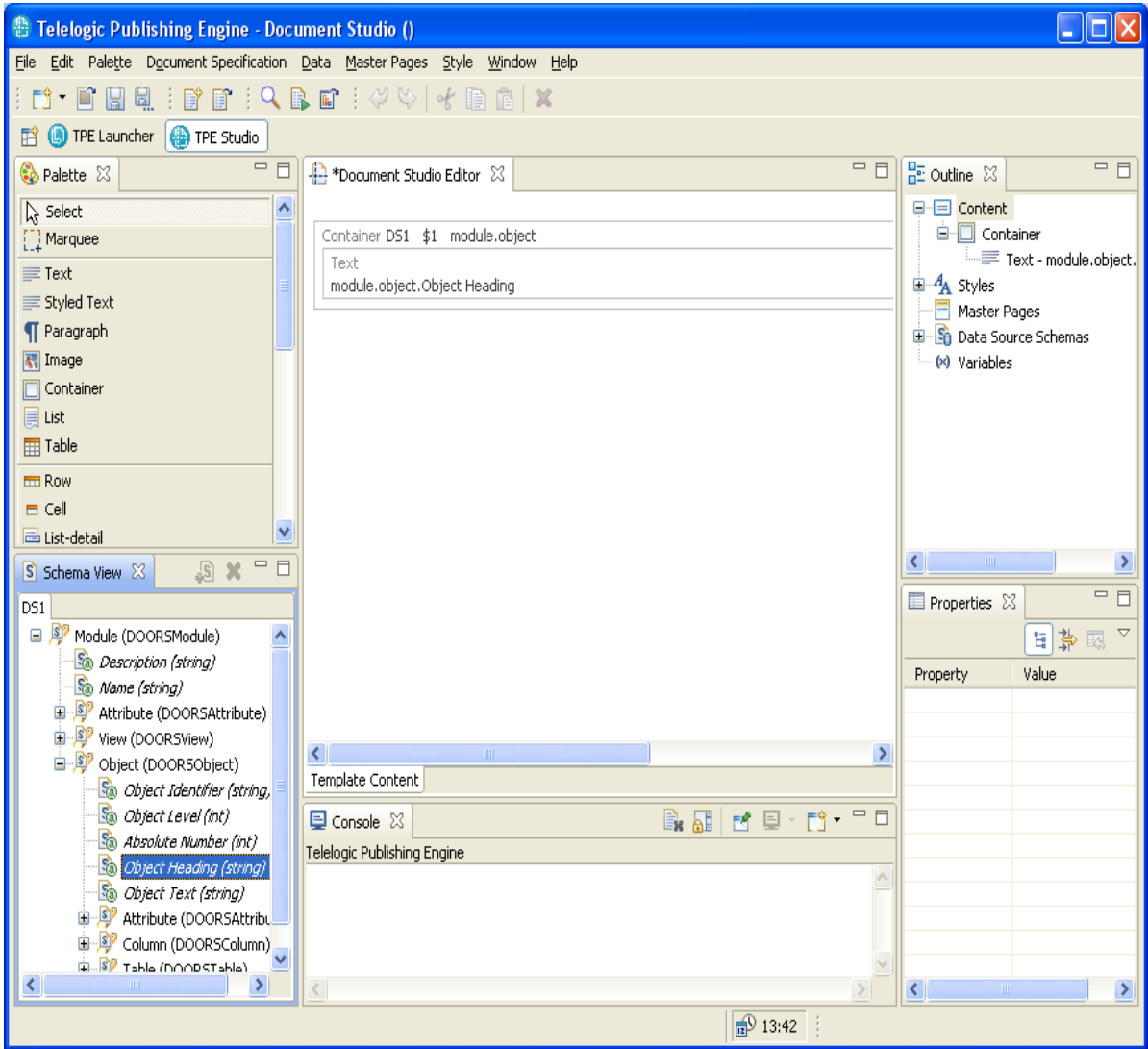


Figure 123

Apply a Condition to the display of the element

The text element containing the object heading should be processed only if the Object Heading is not empty.

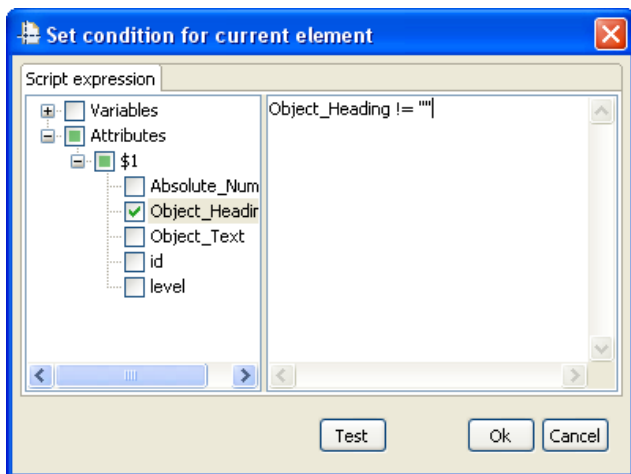


Figure 124

Add the Object text

The Object Text is added in its own paragraph to avoid collating multiple object texts on the same line.

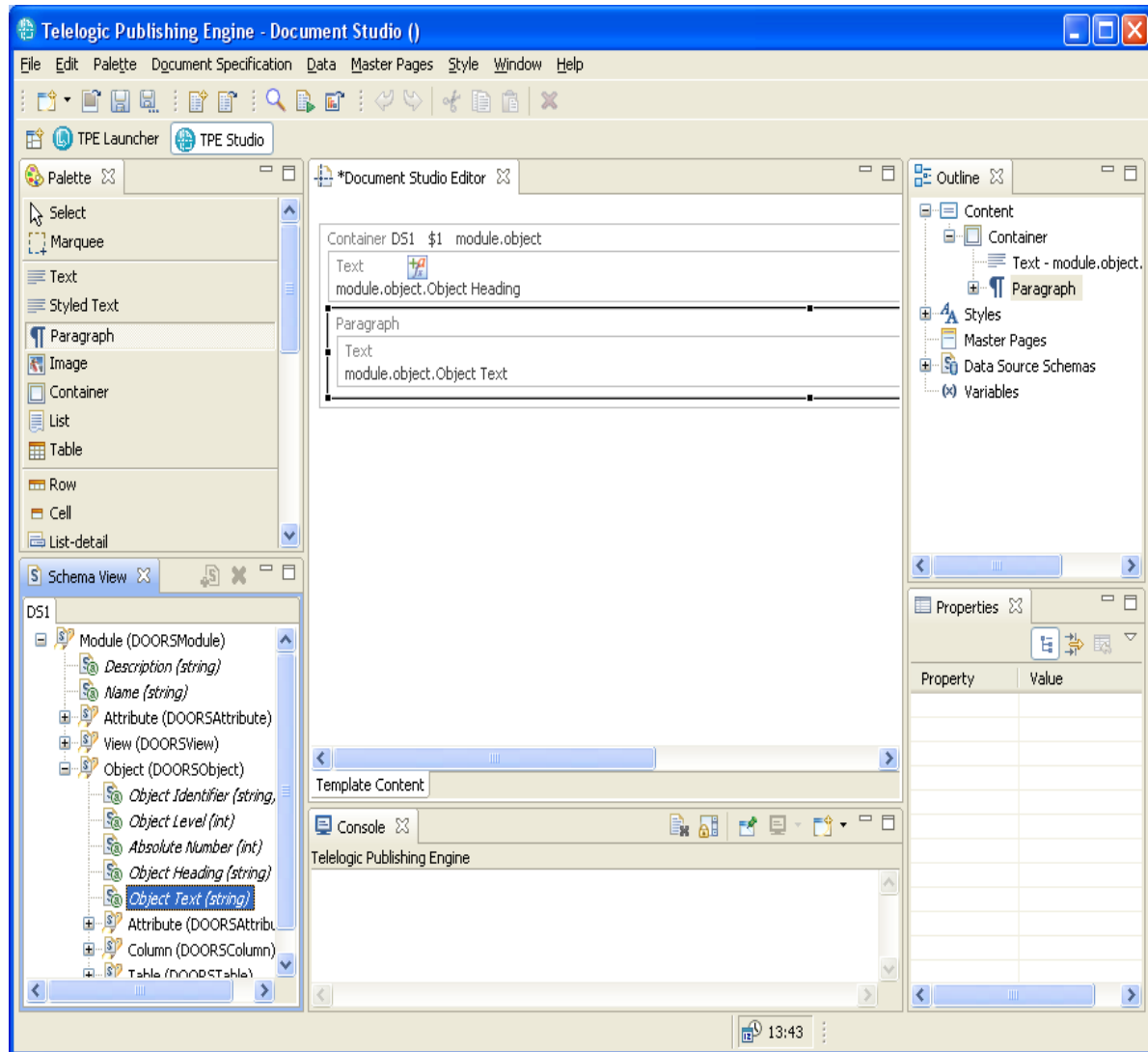


Figure 125

Condition the display of Object Text

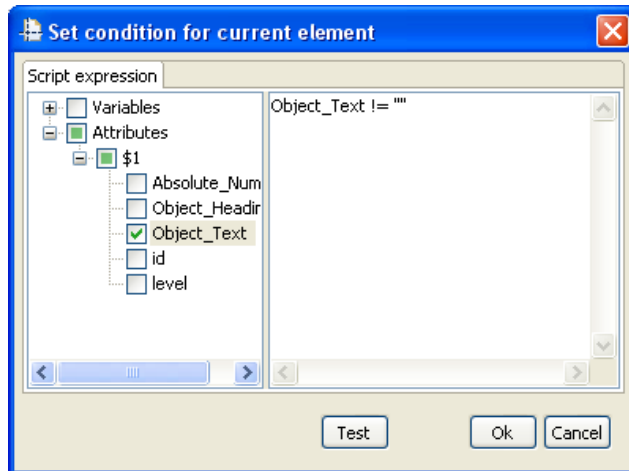


Figure 126

Save the template

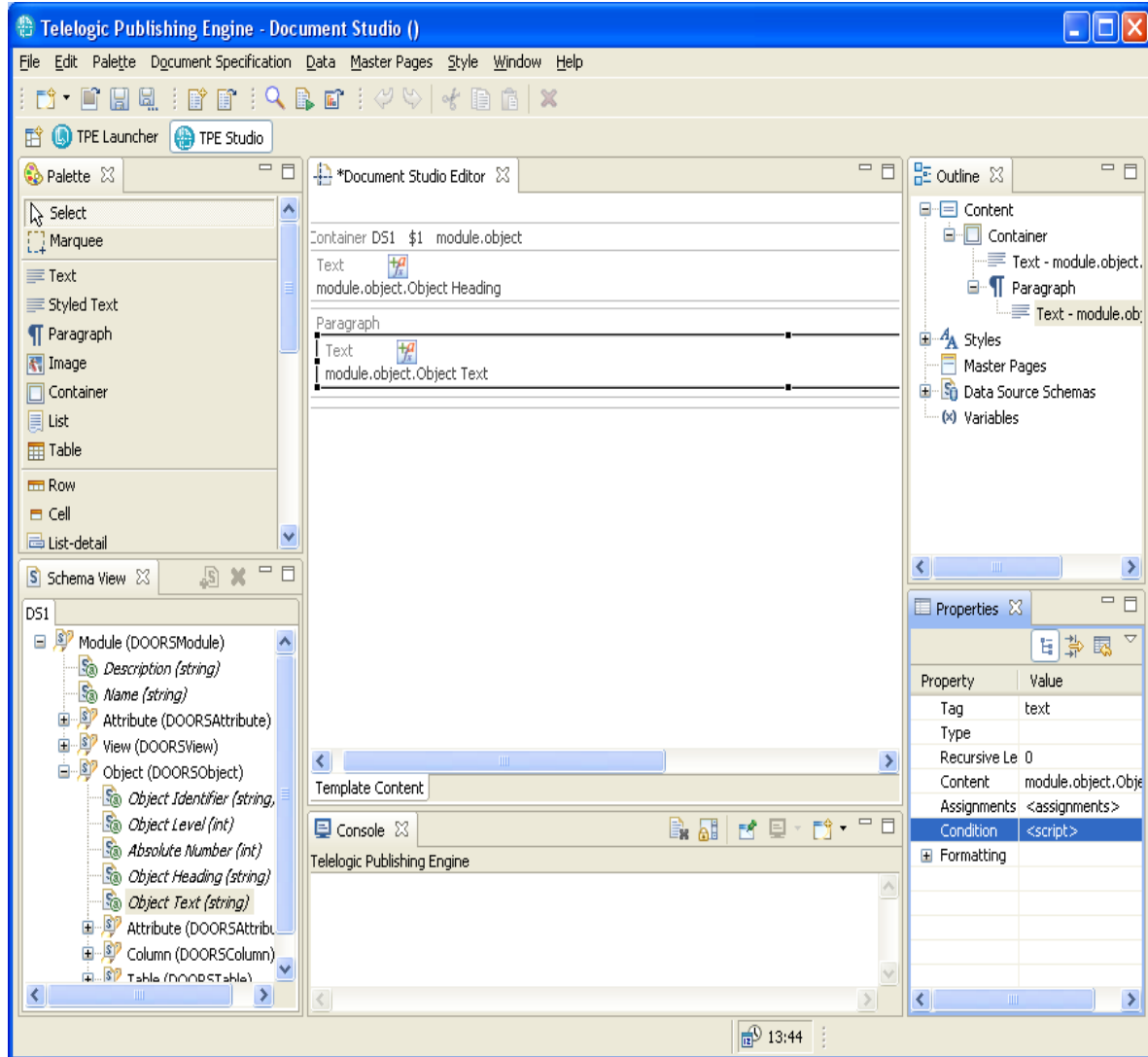


Figure 127

Advanced template options

Add a Table of contents

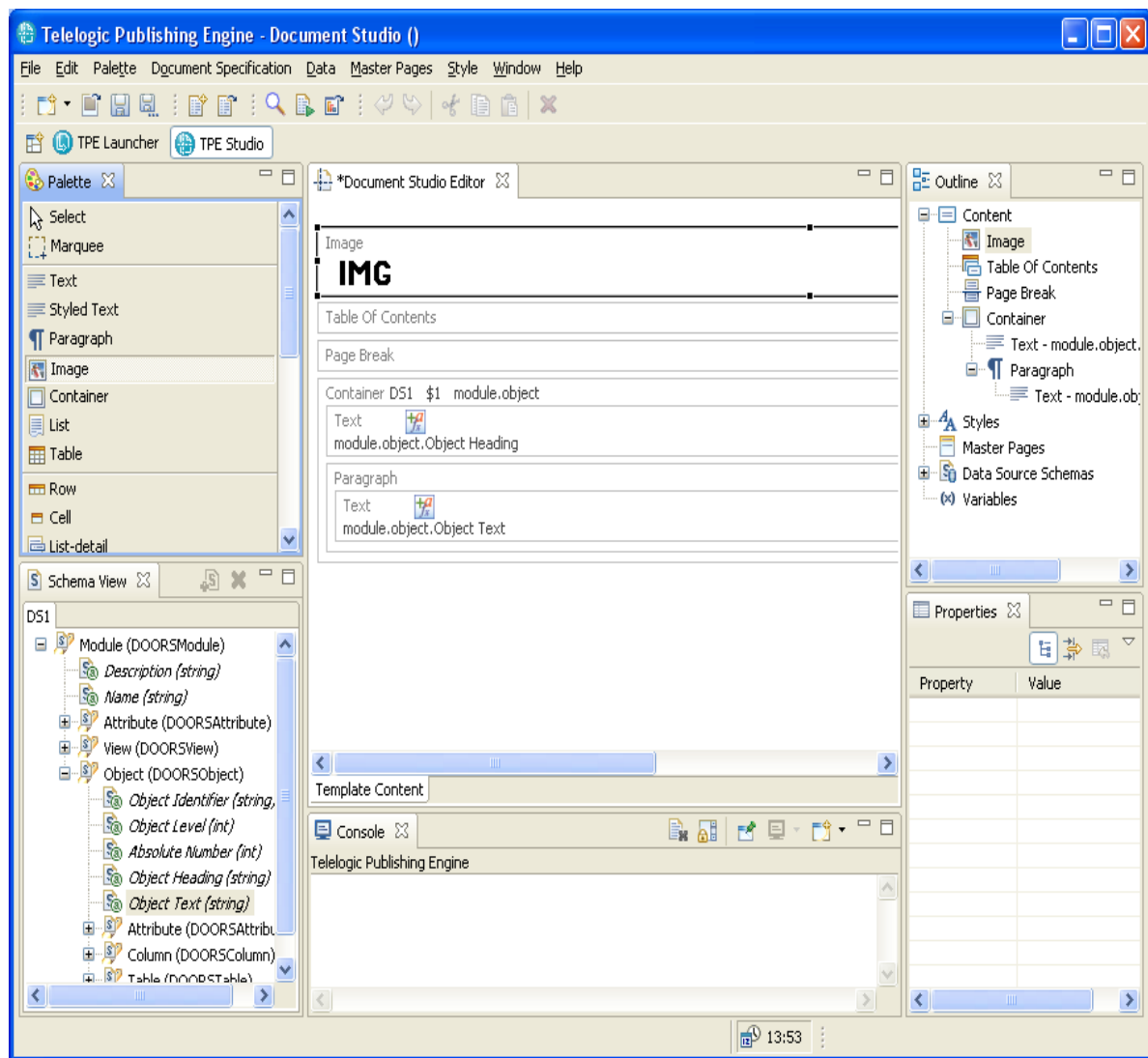


Figure 128

Add an image at the top of the document

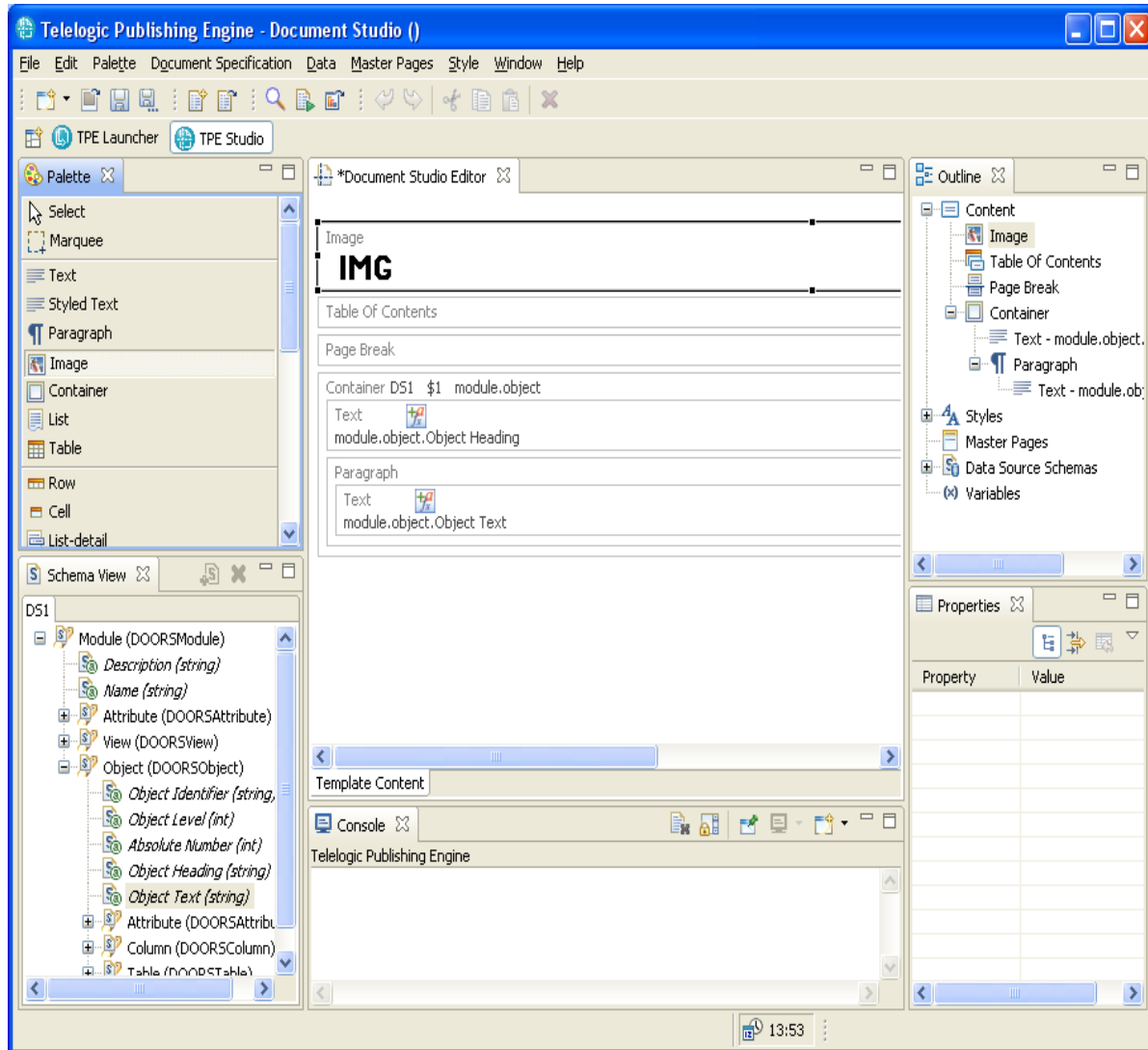


Figure 129

Load an image file from the file system

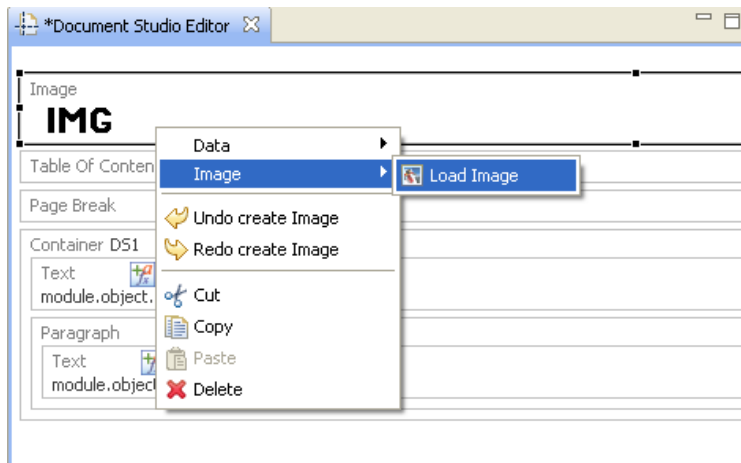


Figure 130

NOTE The selected image is copied in the template.

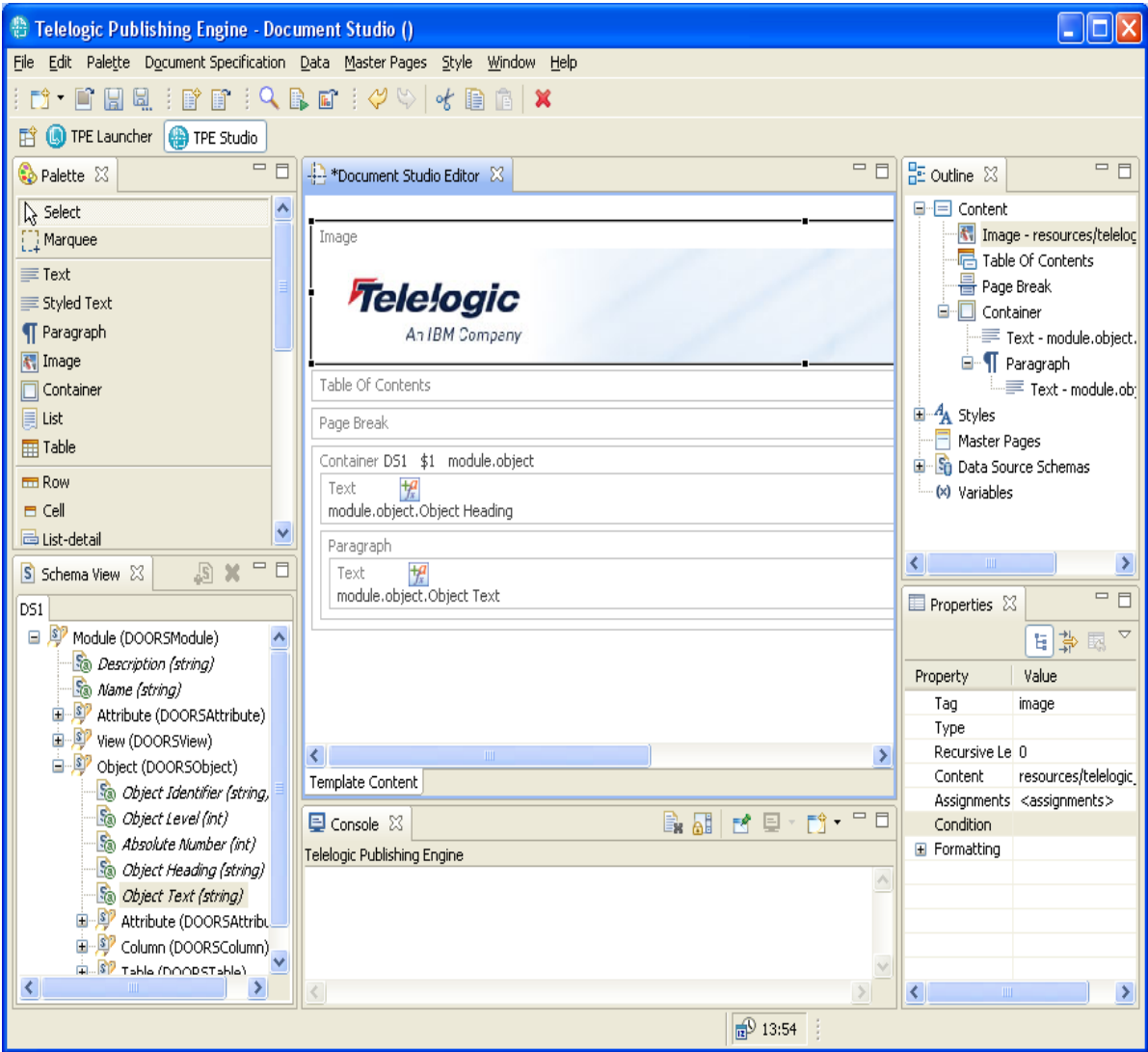


Figure 131

Add a title to the document

Add a paragraph with a text element

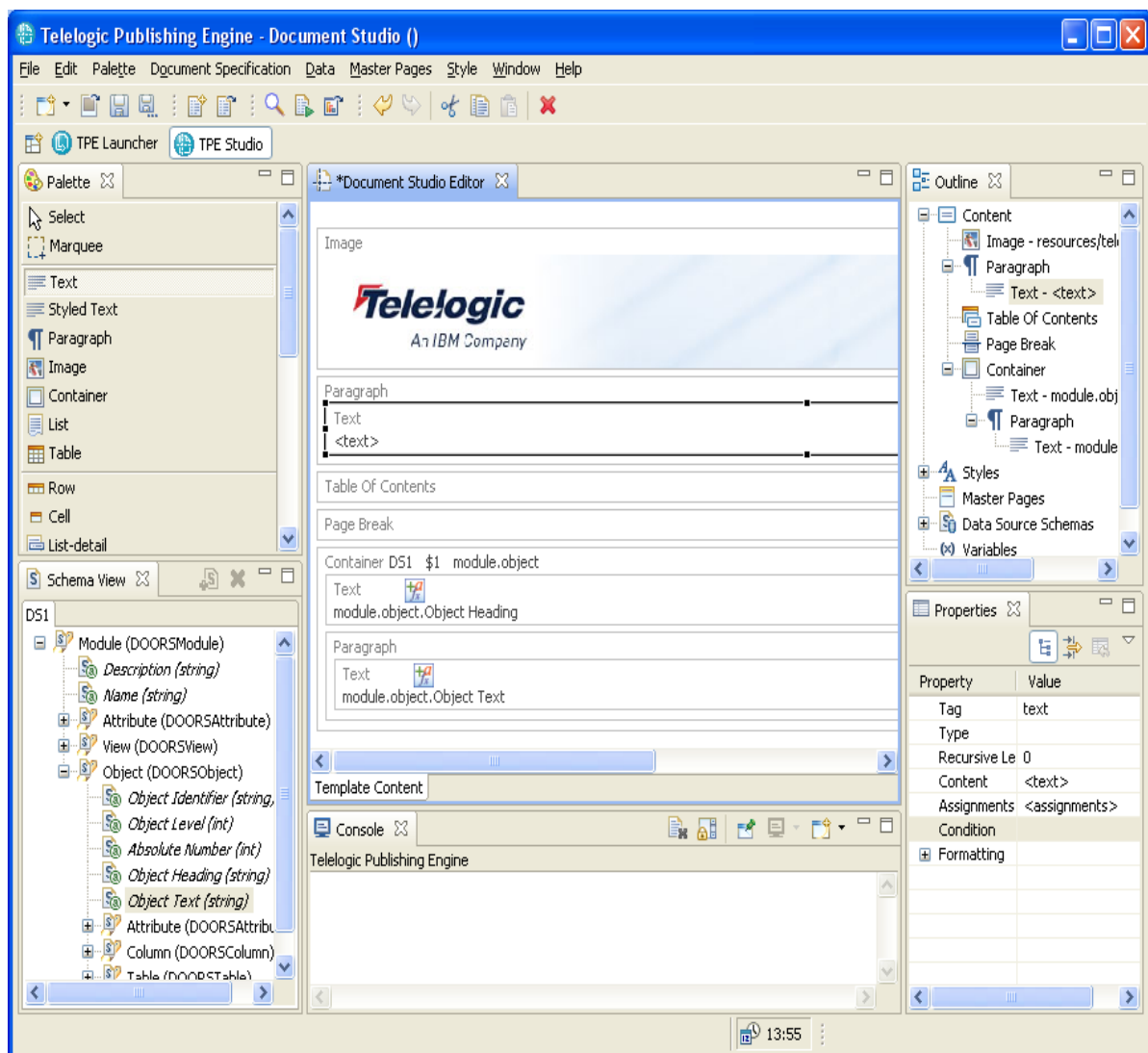


Figure 132

Set the text

Double click the text element or click the “Content” in the property page.

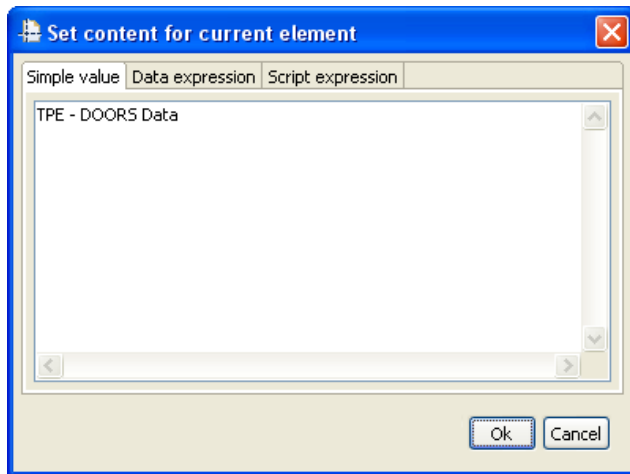


Figure 133

Define a title style

The newly created style is named “TPE_Title”

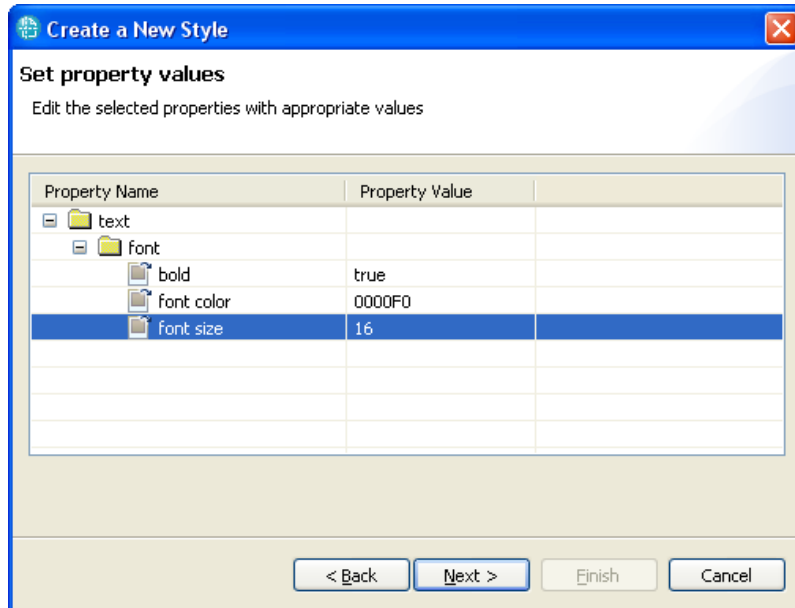


Figure 134

Set the style to the paragraph

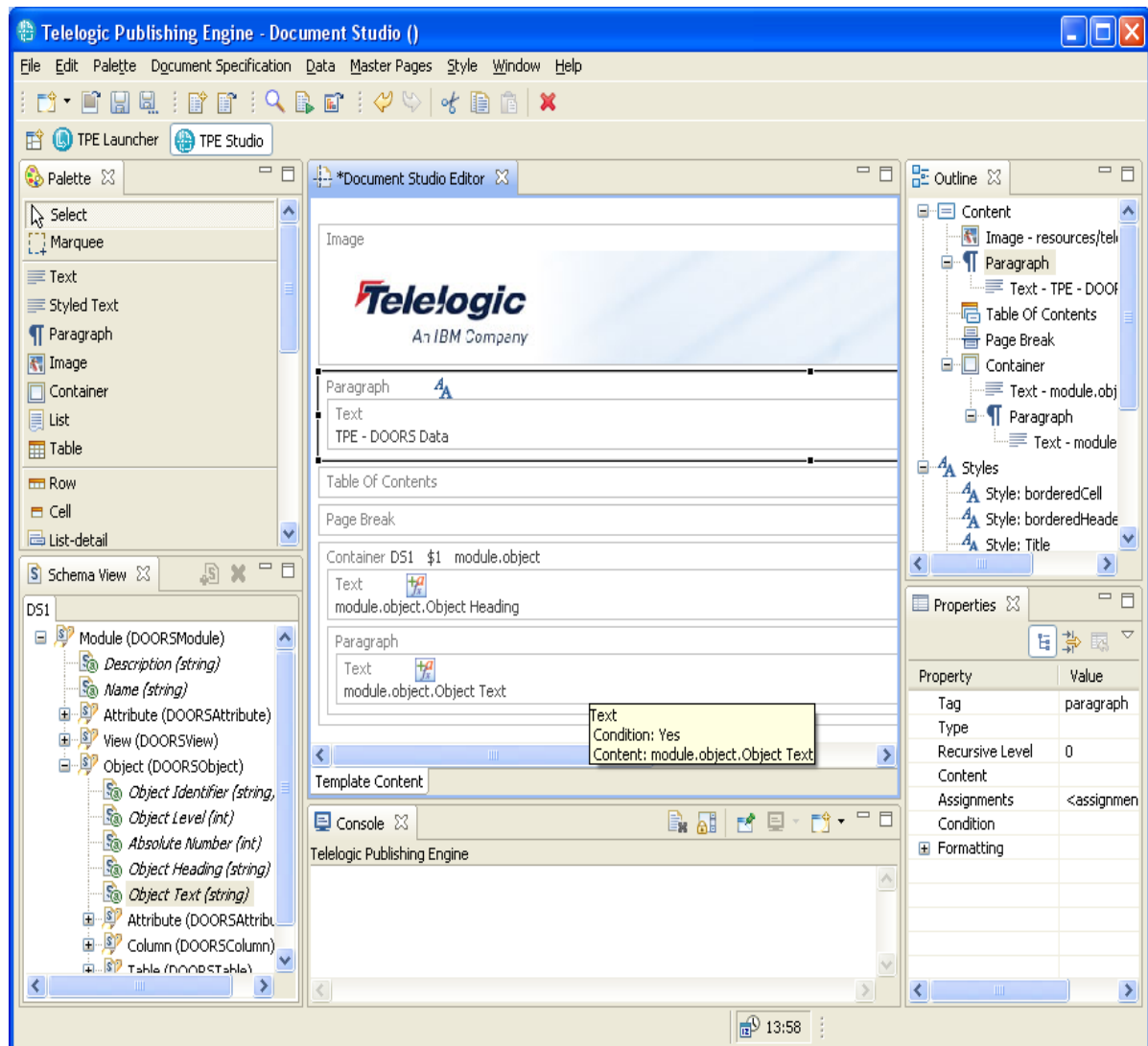


Figure 135

Create a master page for the data

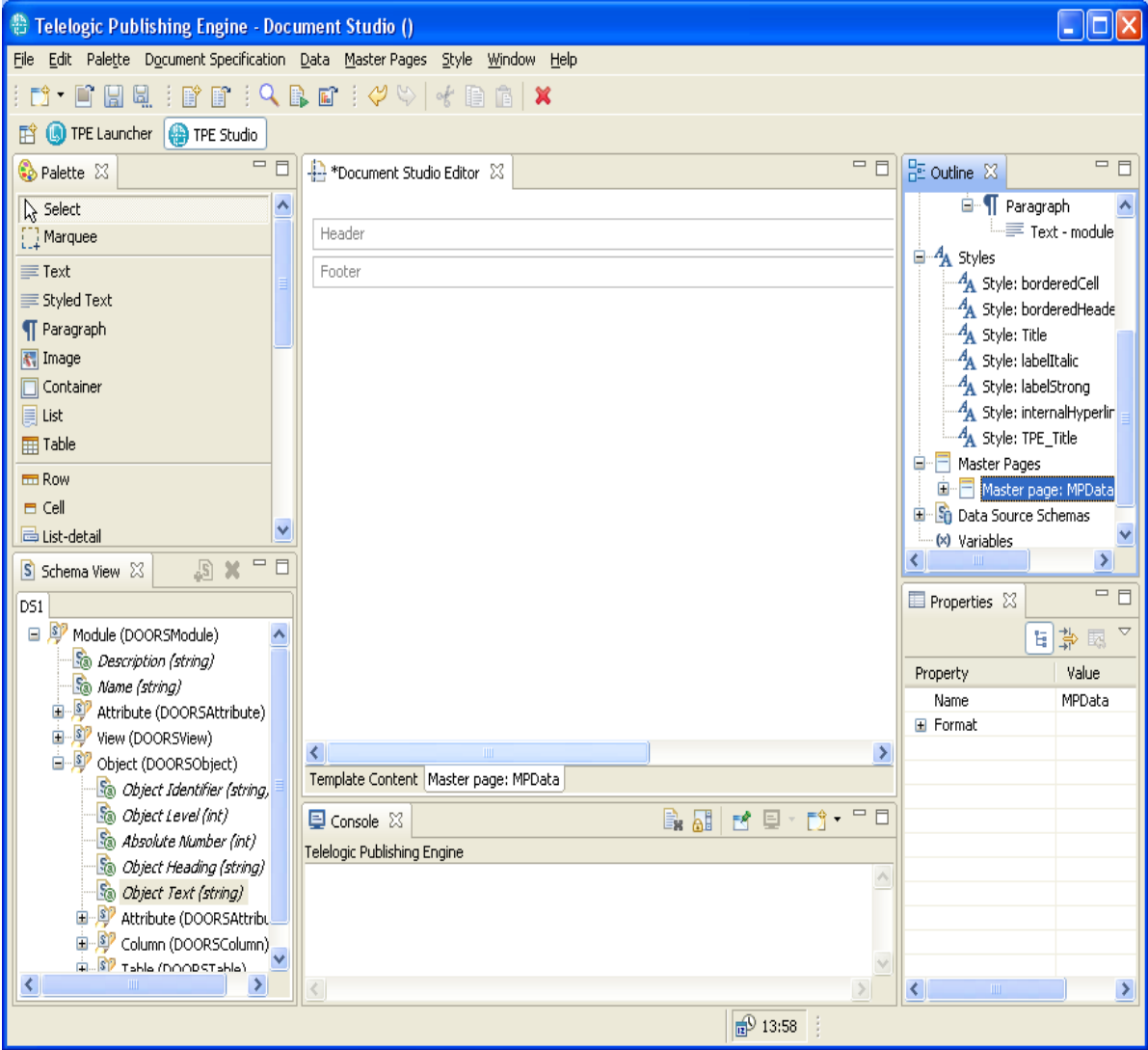


Figure 136

Add some static text in the page's header

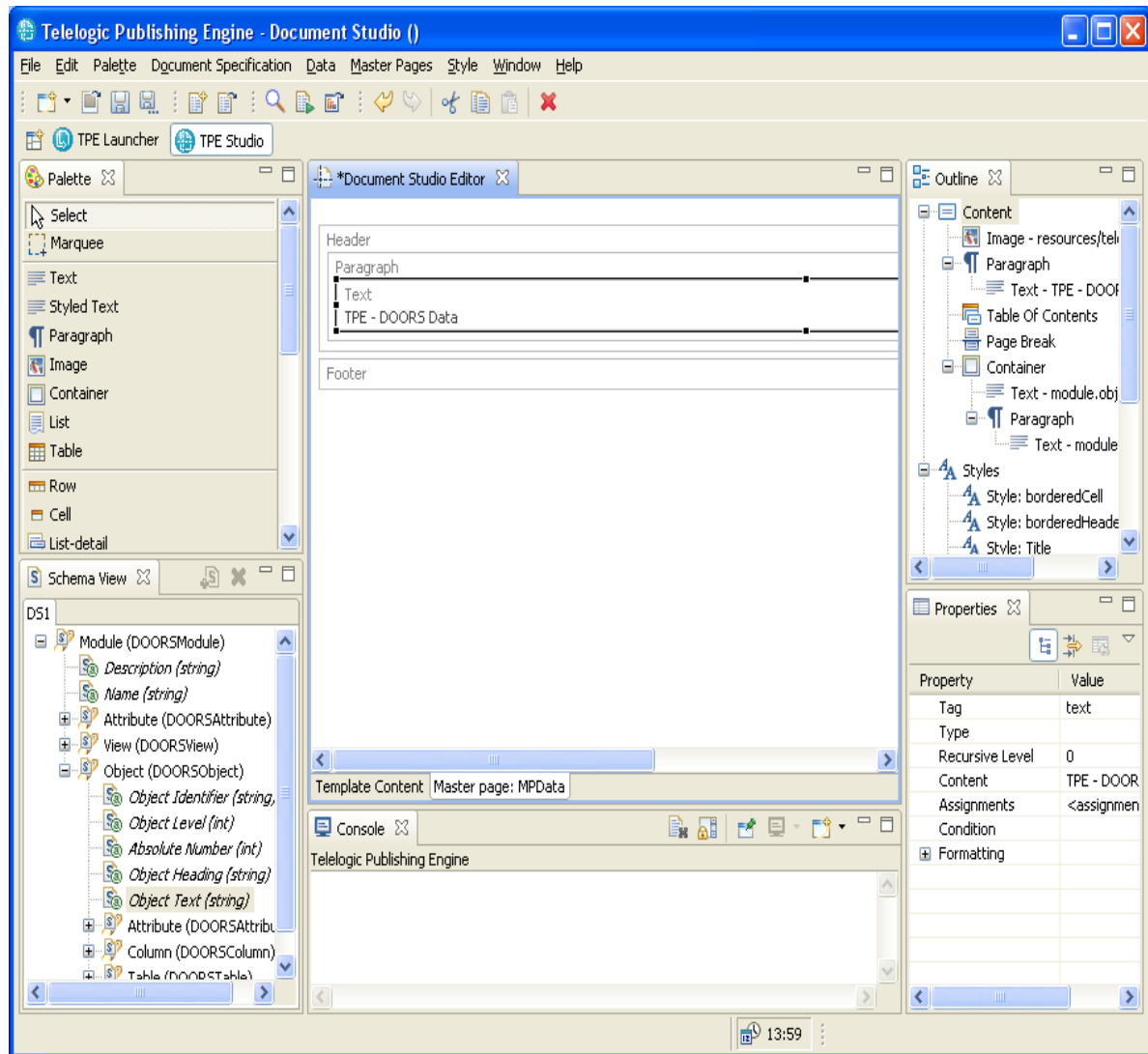


Figure 137

Add a page number field to the footer

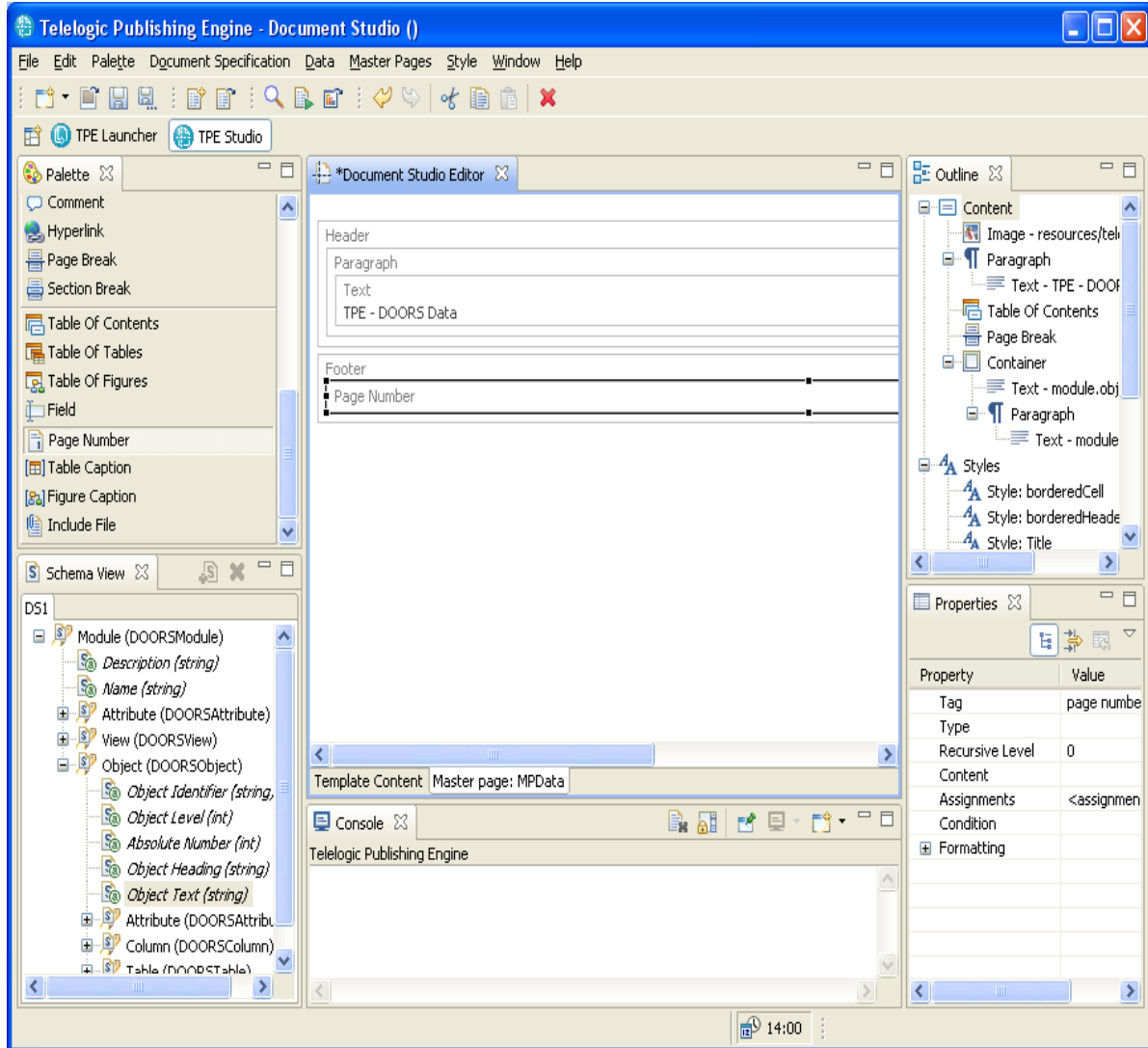


Figure 138

Assign the page

The master page is assigned to the container element.

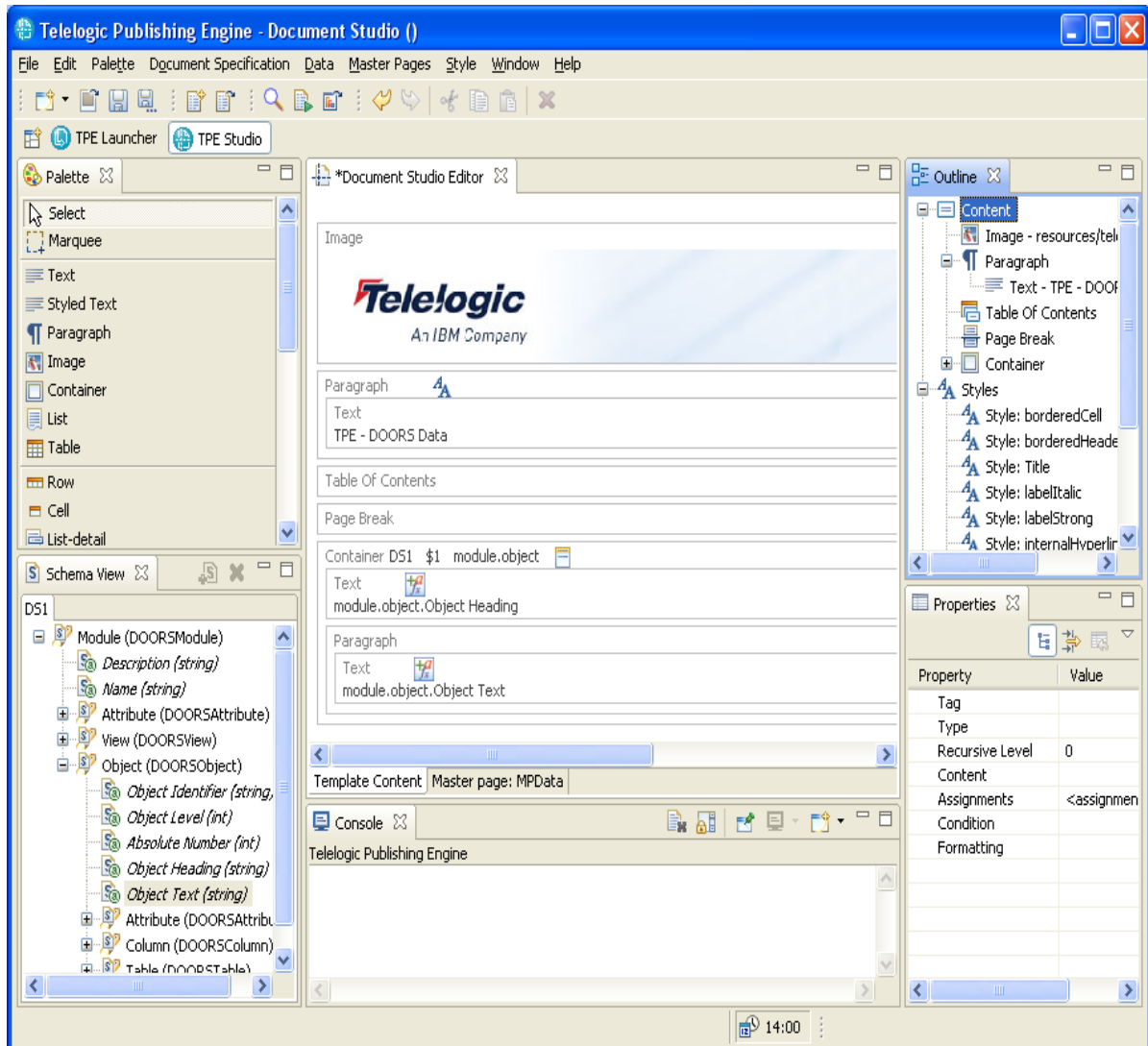


Figure 139

More advanced settings

Define a variable

The variable name is AuthorName.

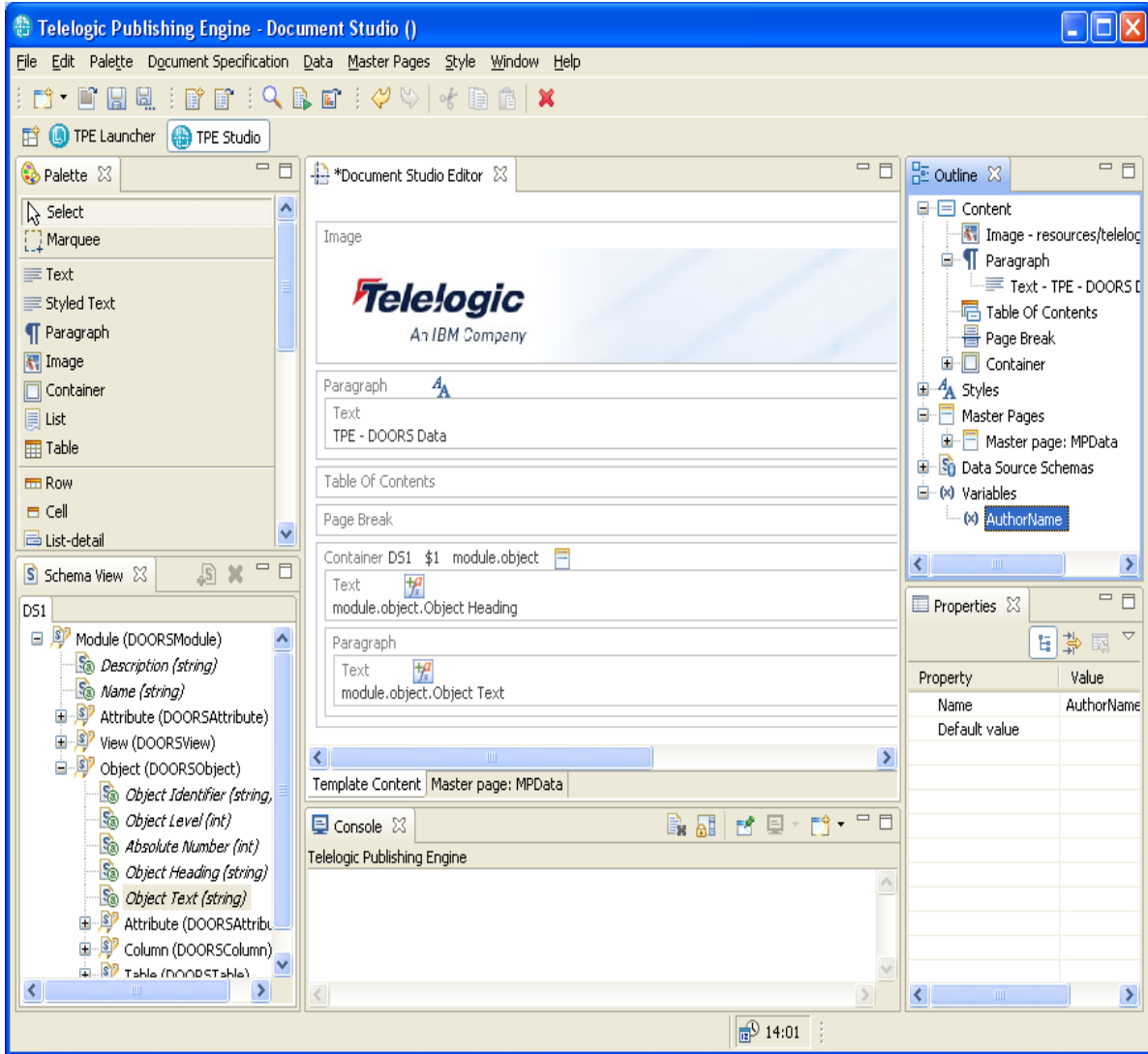


Figure 140

Use the variable in the template after the title

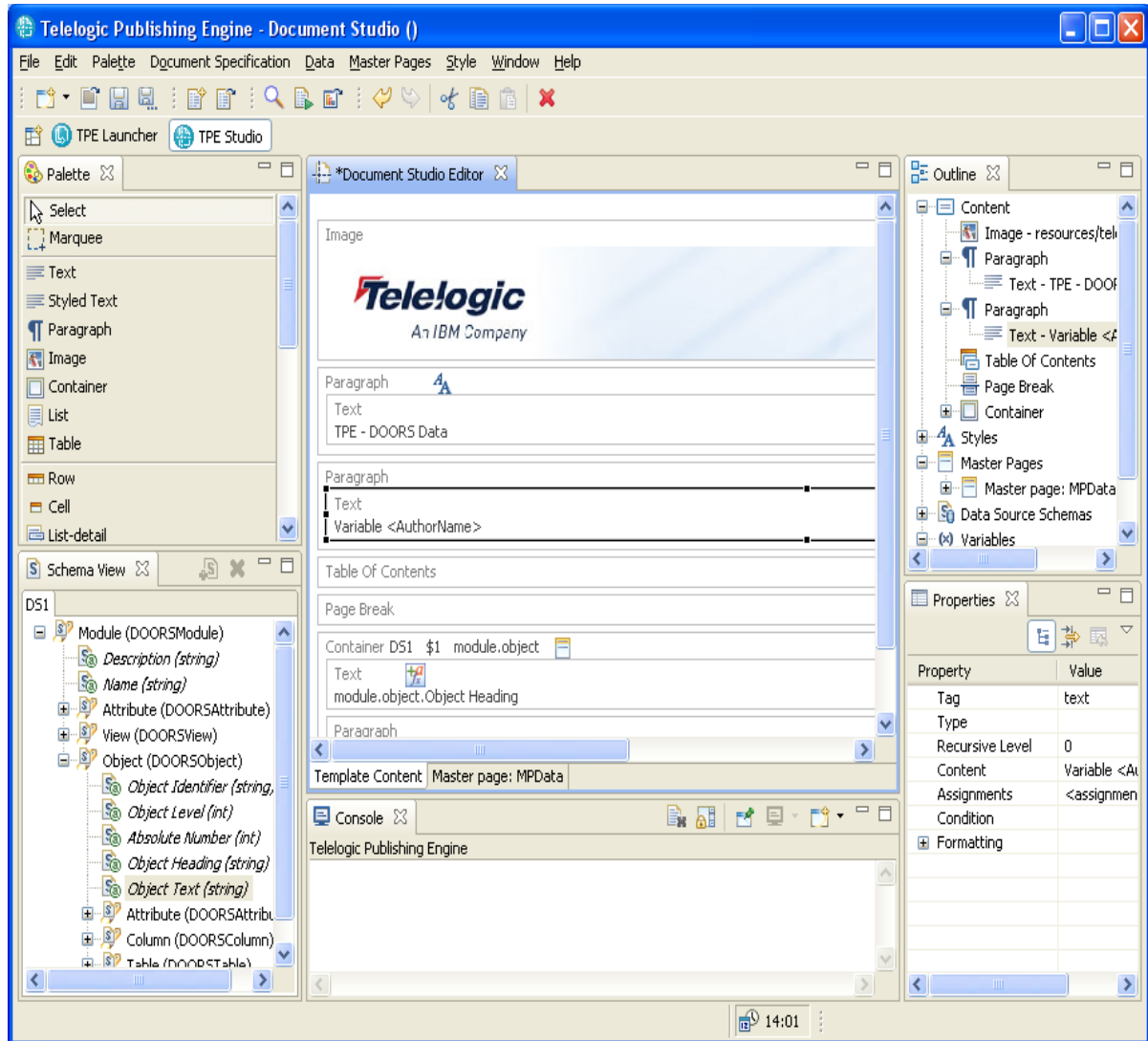


Figure 141

Generate the document

Before generating a document, make sure you have saved the template.

Open the TPE Launcher Perspective

Use the button located in the top left part of the screen, right below the toolbar.

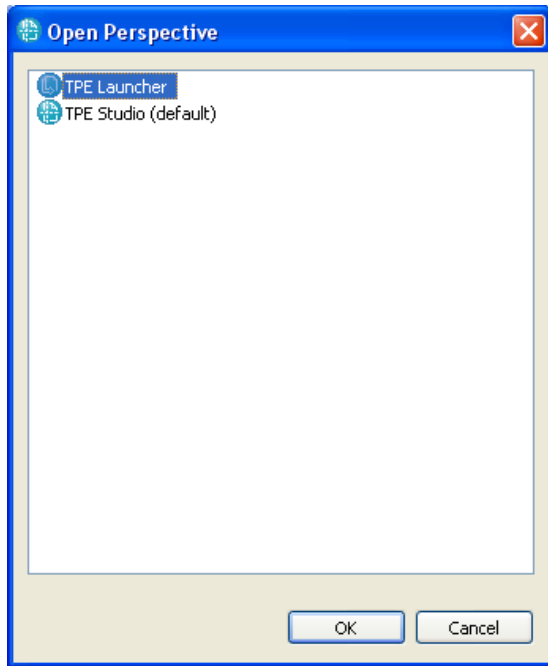


Figure 142

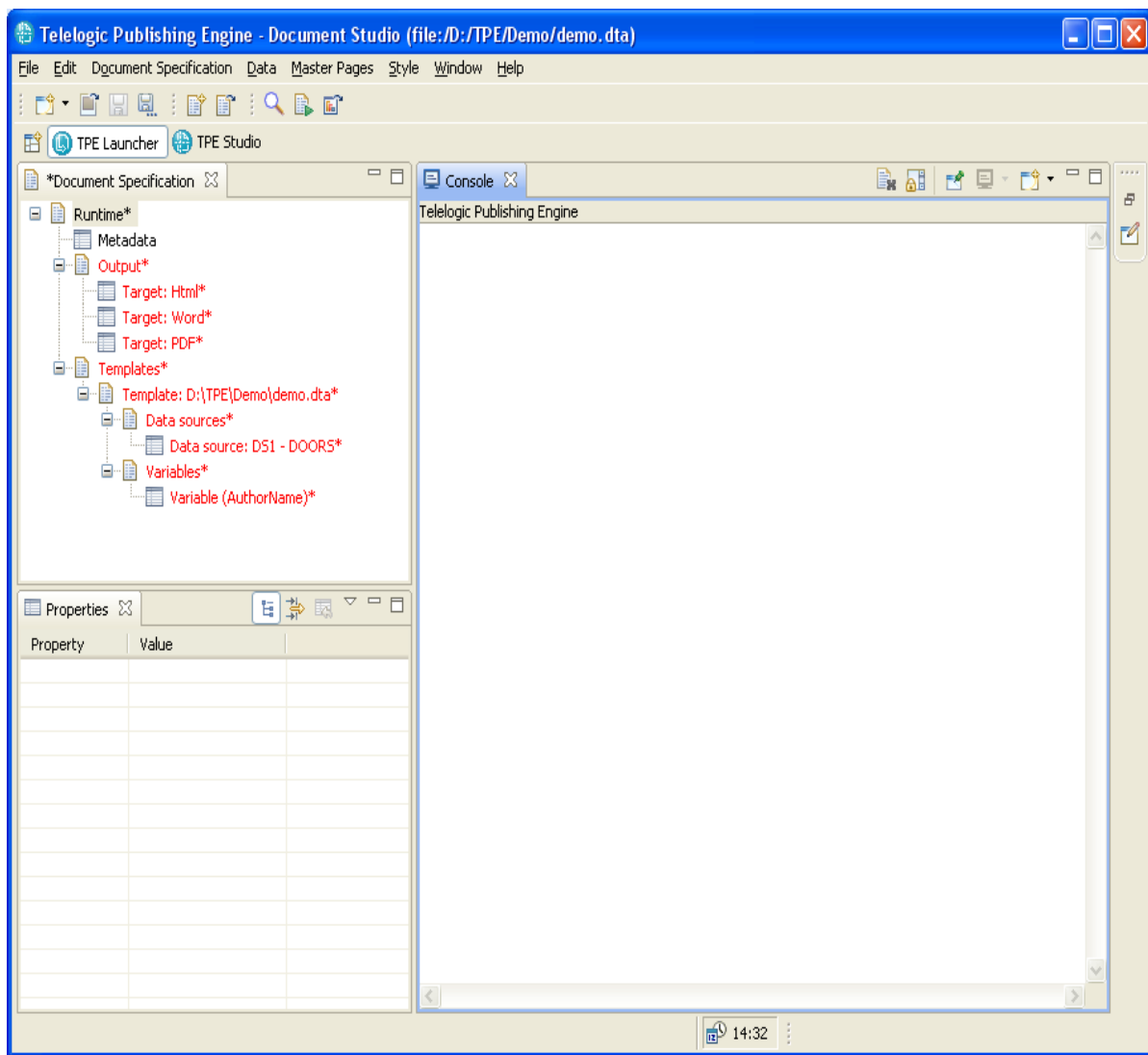


Figure 143

NOTE When the launcher is embedded in TPE Document Studio and the “keep document specification synchronized” option is checked in the preference page, the document specification is synchronized with the edited template at every load/close/save operation. Otherwise you need to manually synchronize the document specification to ensure all the data source schemas from the template have a data source equivalent in the document specification.

Configure the DOORS Data source

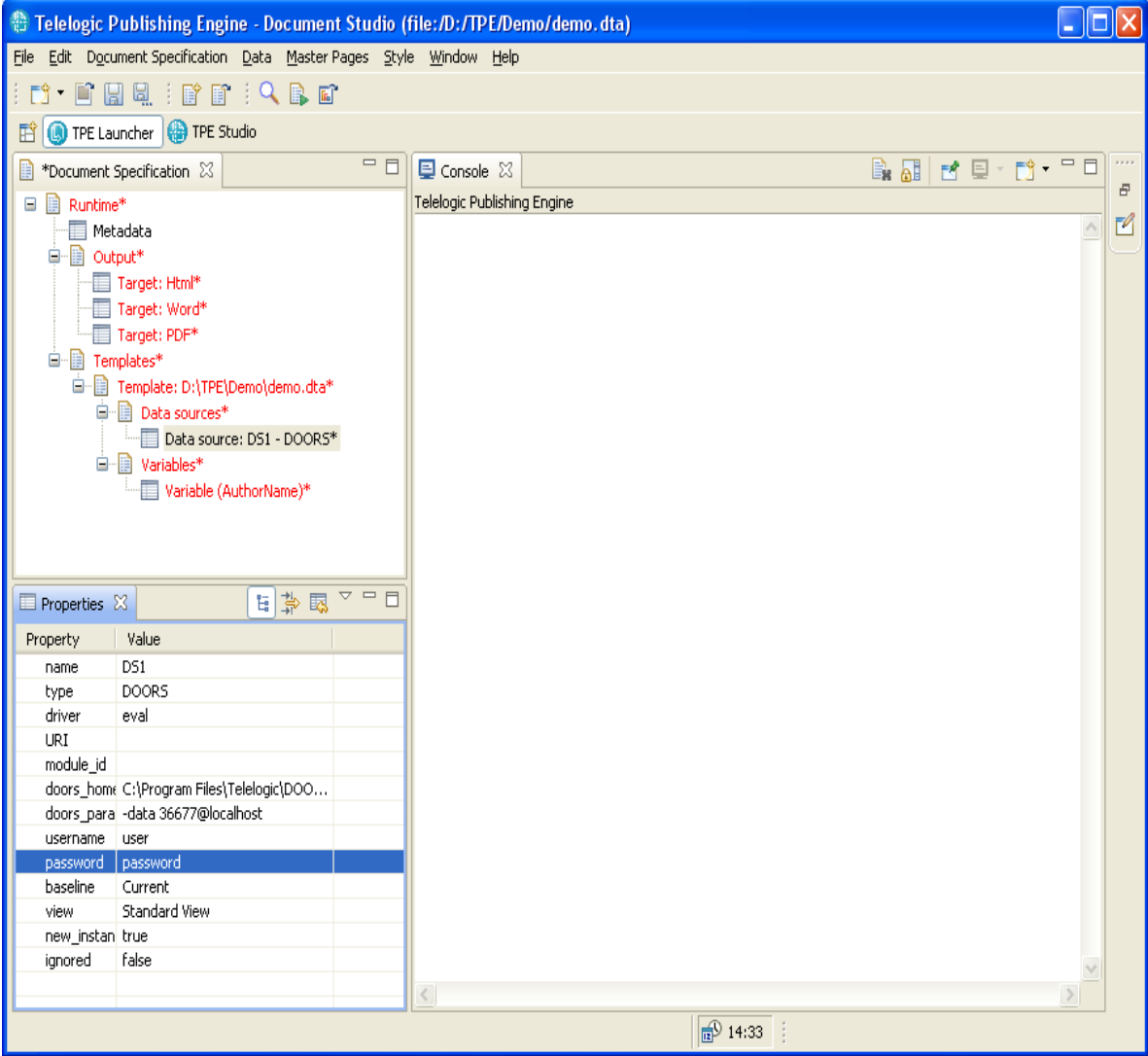


Figure 144

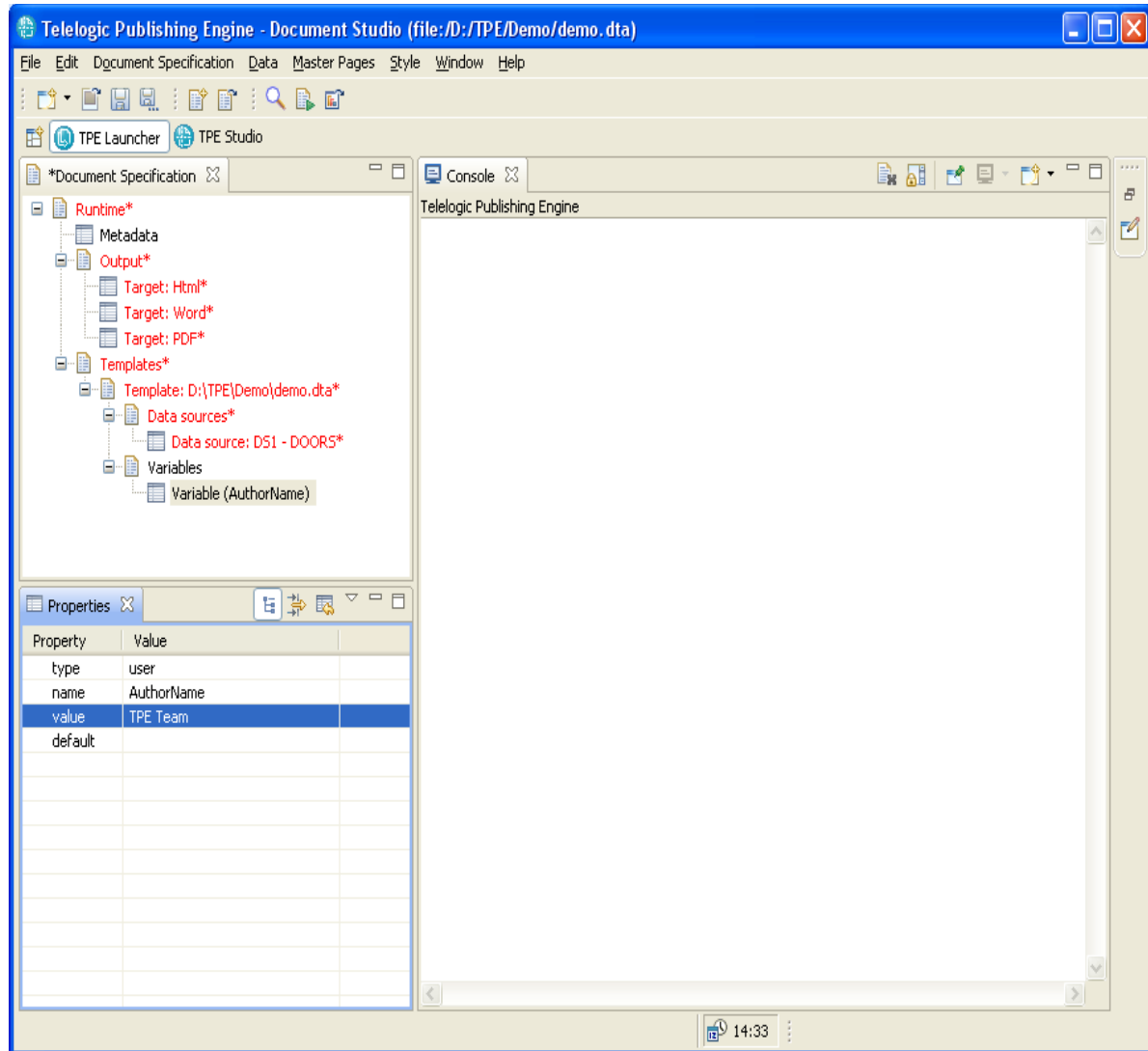
Set a value for the AuthorName variable

Figure 145

Generate the document

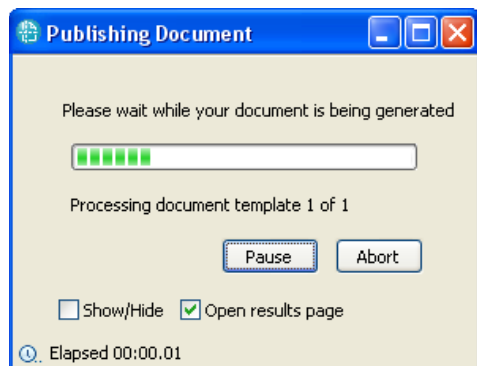


Figure 146

View the results

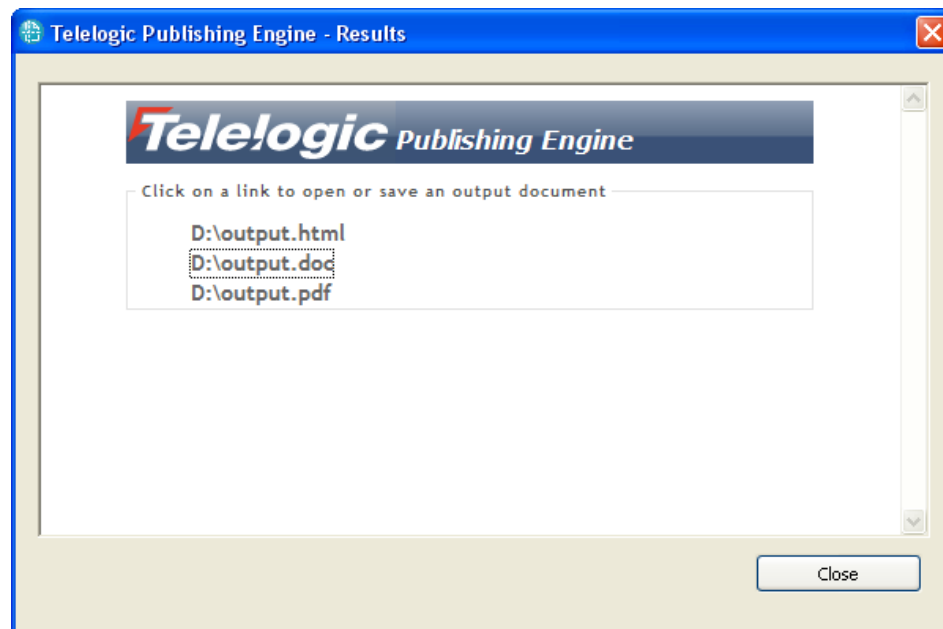


Figure 147

NOTE When you open the generated Word document for the first time you need to manually update the table of contents. You can use the provided “updateTOCs” macro to automate this task.

A template for Tau data

This chapter shows how to build a simple TPE Document Template. The output document will contain the list of top level packages, and for each package the list of classes and the diagrams contained in that package. For each class a table will be generated listing all the class's attribute names and types.

- Package
 - Diagrams
 - Diagram
 - Diagram
 - ...
 - Classes
 - Class
 - Attribute table
 - Class
 - Attribute table
 - ...

Start Document Studio

Add a Tau Data source Schema

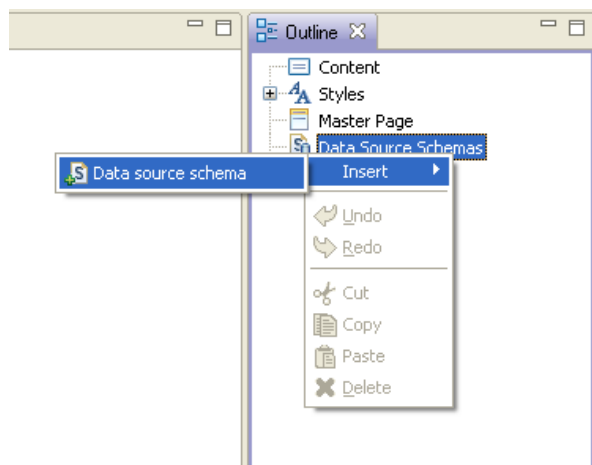


Figure 148 Select "Insert schema" from the outline's context menu

The "Add Datasource schema" wizard is started.

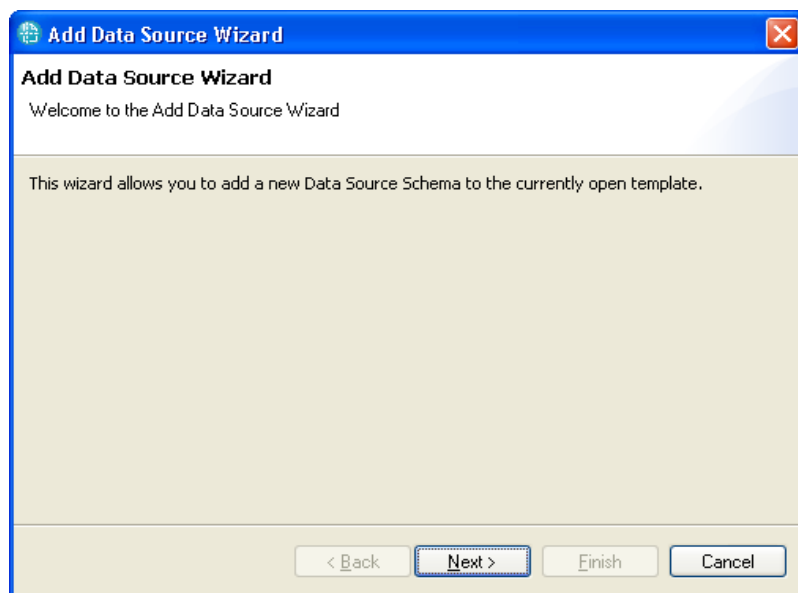


Figure 149

Select the ttdMetamodel.xsd schema provided with TPE. Make sure you set the data source type to Tau. You can use any value for the Data Source ID of the schema.

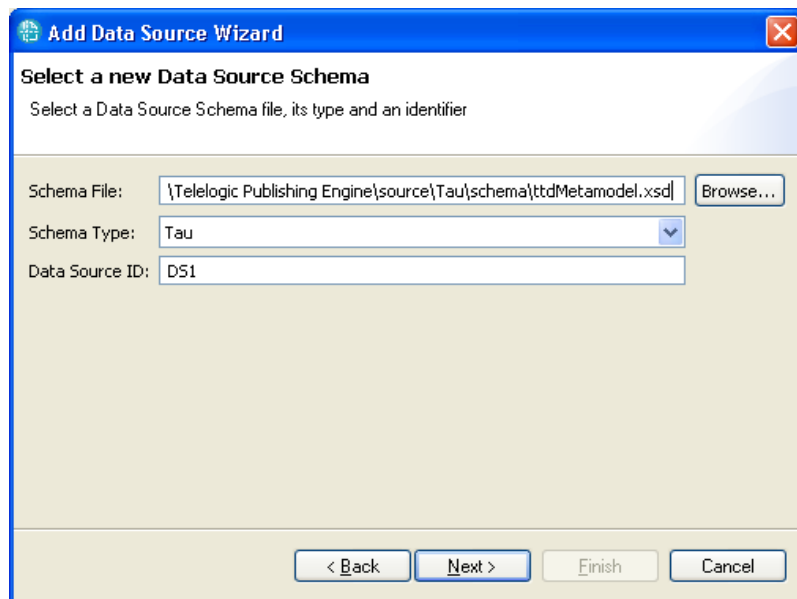


Figure 150

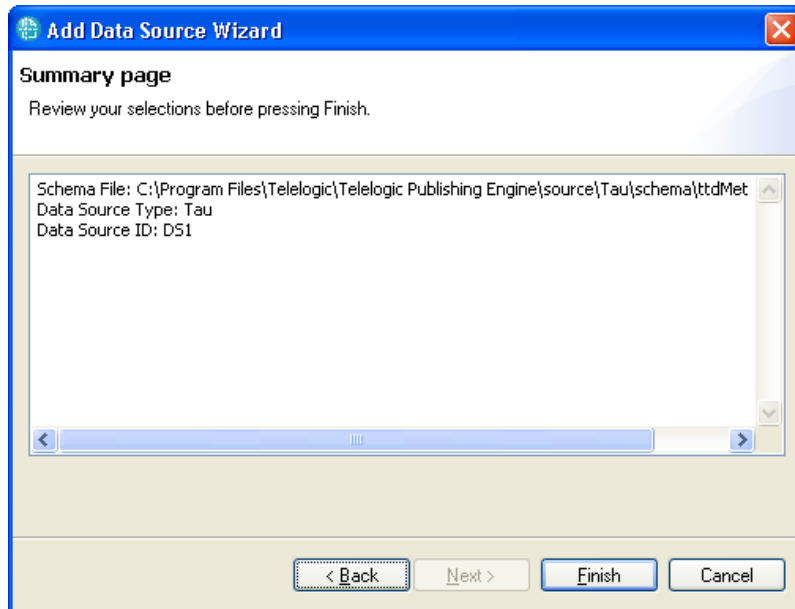


Figure 151

Once the wizard is complete, the new data source schema is added to the template.

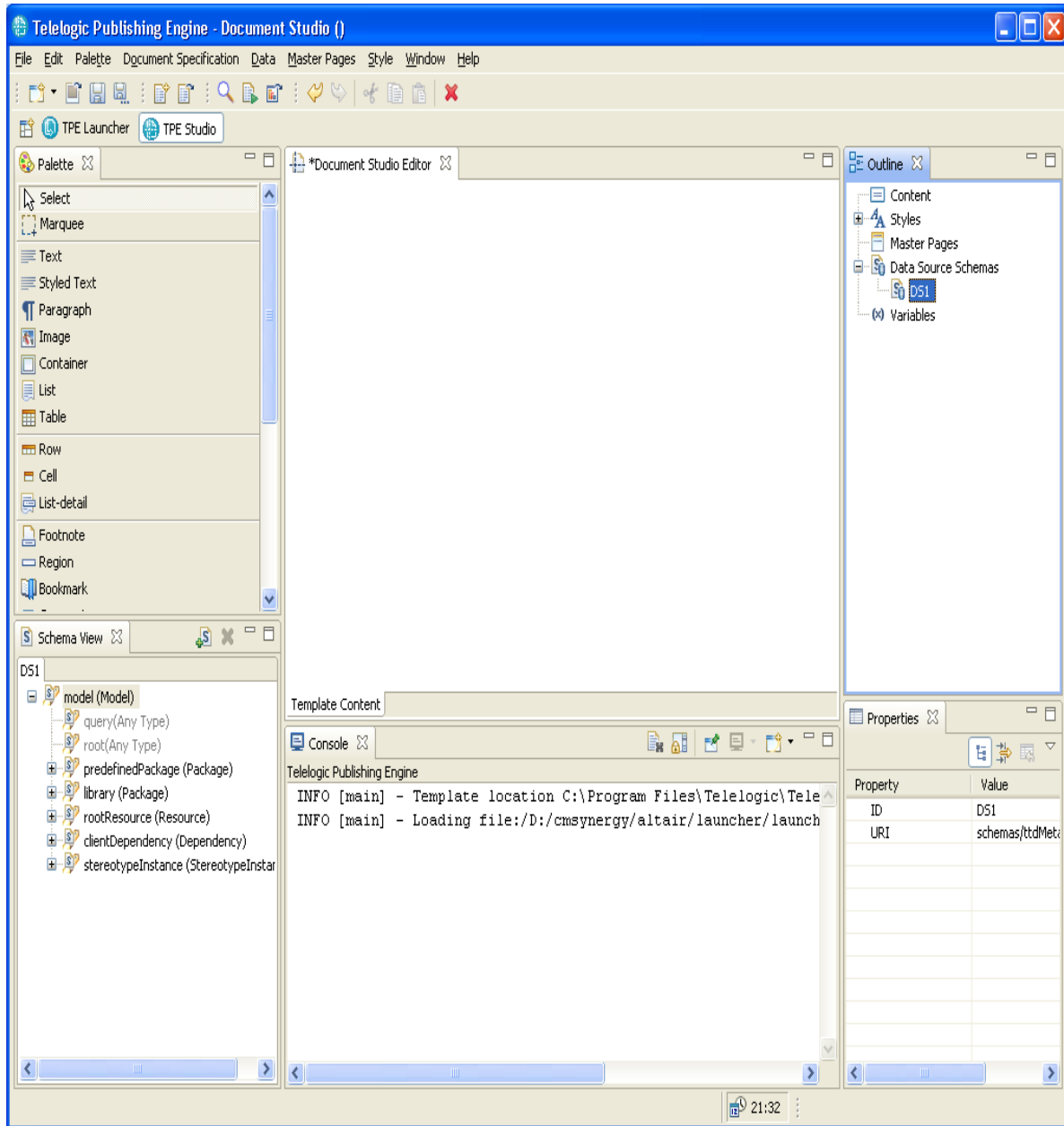


Figure 152 Tau data source schema added to the template

Save the document

Define the template content

Define a query for the top level packages. In order to do that add the “Package” cast under the *model.root* element by clicking the “Cast to type” button.

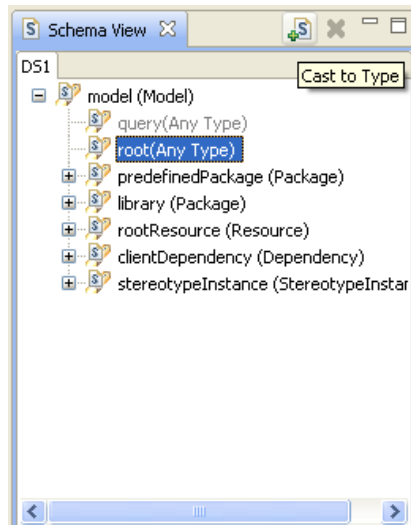


Figure 153

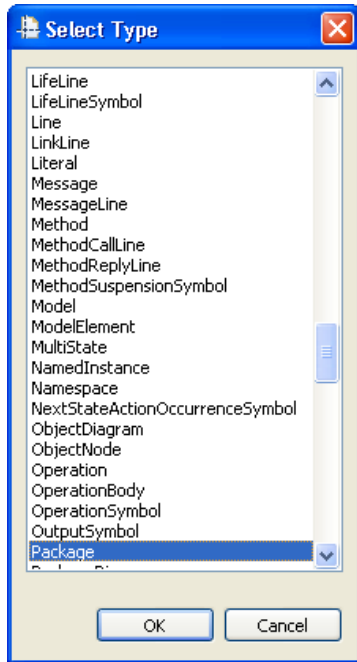


Figure 154

Create a container element in the template and drag the newly added Package element to it.

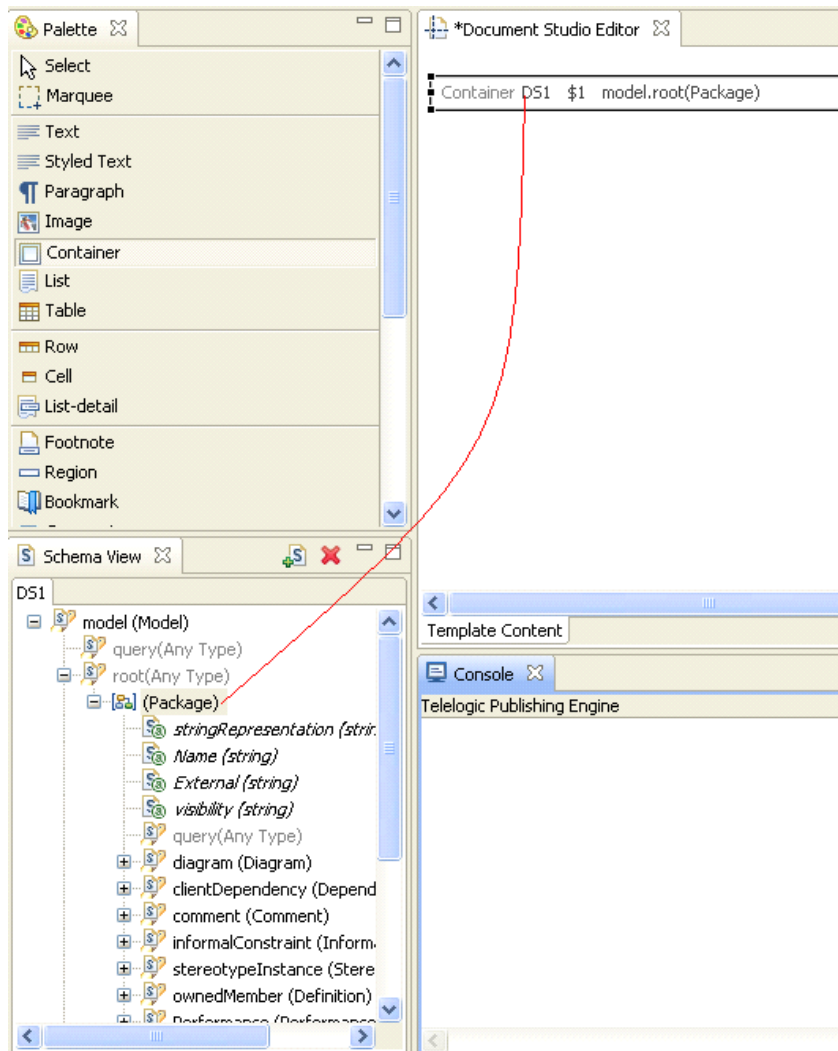


Figure 155

Create a paragraph in the container element and add the “Name” attribute.

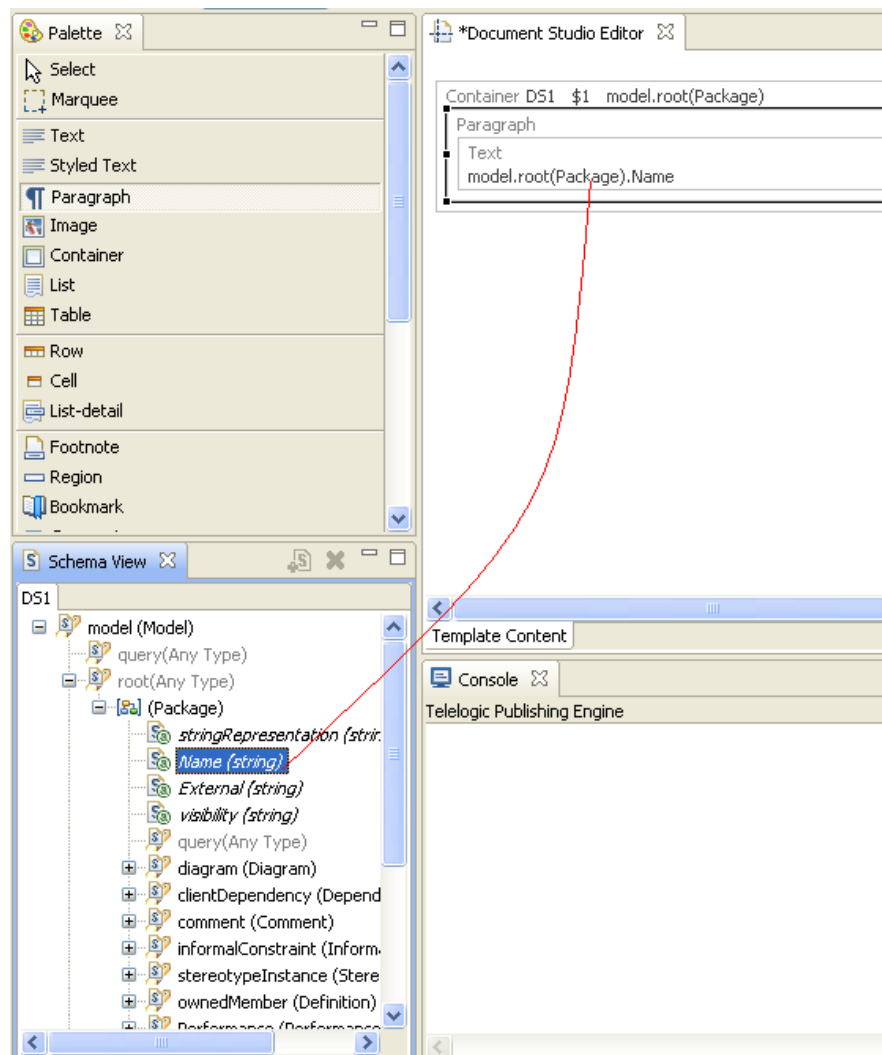


Figure 156

NOTE If you would generate the document using the template in its current form, the output will contain the name of all the top level packages in the model.

- Package
- Package
- Package

Save the document.

To continue, add the static text heading Diagrams for the list of diagrams by adding a Paragraph into the Container, adding Text into the paragraph, then double-click the text and set the value to Diagrams.

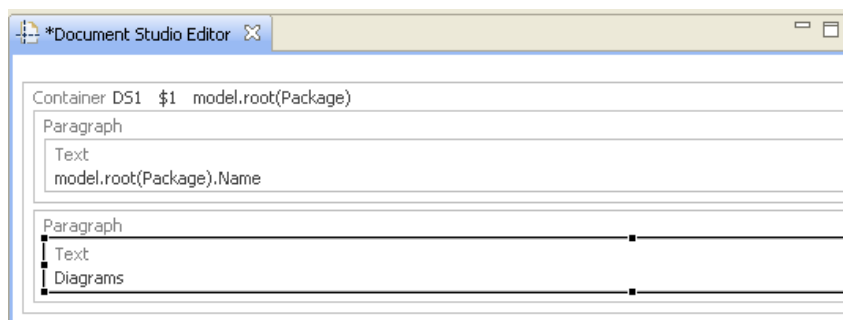


Figure 157

Add the package diagram element to the template.

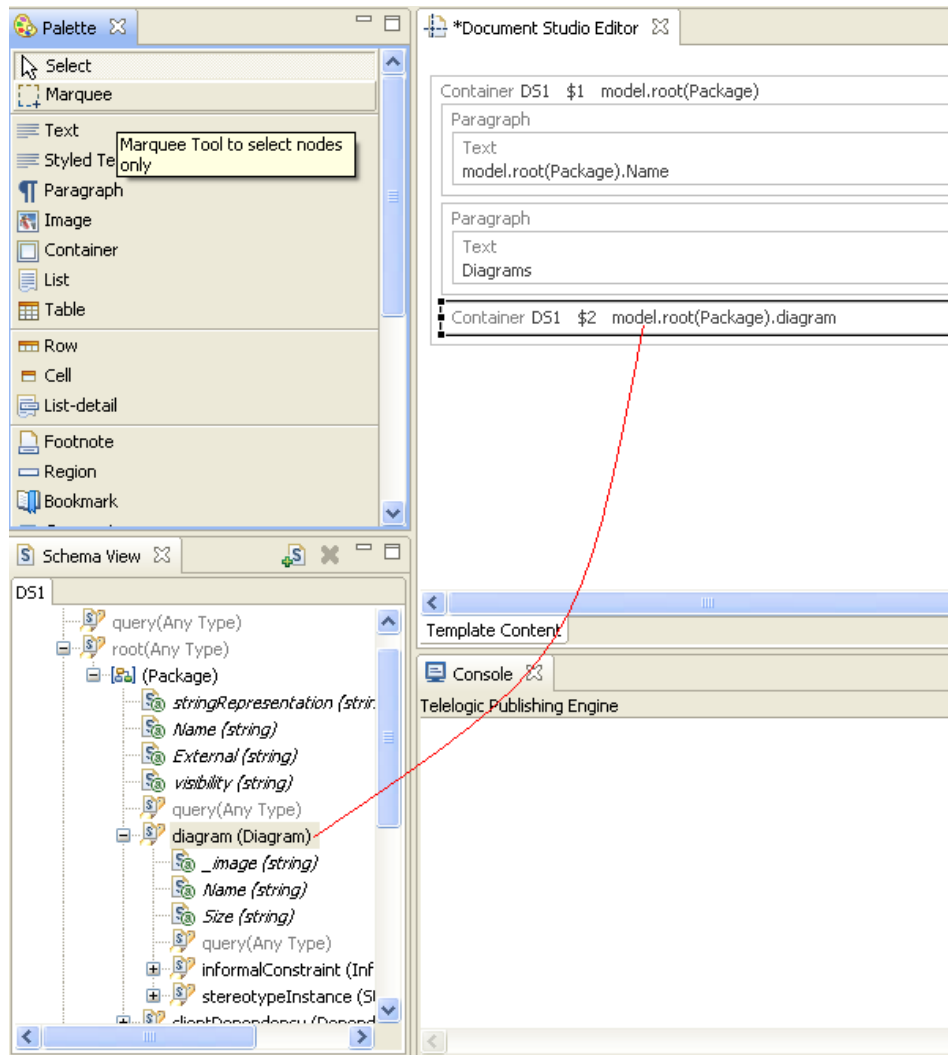


Figure 158

NOTE If you are interested in a specific diagram type you can add a cast to that diagram type below the diagram(Diagram) node and use it instead of using the generic “Diagram” type. Add an image element in the container. Set its content to be the “_image” attribute of *model.root(Package).diagram(Diagram)*.

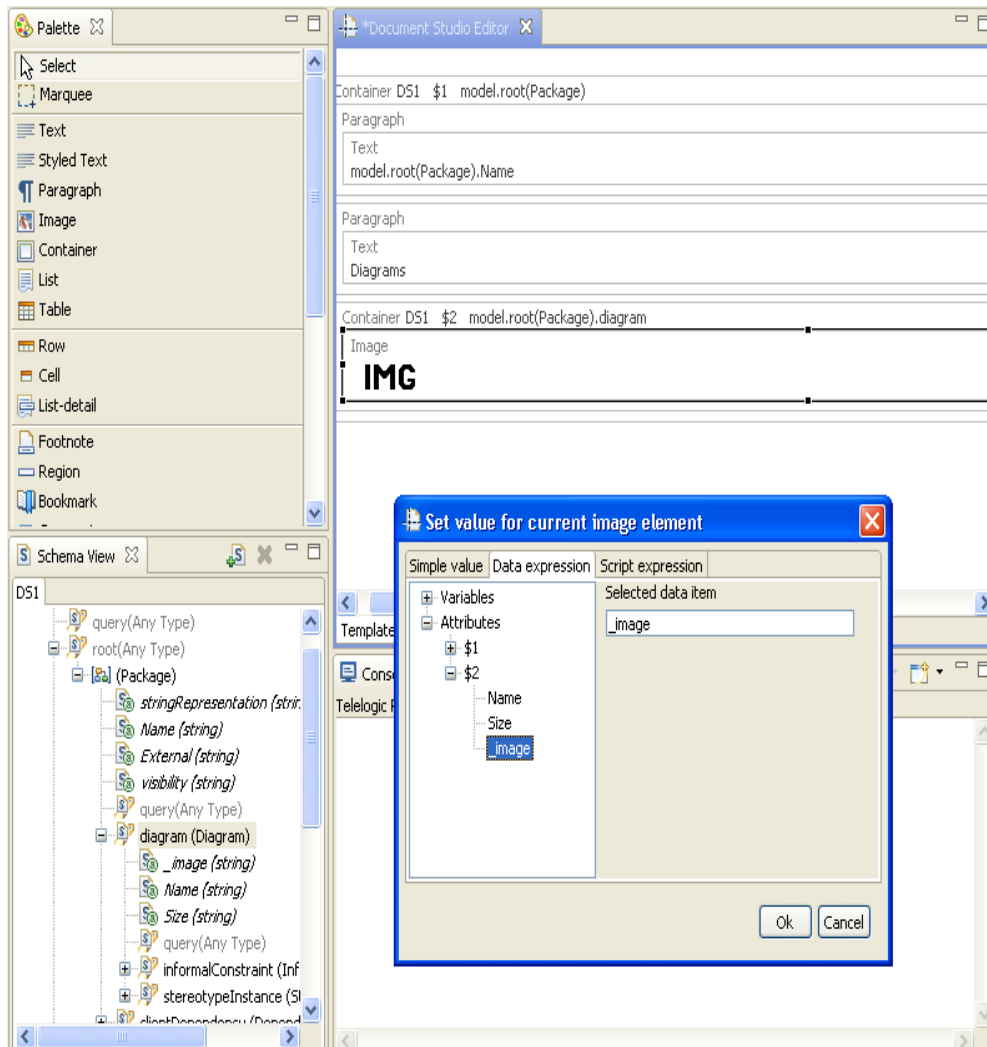


Figure 159

Add the name of the diagram below its image. Additionally you can add a figure caption field so you can build a Table of Figures.



Figure 160

NOTE If you were to generate the document using the template in its current form, the output will contain the name of all the top level packages in the model, and all the diagrams in each package:

- Package
 - Diagrams
 - Diagram (image and name)
 - Diagram (image and name)
 - ...
- Package
 - Diagrams
 - Diagram (image and name)
 - Diagram (image and name)
 - ...
- ...

Save the document.

Add the static text for the class list.

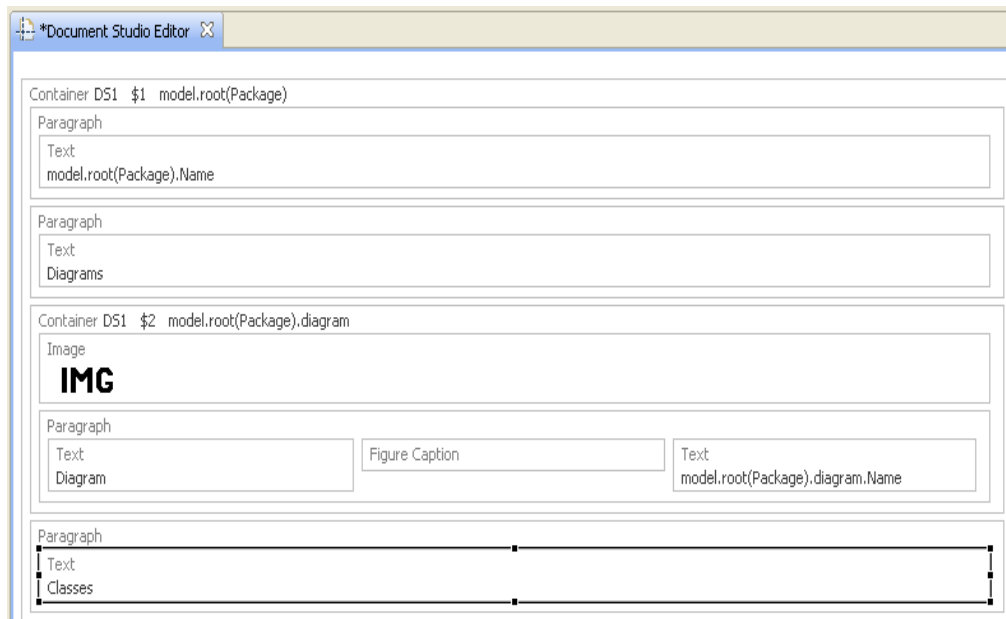


Figure 161

Add the “Class” type cast for the “ownedMember” element of the package and use it to create the query.

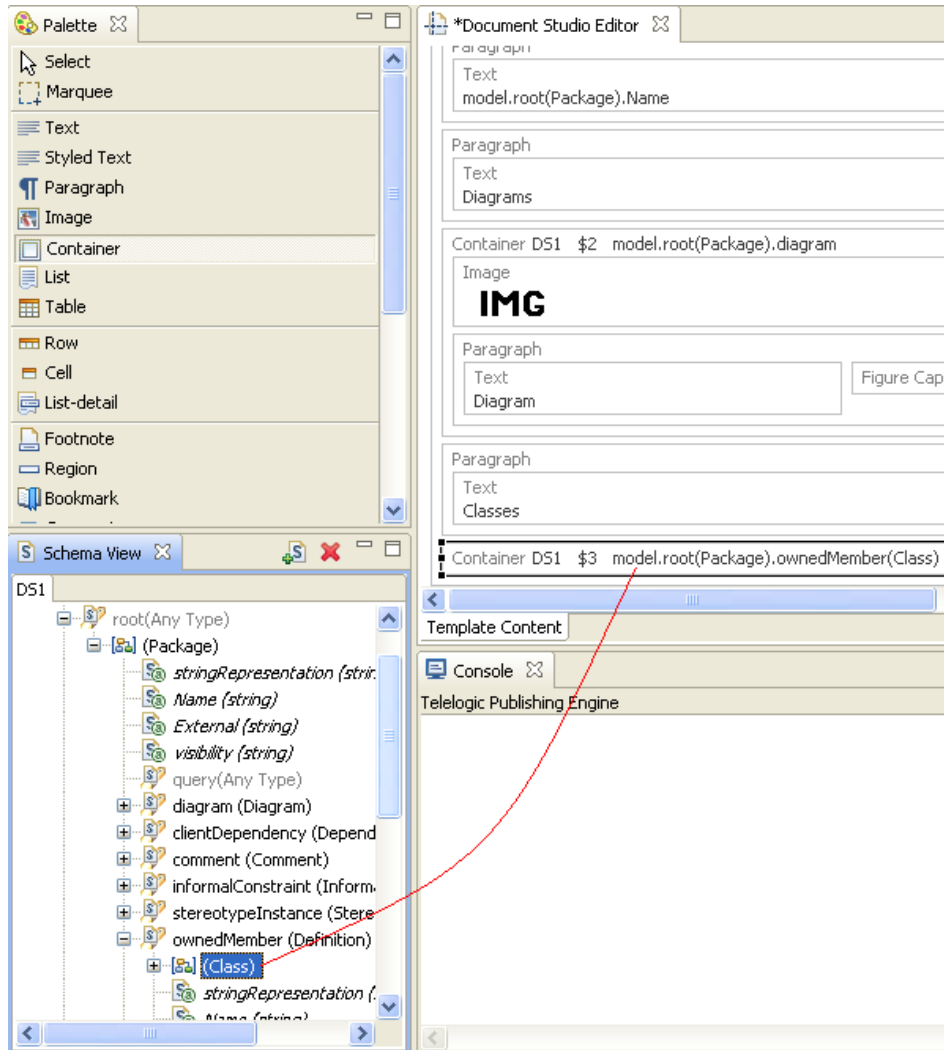


Figure 162

Add the name of the package in its own paragraph:

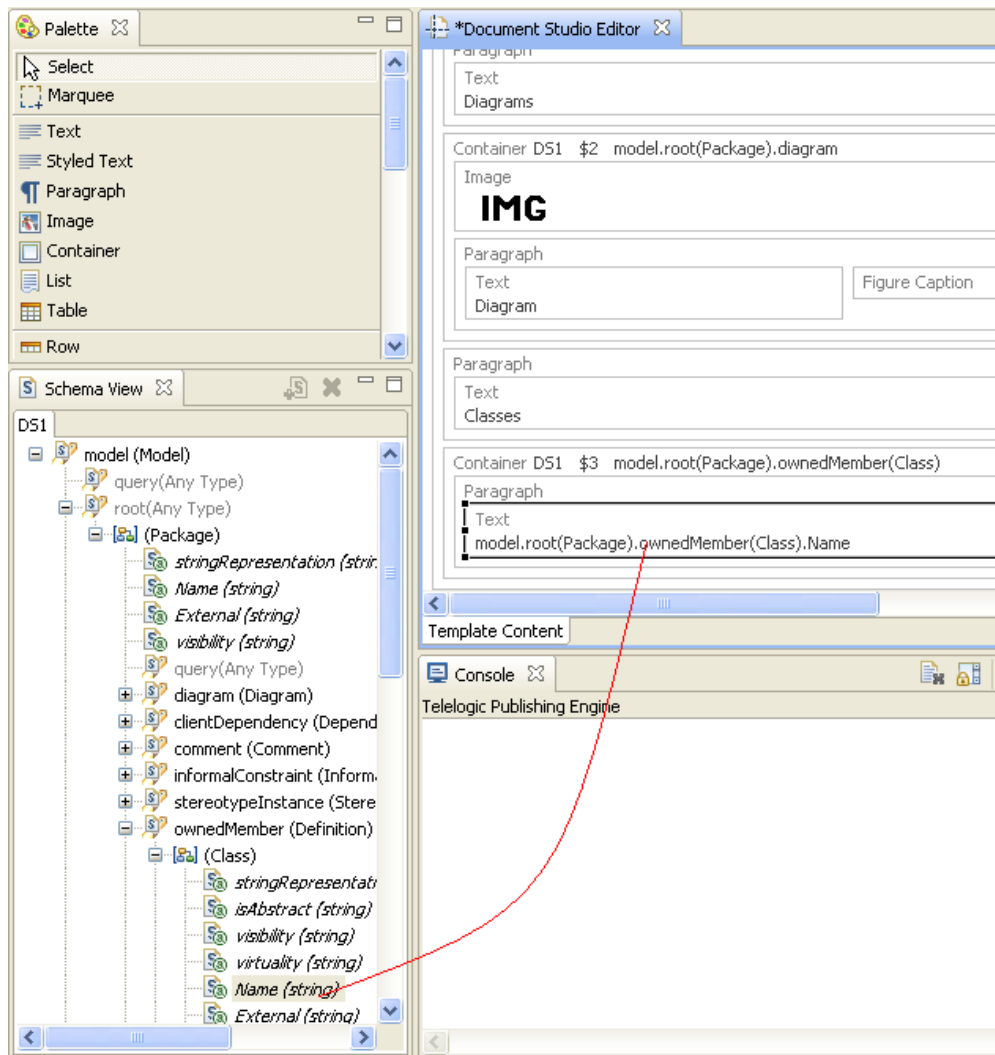


Figure 163

NOTE If you would generate the document using the template in its current form, the output will contain the name of all the top level packages in the model, and all the diagrams in each package:

- Package
 - Diagrams
 - Diagram (image and name)

- Diagram (image and name)
- ...
- Classes
 - Class 1
 - Class 2
 - Class 3
 - ...
- Package
 - Diagrams
 - Diagram (image and name)
 - Diagram (image and name)
 - ...
 - Classes
 - Class 1
 - Class 2
 - Class 3
 - ...
- ...

Save the document.

Create the table with two rows and two cells that will host the class attributes:

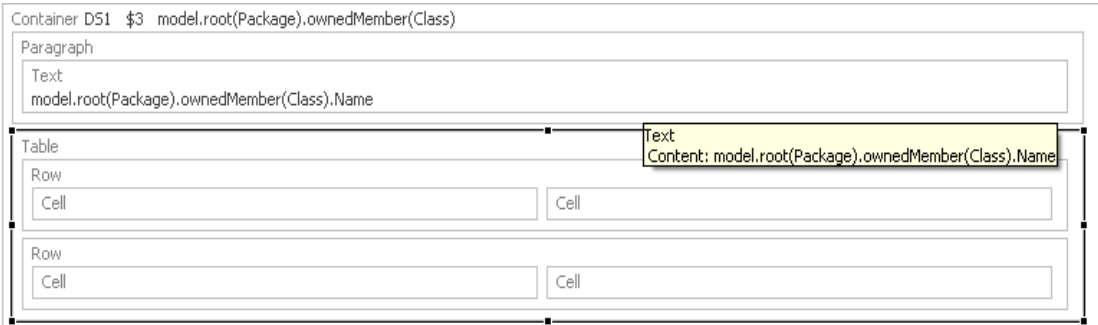


Figure 164

Add the “Attribute” type cast to the “ownedMember” element of Class and use it as a query on the second row of the table. At the same time apply the borderedCell style to all the table’s cells.

Container D51 \$3 model.root(Package).ownedMember(Class)	
Paragraph	
Text model.root(Package).ownedMember(Class).Name	
Table	
Row	
Cell 4A Text Name	Cell 4A Text Type
Row D51 \$4 model.root(Package).ownedMember(Class).ownedMember(Attribute)	
Cell 4A	Cell 4A

Figure 165

Use the attribute name in the first cell:

Container D51 \$3 model.root(Package).ownedMember(Class).ownedMember(Attribute)	
Paragraph	
Text model.root(Package).ownedMember(Class).Name	
Table	
Row	
Cell 4A Text Name	Cell 4A Text Type
Row D51 \$4 model.root(Package).ownedMember(Class).ownedMember(Attribute)	
Cell 4A Text model.root(Package).ownedMember(Class).ownedMember(Attribute)	Cell 4A

Figure 166

Drag the attribute's type element as q query on the 2nd cell of the 2nd row and drag the name attribute in the cell.

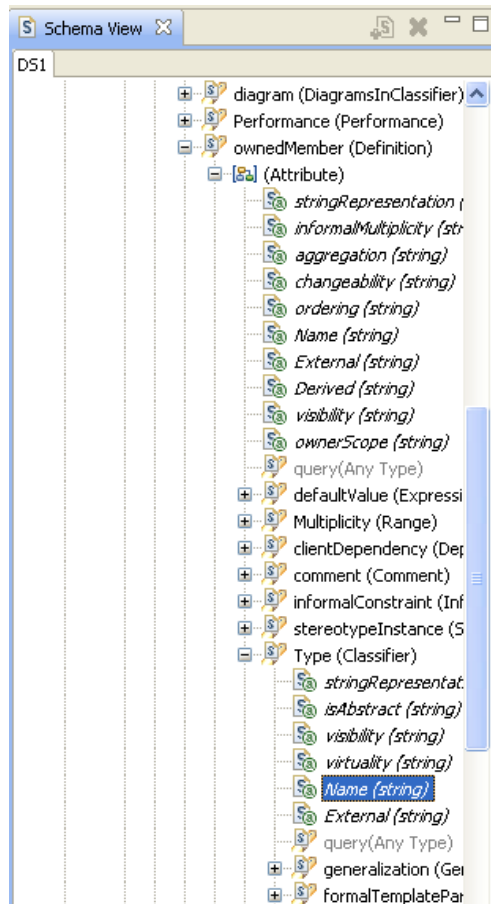


Figure 167

Container D51 \$3 model.root(Package).ownedMember(Class)	
Paragraph	
Text model.root(Package).ownedMember(Class).Name	
Table	
Row	
Cell ^A _A	Cell ^A _A
Text Name	Text Type
Row D51 \$4 model.root(Package).ownedMember(Class).ownedMember(Attribute)	
Cell ^A _A	Cell D51 \$5 model.root(Package).ownedMember(Class).ownedMember
Text model.root(Package).ownedMember(Class).ownedMember(Attribute)	Text model.root(Package).ownedMember(Class).ownedMember(Attribute)

Figure 168

NOTE Setting a query on a cell/row will generate as many cells/row as the query returns. But since an attribute has only 1 type, the query will return a single element.

Save the document.

Generate the document

Open the launcher perspective

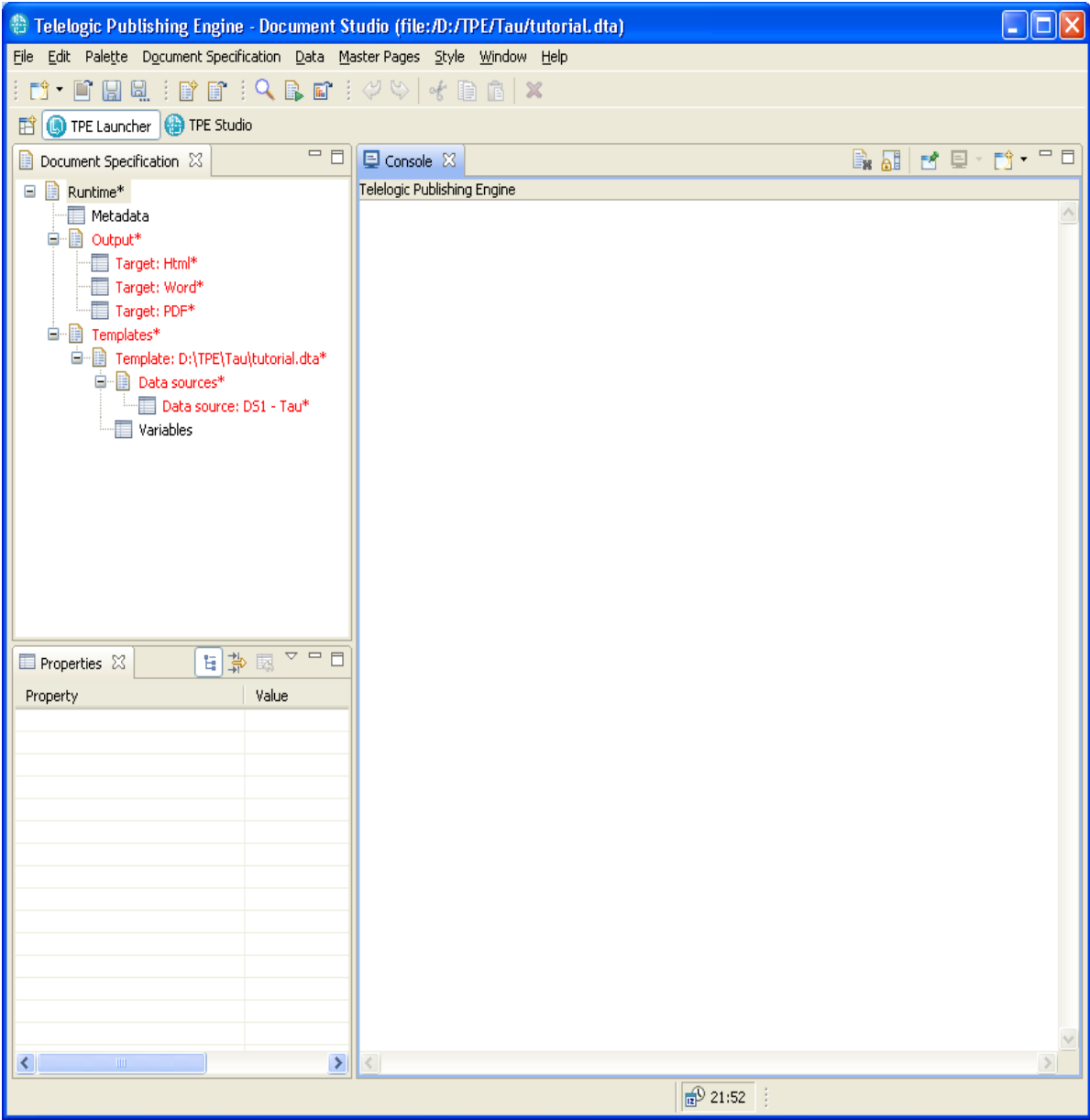


Figure 169

Select the Tau project (.ttp file) you wish to use as concrete data source for the DS1 schema used in the template. You can use one of the examples provided with Tau, which can be found in the examples directory below the Tau installation directory (on Windows the default location of the examples is C:\Program Files\Telelogic\Tau_4.2\examples):

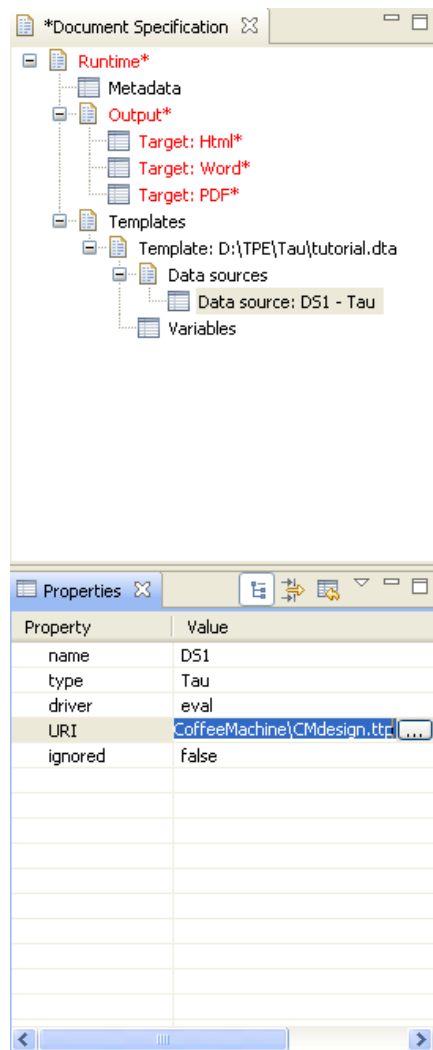


Figure 170

NOTE Optionally, you can configure each output target.

Start the document generation process. TPE will ask you to save the changes to the document template, if any unsaved changes exist.

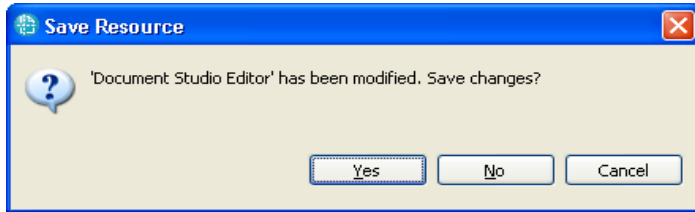


Figure 171

Unless you cancel the save the document generation process will start

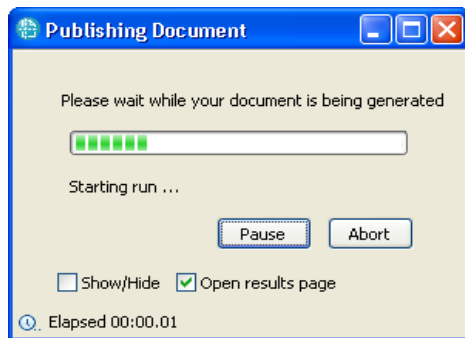


Figure 172

Once finished a window will be displayed allowing you to view/save the results

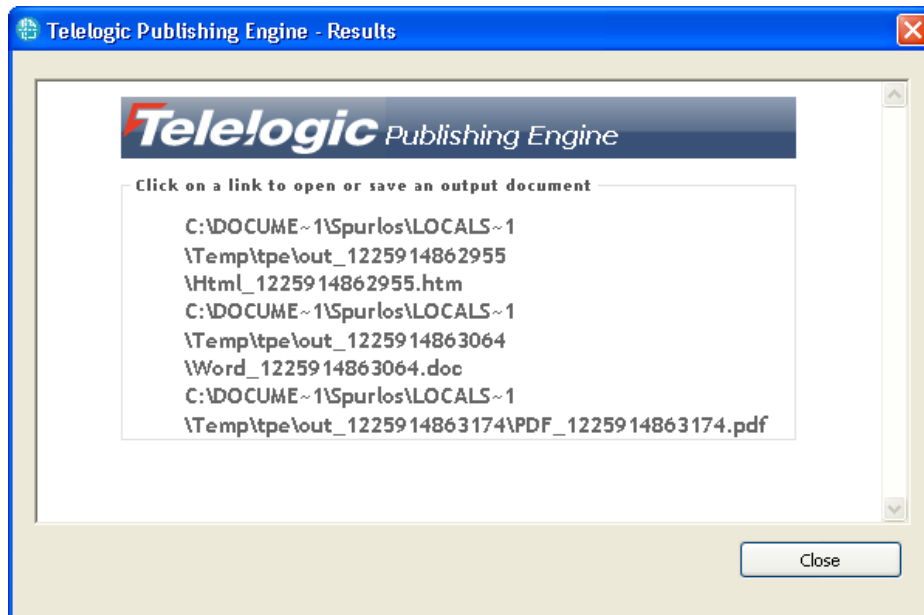


Figure 173

Appendix: Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send written license inquiries to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send written inquiries to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions. Therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Intellectual Property Dept. for Rational Software
IBM Corporation
1 Rogers Street
Cambridge, Massachusetts 02142
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these

names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, Telelogic, and Telelogic DOORS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.html.

Microsoft, Windows, Windows 2003, Windows XP, Windows Vista and/or other Microsoft products referenced herein are either trademarks or registered trademarks of Microsoft Corporation.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

