PL/PDF Template
User Guide
V4.0.0

Reporting for your Oracle database

As of version 1.4.0, PL/PDF enables you to use existing PDFs as templates or background layers to your PDF document generated by PL/PDF. When a template is used, the template PDF is drawn first onto the page, followed by the objects generated by the PL/PDF code.

By using templates:
- A background layer to a page can be inserted similarly to the MS Word
- template (dot file).
- A standard form can be filled out. A form created by another tool can be used as a basis for a PL/PDF document. The blank fields then can be filled from the database using PL/PDF.

A template in PL/PDF is always one page only, but in one document generated by PL/PDF more than one template may be used. Also, a template may be used more than once.

The template must be a PDF format document. If you wish to use a document in another format as a template then it needs to be converted to PDF. Documents can be converted to PDF by Adobe Acrobat (not Reader) or NitroPDF or by a printer driver (e.g. pdf995).

PDF features not supported:
- Password protected PDFs and PDFs with security policies. The security features of a PDF document can be checked in Acrobat Reader: File - Document Properties - Security.
- PDF features introduced after version 1.3 (Acrobat 4.x). PL/PDF generates documents using version 1.3, therefore if the template has features that were introduced after this version, it cannot reproduce them even though it will not give an error. The version can be checked in Acrobat Reader: File - Document Properties - General.
- Certain types of compression. The content of a PDF page is made up from several objects that may have different types of compression to achieve a smaller PDF file in the end. PL/PDF can only handle gzip compression (FlatDecode), with certain exceptions. More detailed information is available in the error messages section.
- Interactive features in the templates will not cause an error but will not work in the resulting PDF.
- If the template PDF contains rotation then the resulting PDF generated by PL/PDF will not be rotated.

In PL/PDF templates are implemented using the plpdf_type.tr_tpl_data data structure. Templates are created, stored and used through this data structure.

It is recommended that you create and store all the templates you may wish to use in the future as this process takes a long time. Templates are stored in the PLPDF_TEMPLATE% tables. In the program where the resulting PDF is created the template can be retrieved and used quickly. You can also create and use a template without storing (and retrieving) it but the performance will not be as good.

How to create and save a template:

A page from an existing PDF can be converted into a template by using the plpdf_parser.GetTemplate function.

GetTemplate

Type: function
  - − p_blob blob: The PDF is passed as a BLOB type parameter that can originate from a database table or an external file (BFILE)
  - − p_page_id number: the physical page number of the page in the PDF that is to be converted to a template

Return: plpdf_type.tr_tpl_data data structure that can either be used immediately or can be stored for later usage. To store the template use:

SaveTemplate

Type: procedure
  - − p_id number: ID, sequence number that can be used later to retrieve
  - − the template (Primary Key in the PLPDF_TEMPLATE table)
  - − p_tpl plpdf_type.tr_tpl_data: template data structure
  - − p_descr varchar2 default null: optional description
  - − p_commit boolean default true: should a COMMIT be invoked after the INSERT statements

Return: -

Repeat the GetTemplate command in conjunction with the SaveTemplate command for all your potential templates. These commands can be done independently of the program where the template will be used.

How to retrieve and use the template:

In the program where the templates are to be used, first load the templates into the plpdf_type.tr_tpl_data data structure. To retrieve the template use:

LoadTemplate

Type: function
  - − p_id number: template ID (Primary Key in the PLPDF_TEMPLATE table)

Return: plpdf_type.tr_tpl_data: template data structure Once the plpdf_type.tr_tpl_data data structure is available the template can be included in the PDF file using:

plpdf.InsTemplate

Type: function
  - − p_tpl plpdf_type.tr_tpl_data: ID, sequence number that can be used later to retrieve the template (Primary Key in the PLPDF_TEMPLATE table)

Return: number ID that the UseTemplate command will require as an input.

It is practical to call this procedure right after calling plpdf.init and LoadTemplate.

The data structure for plpdf.InsTemplate can also be produced directly using the plpdf_parser.GetTemplate function. However, to achieve a better performance, it is recommended that template(s) are not created every time directly, but rather that template(s) should be created ahead of time and stored in the database.

To assign a templates included by InsTemplate to a specific page use:

plpdf.UseTemplate

Type: procedure
- p_tplidx number: ID, returned by plpdf.InsTemplate
- p_fittopage boolean default true: stretch the template to the page size or not

Return: - If a template is used on more than one page, then InsTemplate only needs to be called once and then UseTemplate can be called multiple times.

```
**********************************************************************
Simple example to get and save template:
declare
v_pdf blob;
v_tpl plpdf_type.tr_tpl_data;
begin
...
-- Get the blob from somewhere
...
-- Get Template
v_tpl := plpdf_parser.GetTemplate(l_pdf,1);
plpdf_parser.SaveTemplate(10,v_tpl);
end;
Simple example to retrieve and use template:
declare
v_tpl plpdf_type.tr_tpl_data;
l_tpl_id number;
begin
-- Initialize PDF
plpdf.init;
-- Insert Template
l_tpl_id := plpdf.InsTemplate(v_tpl);
...
plpdf.NewPage;
-- Use Template
plpdf.useTemplate(l_tpl_id);


**********************************************************************
```

Error messages and related solutions:

· ERR-203: PARSER error: &1-&2-&3-&4.: general analysis error: either the PDF is corrupt, incomplete or includes features that are not supported by PL/PDF. **Solution**: reduce the size of the PDF, so that it only includes the page that is needed for the template (see pdftk cat later)

- ERR-215: Invalid/Unsupported Template: &1.: similar to ERR-203
- ERR-216: Unsupported Compress in Template: &1. **Solution**: uncompress the PDF (see pdftk uncompress later)
- ERR-217: Multiple Filter is not supported in Template. **Solution**: uncompress the PDF (see pdftk uncompress later)
- ERR-218: Encryption unsupported in Template: as discussed earlier password protected and PDFs with security policies are not supported. **Solution**: If the owner or user password is known, then the password protection or security policy can be removed.

In all other cases when there is an error, please first try to UNCOMPRESS and CAT with the pdftk tool before contacting us for support.

PDFTK

The pdftk (www.accesspdf.com/pdftk/) is a free command line tool with which an existing PDF can be modified. The tool and its documentation can be downloaded from the website above. As far PL/PDF templates are concerned two commands are very useful:

- UNCOMPRESS: Remove PDF page stream compression by applying the uncompress filter. Example: pdftk mydoc.pdf output mydoc.clear.pdf uncompress.
- CAT: Catenates pages from input PDFs to create a new PDF. Page order in the new PDF is specified by the order of the given page ranges. Example: pdftk A=one.pdf cat A1 output one_1.pdf

A GUI is also available for the pdftk tool at http://www.paehl.de/pdf/?GUI_for_PDFTK.