Open Office API-Viewer

Nicole Scholz

Vienna University of Economics and Business Administration

Reg. No. 0251817

E-Mail: h0251817@wu-wien.ac.at

February, 2009

Diploma Thesis

Institute for Management Information Systems

Prof. Dr. Rony G. Flatscher

Ich versichere,

dass ich die Diplomarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

dass ich die Ausarbeitung zu dem obigen Thema bisher weder im In- noch im Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

dass diese Arbeit mit der vom Betreuer beurteilten Arbeit übereinstimmt.

Datum:		Unterschrift:	

Table of Contents

Figures

Code

1 Introduction

This diploma thesis is about a program called API-Viewer. The aim of the API-Viewer is to offer the possibility to get information about the Open Office UNO IDL structure fast and clear defined in a sWriter document. The main purpose of the API-Viewer will be to present the description of the UNO IDL structure and how individual elements are connected with each other. In Open Office there are a lot of `services` and `interfaces` which depend on a `module` and at the same time belong to another `service`. Because of these different connections between the UNO IDL's, users can loose the overview of them easily.

For example the `module` "`com.sun.star.awt`" consists of one `module`, 98 `services`, 111 `interfaces`, 29 `structs`, two `exceptions`, six `enums` and 36 `constant groups`. Nearly every included `service` and `interface` again consists of four to six `methods`. But this is only one `module` of the Open Office API. There are a lot of other `modules`, `services` and `interfaces` with a lot more including UNO IDL's.

To get a better overview and imagination how many elements an UNO IDL includes, the API-Viewer will illustrate the UNO IDL's in different ways like in a table with additional information about some IDL's or in a graphical view, where the elements will be displayed in rectangles ordered by types, for example by `UNO_SERVICE` or by `UNO_INTERFACE`.

The API-Viewer will operate and work through the UNO IDL's. First users shall be able to start requests about Open Office UNO IDL's through a GUI-frontend program or by calling the API-Viewer from their own program. These queries will return the elements included in the UNO IDL's to the user request. Requests about `methods`, `constants` and `enums` will provide additional important information to the included elements like their values. The API-Viewer will modify and format the query output using methods like sorting the included elements or setting hyperlinks. The hyperlinks will be connected to the information side of the linked UNO IDL either to the Open Office API in internet or the local installed Open Office API. Afterwards the API-Viewer will insert the result into an Open Office Writer document where it can be easily processed. The user will have a great variety of how the output should be formatted and presented.

1.1 Requirements

To enter the requested information into a Writer document Open Office tasks have to be automated. To achieve this goal four basic elements are needed: Open Office.org, Object Rexx, Java and BSF4Rexx. All of these programs can be downloaded from the internet for free.

The API-Viewer is platform-independent because it is based on Java, ooRexx and Open Office which can be run on most popular operating systems. It can be started separately with the GUI-based `frontend.rex` program (intended for users to choose the display and format of the output) or within other programs (to offer an `interface` for users to implement in their own programs).

1.2 Roadmap

The first chapter gives a brief overview of the required software and provides links where to download these parts. In the following chapter the structure of the API-Viewer and the possibilities to customize the output in Open Office are described. The fourth and last chapter describes some code snippets which include some main code elements used in the API-Viewer.

1.3 Help and Further Support

For the creation of the API-Viewer are a lot of sources available. Perhaps the most important sources are the **Open Office API**, the Open **Office Developers Guide**, the **Open Office Code Snippet Base** and the **Open Office Mailing List**.

- The Open Office API describes the different components which are needed for the automation of Open Office. It can be found at the Open Office homepage [OOOAPI].

- At the Open Office Code Snippet Base short examples can be found about Open Office automation [OOOCSB]. Different examples and implementations are created by the Open Office user community for the common use.

- The Developers Guide gives an overview about the Open Office components and how they can be instrumented with Java. The basic information about Open Office Writer features and appropriate examples can be found at the Open Office homepage [OOOD08].

- An important source where programmers can get personal and individual help for their questions and problems is the Open Office Mailing List. There are different lists and each of them can be subscribed or searched at the Open Office homepage [OOOPML].

1.4 Used Versions

The API-Viewer has been programmed and tested with the following programs:

- Windows XP Professional 2002 Service Pack 2,
- Java 1.6.0_06,
- ooRexx 3.2.0,
- BSF4Rexx dist. 20080913,
- Open Office 3.0 and 2.4,
- Open Office SDK 2.4,
- Vim 7.2.

2 Description of the Main Elements

The following chapters give a brief overview about the elements that were used to create the API-Viewer. The API-Viewer is based on three fundaments: Open Office.org, Object Rexx and BSF4Rexx.

2.1 Open Office.org

In 1980 StarDivision created StarOffice. In 1999 Sun Microsystems bought the StarOffice software and enhanced it to OpenOffice.org. Open Office can be downloaded for free. The latest version is available at the Open Office homepage [OOOorgd] [OOOorga].

Open Office is an open-source project. The primary contributer to Open Office is Sun Microsystem and it is maintained by an open source community having the goal that the programs are running on all major platforms and providing access to all data through APIs and XML-based file format [OOOorga].

All Open Office.org programs are compatible with the file formats of other major office software packages. This includes programs supporting the common office work. The following six products are included in Open Office [OOOorgb]:

-  **Writer:** The Writer is the word processor of the office software family. It can be used for writing letters, papers, books, memos and much more. The Writer offers a lot of possibilities to format written text and to insert tables and diagrams [OOOorgb].

-  **Calc:** The Calc program is a spreadsheet software providing all necessary tools for making calculations, analysis or creating diagrams and reports [OOOorgb].

-  **Impress:** Impress can be used to create multimedia presentations [OOOorgb].

-  **Draw:** Draw provides tools to create graphics and diagrams. Therefore with Draw shapes can be painted in various ways and diagrams can be made easily [OOOorgb].

-  **Base:** Base can be used to create databases, forms and reports. Base offers a lot of ways to modify tables, forms and queries within a database [OOOorgb].

-  **Math:** Math can be used to create mathematical equations [OOOorgb].

All these products are able to read and process files which are saved in a format of Microsoft Office too. Files can be saved in Open Office, in Microsoft and other formats like `xml` so that users not using Open Office can read the documents.

2.2 Open Object Rexx

Object Rexx is an object oriented version of Rexx, an interpreted scripting language developed by IBM. Rexx is an acronym for "REstructured eXtended eXecutor". IBM enhanced Rexx so that it could be used on all IBM platforms. Since 1987 commercial and free versions for all major operating systems are existing [ooRex08].

In the next step an object oriented version of Rexx has been created at the beginning of the nineties called Object Rexx [ooRex08]. Object Rexx has become an open source project managed by Rexx Language Association (RexxLA). The Rexx Language Association is a non-profit organization with members all over the world. It provides a free implementation of Open Object Rexx [ooRex08].

ooRexx is an object-oriented extension of the classic Rexx language and is distributed under the Common Public License (CPL) v1.0. It is an object-oriented program language and therefore includes constructs like classes, objects and methods. But this extension does not simply replace the function of the classic Rexx language. It also provides new functions additionally to classic Rexx [ooRex08].

The following list shows the main aspects of ooRexx:

- **An English-like language:** A lot of instructions of Object Rexx are English words like SAY, PULL and EXIT. This makes the use of this programming language easier [ooRex08].

- **Fewer rules:** There are only a few rules about code format, for example there is the possibility to add multiple instructions in a single line and instructions may start in any column. Keywords can also be used to name normal variables because they are only reserved in context. The Rexx interpreter can distinguish between the two names and knows to which variable it refers [ooRex08].

- **Interpreted, not compiled:** To start a program it does not need to be compiled. The language processor reads each line from the source file and executes it [ooRex08].

- **Built-in functions and methods:** Rexx provides many built-in functions like comparison operations for text and numbers [ooRex08].

- **Typeless variables:** In Rexx every data is treated as an object of different kind so variables can hold any type of object. The type of a variable does not need to be declared to use it the proper way [ooRex08].

- **String handling:** Rexx offers a functionality to manipulate character strings. This enables programs to read and separate characters, numbers and mixed input [ooRex08].

- **Decimal Arithmetic:** The arithmetic operations in Rexx are based on decimal arithmetic [ooRex08].

- **Clear error messages and powerful debugging:** When the Rexx program encounters an error, messages with meaningful explanations are displayed. With the `TRACE` instruction a debugging tool is available [ooRex08].

The latest version of ooRexx can be downloaded at the ooRexx homepage at the download area [ooRexx08a].

2.3 BSF4Rexx

BSF is the Bean Scripting Framework and is an Apache Software Foundation (ASF) open-source project of IBM. It was developed for scripting languages and offers the possibility to use Java objects and methods over defined interfaces [Flat08b].

In 2000 Peter Kalendar made the proof of concept and professor Rony G. Flatscher enhanced the program. The first version was called Essener Version. All Rexx programs have to be started with Java. In 2002/2003 the Augsburger Version was developed and offered the possibility to start Java from Object Rexx. Another feature of the Augsburger Version is that every Java-object or Java-class can be accessed and that type less arguments can be used. Three years later the Vienna Version was developed. This version contains new functions concerning the automation of Open Office.org, strict typing is not required any more [Flat08b].

The latest version of BSF4Rexx and the installation guide consisting of two readme files - the `readmeBSF4Rexx.txt` and `readmeOOo.txt` - are available at: http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current.

2.4 The UNO Environment

The Universal Network Objects (UNO) component provides an environment that enables the usage of network objects across programming languages. This UNO objects run on every environment [OOOUNO].

Important components of the UNO framework are:

- The abstract meta language UNOIDL. With this language header files and libraries can be created for the implementation of UNO objects in the programming language which is used [OOOUNO].
- A service manager offering a database for registered components. These components can be created by their name. Using the objects is transparent to the programmer, communication works through the `interface` provided by the UNOIDL [OOOUNO].

- Bridges using the UNO remote protocol (URP) arrange communication between two different implementation languages. These bridges send method calls and receive return values [OOOUNO].

- There is an application programming interface (API) describing the Open Office objects and features [OOOUNO].

2.4.1 The Open Office API concept

The main aim of the Open Office API is to provide the Open Office functionality for every user. The Open Office API includes different data types:

- `Interfaces` providing the functionality of objects to the outside. The internal aspects of the object are not visible [OOOUNO].

- `Services` consisting of `interfaces`. For each `service` a description is provided [OOOUNO].

- `Properties` are included in `services`. They are defining the features of an object in a `service` instance [OOOUNO].

- `Structs` are describing a couple of elements within a record. They do not encapsulate the included data [OOOUNO].

- Predefined values like `constants` or `enums`: `Constants` include a group of `constant` values. The type `enum` is like the enumeration type in C++. An `enum` consists of a sequentially ordered list of values. The sequence always starts at zero and always adds one for each new value [OOOUNO].

- `Modules` which are name spaces like in C++. They are grouping the other data types like `services`, `interfaces` and `structs` [OOOUNO].

- `Exceptions` show if errors occur. The thrown exception gives a basic idea and description of the error [OOOUNO].

- `Typedefs` have to be replaced with an alias name when they are mapped from the UNOIDL to Java because they are not visible in the Java language binding [OOOUNO].

- `Singletons` are used for named objects where only one instance may be created [OOOUNO].

2.5 Other Programs

BSF4Rexx can access Open Office using the Open Office Java bridge. Therefore Java has to be installed on the computer. There are different versions for all major operating systems available at the Java homepage [JavaSun]. To install Java on a Windows operation system the downloaded `.exe` program has to be started. At the Java Sun homepage there are installations notes for Windows, Linux and Solaris operating system which help to install and configure Java correctly on the target computer [JavaSuna].

To highlight the ooRexx syntax during programming, editors like the Vim editor can be used. This editor can be downloaded at the Vim homepage [Vimorg]. After downloading on a Windows operation system, the `.exe` program has to be started. For installing, the standard configuration can be used. After finishing the installation, programming can be started.

3 API-Viewer

The API-Viewer is a program which allows the user to send requests about the API elements of Open Office. These elements are processed and prepared for the display in a Writer document. They are added to two different tables. The first table lists the elements sorted alphabetically, and the second the members sorted by the UNO type. For some elements information is provided like it is the case for `methods`, `services` or `interfaces`. For each element a hyperlink is added which links directly to the Open Office API that the user can look up for more detailed information. Furthermore a graphical display of the queried IDL and its containing elements is created. So the user gets an overview over the requested UNO IDL and the included elements.

The created sWriter document can be stored in two different formats. It is possible to store it as an `.odt` file or as a `.pdf` file. The `pdf` format has the advantage that it can be opened without Open Office. How a sWriter document can be stored and printed automatically can be found in chapter 4.14.

There are different ways to get access to the API-Viewer. First you can start the `frontend.rex` program. This will display a graphical interface of the API-Viewer which makes it easier for the user to choose the settings which could be used for the view of the user's API query. There is also the possibility to start the API-Viewer through another program. People who want to integrate the API-Viewer in their program or would like to start queries during programming can call the API-Viewer within their own program. To start the API-Viewer the `analyzeAndCreateReference` routine has to be called. This routine has different arguments to enable the user to customize the display of the document. Only one argument has to be specified from the user, the others have default values. `analyzeAndCreateReference queryIDL [, cfLayer [, cfView [, cfURL [, localLink [, cffirstlevel [,cfsecondlevel [,cfthirdlevel [,cfFont]]]]]]]]` Only an UNO IDL has to be passed to the `analyzeAndCreateReference` routine so that the sWriter document can be created.

`frontend.rex` is the graphical interface from the API-Viewer. It can be started through the command line with the following command: `rexx frontend.rex`.

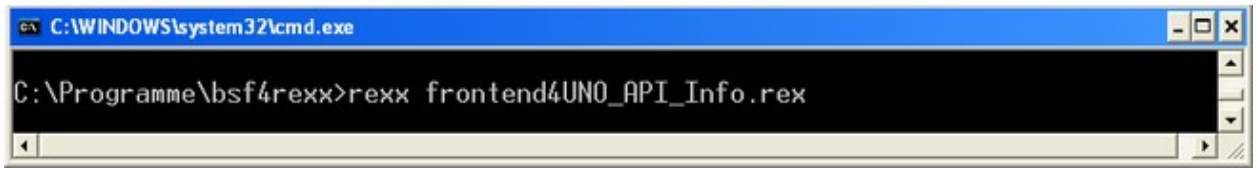


Figure 1: Starting `frontend.rex`

After pushing enter, the program starts and a graphical interface is displayed on the monitor. The following figure shows the graphical user interface. The graphical display gives the user the possibility to easily change the format of the created sWriter document with a mouse click. Therefore the user only has to open the combo box and to choose the needed item. If the user does not change the current selected elements the defaults are used for the document creation.

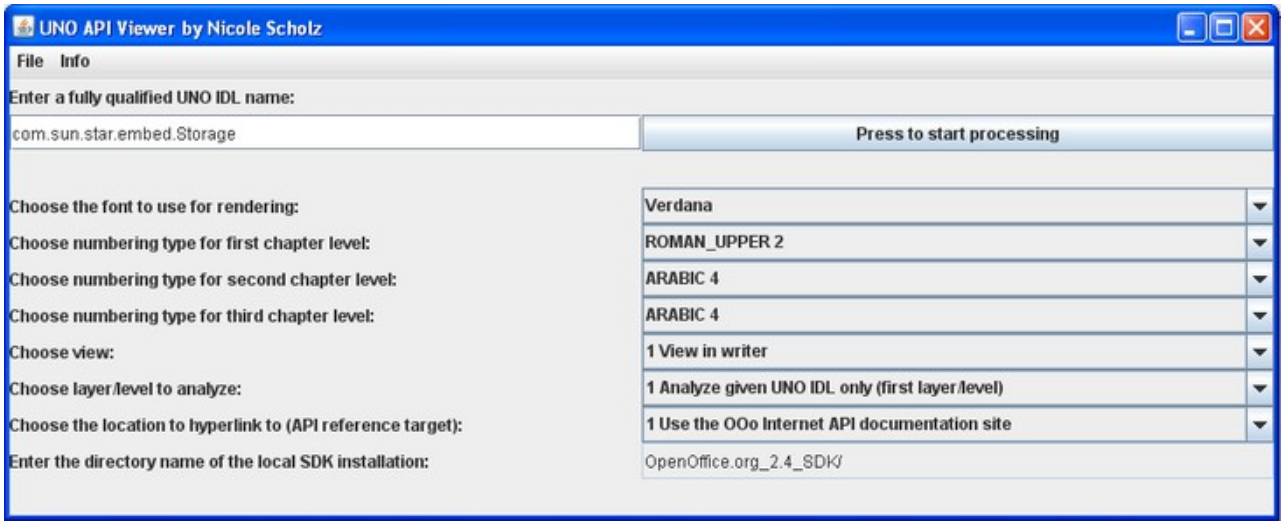


Figure 2: API -Viewer graphical interface

The next two figures show the display of the API – Viewer. First, the table of content is shown. Then on the next page the members of the UNO IDL are listed in two different tables. The first one is sorted alphabetically, the second by the UNO type name the members belong too. Within these type names the members are sorted alphabetically. On the last page the UNO IDL members are displayed graphically. For every UNO type name a rectangle is created and the member names are added to the right UNO type name rectangle. The amount of pages varies with every queried UNO IDL because they have different amounts of members. But the order of the created chapters is always the same: the first is the table of content, the second is the table with the members ordered alphabetically, the third is the table including the members ordered by UNO type name and the fourth is the graphical display of the UNO IDL.

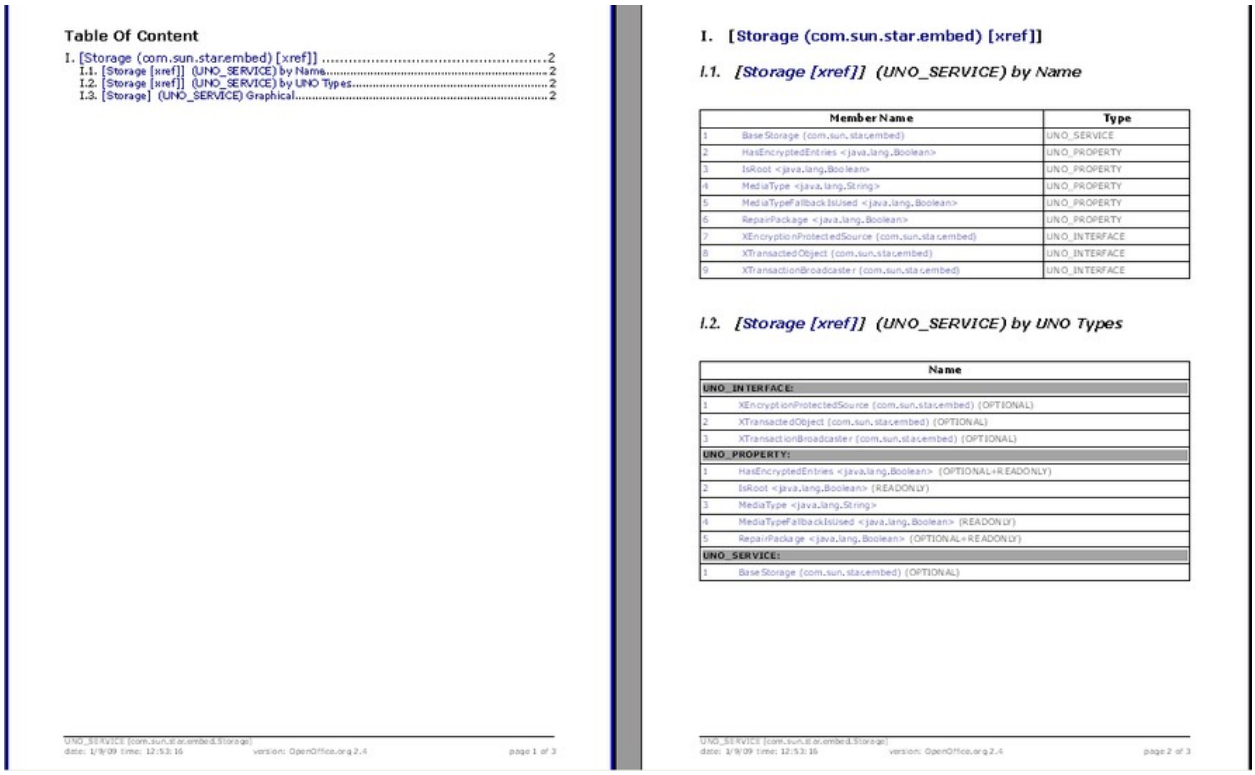


Table Of Content

UNO_SERVICE (com.sun.star.embed.Storage)
date: 1/9/09 time: 12:53:16 version: OpenOffice.org 2.4 page 1 of 3

I. [Storage (com.sun.star.embed) [xref]]

I.1. [Storage [xref]] (UNO_SERVICE) by Name

	Member Name	Type
1	BaseStorage (com.sun.star.embed)	UNO_SERVICE
2	HasEncryptedEntries <java.lang.Boolean>	UNO_PROPERTY
3	IsRoot <java.lang.Boolean>	UNO_PROPERTY
4	MediaType <java.lang.String>	UNO_PROPERTY
5	MediaTypeFallbackIsUsed <java.lang.Boolean>	UNO_PROPERTY
6	RepairPackage <java.lang.Boolean>	UNO_PROPERTY
7	XEncryptionProtectedSource (com.sun.star.embed)	UNO_INTERFACE
8	XTransactedObject (com.sun.star.embed)	UNO_INTERFACE
9	XTransactionBroadcaster (com.sun.star.embed)	UNO_INTERFACE

I.2. [Storage [xref]] (UNO_SERVICE) by UNO Types

	Name
UNO_INTERFACE:	
1	XEncryptionProtectedSource (com.sun.star.embed) (OPTIONAL)
2	XTransactedObject (com.sun.star.embed) (OPTIONAL)
3	XTransactionBroadcaster (com.sun.star.embed) (OPTIONAL)
UNO_PROPERTY:	
1	HasEncryptedEntries <java.lang.Boolean> (OPTIONAL+READONLY)
2	IsRoot <java.lang.Boolean> (READONLY)
3	MediaType <java.lang.String>
4	MediaTypeFallbackIsUsed <java.lang.Boolean> (READONLY)
5	RepairPackage <java.lang.Boolean> (OPTIONAL+READONLY)
UNO_SERVICE:	
1	BaseStorage (com.sun.star.embed) (OPTIONAL)

UNO_SERVICE (com.sun.star.embed.Storage)
date: 1/9/09 time: 12:53:16 version: OpenOffice.org 2.4 page 2 of 3

Figure 3: Display of UNO IDL – page 1 and 2

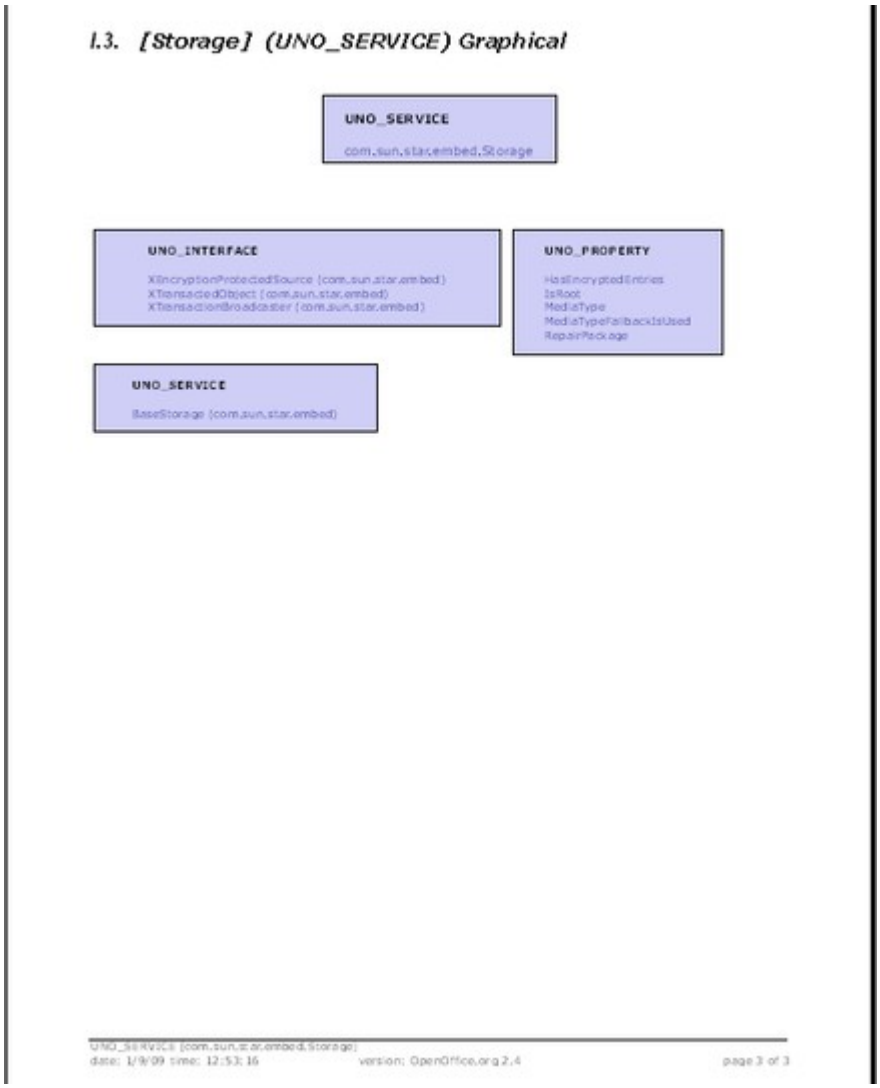


Figure 4: Display of UNO IDL - page 3

3.1 CreateApiInfo.rex

In the following chapter the structure and the functionality of the `createApiInfo.rex` program are described. The focus is on how the program works and which different ways of demonstration of the queried IDL's can be created and displayed.

3.1.1 UNO_ReflectionUtil.cls

The `createApiInfo.rex` program requires one other program: the `UNO_ReflectionUtil.cls`. The `UNO_ReflectionUtil.cls` provides the UNO classes, different methods and attributes to each of them so the queries of the API-Viewer can be

made fast and easy. The `UNO_ReflectionUtil.cls` loads the `UNO.CLS` which is required to get a connection to Open Office. This connection has to be implemented to ensure Open Office can be automated with Object Rexx. The `UNO service` from Open Office.org provides bridges to objects written in different programming languages so that it can also be used to get the connection to Open Office. After connecting to Open Office method calls can be send and return values can be received between Rexx and Open Office [OOOD08a]. The `UNO.CLS` loads the `BSF:CLS` which is needed to connect to Java so that Java classes can be used with Object Rexx. To generate these connections the following code is used [RGF08e].

```
-- get OOo-reflection utilities (which loads UNO.CLS, which loads BSF.CLS)
::requires UNO_ReflectionUtil.cls
```

Code 1: Requirement of UNO_ReflectionUtil.cls

3.1.2 AnalyzeAndCreateReference routine

The main routine of the API-Viewer is the `analyzeAndCreateReference` routine. In this routine the other routines are called and the whole query procedure is started. This routine needs six arguments to start the query process. All these arguments have default values too if not every element is specified. These default values are set that the query about the UNO IDL's can be started if not every element is passed to the `analyseAndCreateReference` routine. So the routine can be easily called within another program without specifying every argument. The first element which is committed to the routine is the `queryIDL`. This variable contains the queried UNO IDL and if it is not specified the nil object is used. Second, the layers which should be shown are added to the routine. This variable can have two values: one and two. If the `cfLayer` is one, only the UNO IDL and its elements are shown. The next variable the `cfView` can have four values depending on the chosen view:

- It can be chosen that the query is only shown in a Writer document,
- that it is shown as Writer document and saved as `*.odt`-file,
- that the query is saved as `*.odt`-file and the document is printed

- or that it is saved as `*.odt`-file and saved additionally as `*.PDF`-file.

If the `cfLayer` is not specified, the default value is one and therefore only one layer is displayed. The `cfURL` can have one of these three values:

- one if the internet API is chosen,
- two if the local API without jump marks is selected and
- three if the local API with jump marks is requested.

If nothing is selected the `cfURL` is set to one so the internet API is used. The variable `localLink` is set to an empty string if nothing is passed to the routine. Finally the `cfFont` variable is defined with the font the user chooses. If no font is selected then it is set to `Arial`. To retrieve the arguments the command `use arg` is used. This command retrieves argument objects provided in functions, routines or procedures [AFHMMPw].

```
::routine analyzeAndCreateReference public
  use strict arg queryIDL=.nil, cfLayer=1, cfView=1, cfURL=1, localLink="", cfFont="Arial"
```

Code 2: `analyzeAndCreatReference` routine arguments

The `"com.sun.star.frame.Desktop"` `service` is the environment for the components. It is needed to load the required components to create a Writer document. To create this `service` the program code `UNO.createDesktop()` has to be executed. Next, the `XComponentLoader` `interface` has to be accessed. The component loader is needed for components to be loaded into the frame environment. The `XComponentLoader` is accessed through the `XDesktop` as the following code shows. [OOOAPIa]

```
-- start the Writer of Open Office
-- get the UNO Desktop service object
oDesktop         = UNO.createDesktop()
-- get componentLoader interface
xComponentLoader = oDesktop~XDesktop~XComponentLoader
```

Code 3: Start Open Office sWriter

These lines of code result in a opened blank sWriter file with a document title.

3.1.3 Creating the “Table of Content”

The generated sWriter document always has a table of content on the first page. So the user has a quick overview what elements of the request are created and on which page he finds them. Each entry in the table of content includes a hyperlink which is linked to the page where the headline is placed. So users can jump from the table of content to the chapter they like. The table of content is formatted and has the same font as the rest of the sWriter document to achieve a consistent layout within this document. The character height depends on the level of entries. If the first level is requested, the height it is set to twelve. For second level it is only set to ten. Furthermore the second level is indented. This arranges the table of content so that it is clear which level includes other chapters. How a table of content can be created is defined in the chapter 4.9.

Another important point is that the underlining for the hyperlinks is deactivated for the whole document. When the hyperlinks in the table of content are activated so that the user can jump to the chapters the elements are usually underlined. This makes the table of content difficult to read. So the underlining is deactivated that the table of content is well arranged and well too read. How to deactivate the hyperlink underlining can be found in the chapter 4.8.

The following Figure shows the table of content of a query requesting `"com.sun.star.embed.Storage"`. The title “Table of Content” font height is set 14, the first level font height is set 12 and the second level font height is set to 10. Only the text is underlined and referenced as hyperlink. The space between the text and the right aligned page number is filled with points.

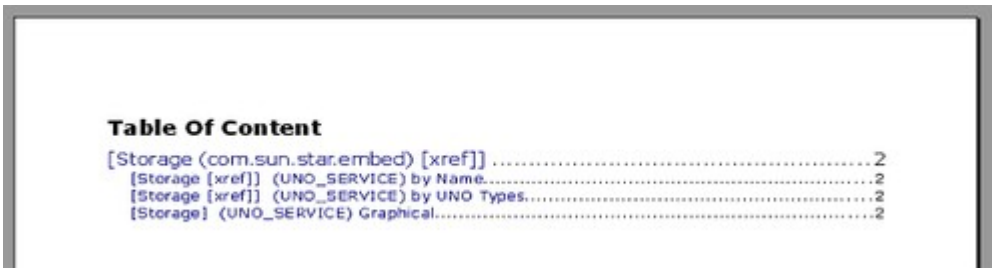


Figure 5: “Table of Content”

3.1.4 Chapter Numbering

The created sWriter document can have a lot of chapters, if an UNO IDL with many members is displayed. Furthermore, when a second layer is shown, the chapter amount increases. To help the user to be able to keep an overview about the chapters and the listed UNO IDL's a chapter numbering is added to the document.

There is the possibility for the user to choose from different numbering types. Users can choose which numbering type they would like to have on which numbering level. As the document has three chapter levels, users can customize the numbering types for three levels. It is possible to choose the same numbering type which is provided by Open Office. The numbering types are taken directly from Open Office to ensure that they are always up to date.

This gives the user the possibility to distinguish between a normal request and a request including the second layer. The user can choose for both documents different chapter numbering types. If the user does not define which numbering type he wants to use, then for the first level Roman upper letters are used and for the other levels Arabic numbers.

How the chapter numbering can be programmed can be read in the chapter 4.10.

The following three figures show two numbering possibilities. In the first figure for the first level, chars upper letters are used. For the next two levels Arabic numbers are chosen. In the second figure, for the first level Arabic numbers are used and for the second chars lower letters. In the last figure as example for the first level, Tibetan alphabet letters are used, for the second Japanese characters and for the third Thai alphabet letters.

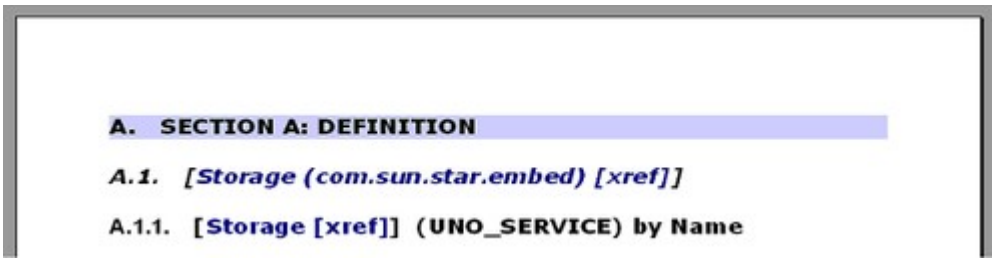


Figure 6: Chapter numbering - numbering types 1

Figure 7: Chapter numbering - numbering types 2

Figure 8: Chapter numbering – numbering types 3

3.1.5 Footer

The created sWriter document always includes a footer. This footer contains additional information like the displayed IDL, date and time of creation, used Open Office installation and the page number. This information should give the user the possibility to see on which date the document was generated and with which Open Office version. So it is obvious which elements have changed between the different versions. How a footer can be programmed is explained in the chapter 4.3.

The used Open Office installation is important because the UNO IDL's can change from version to version. Therefore the current version is displayed in the footer, so that users can always see which document is created under which version.

To get the current installed Open Office version the routine `query_OOo_Installation` is called at the begin of the `createApiInfo.rex` program. The routine has the argument `xContent` which is set `.nil` when no object is passed to the routine. The Nil object is an object with no containing data [AFHMMPc]. If the `xContext` variable is `.nil` then a connection to the UNO `service` is established with `UNO.connect()` to retrieve the `xContext` object [RGF08c]. Then the `XMultiComponentFactory` is accessed with the command `getServiceManager` [OOOAPIs]. Afterwards the `service` object of the configuration

provider `"com.sun.star.configuration.ConfigurationProvider"` is queried and the `XMultiServiceFactory` is added to it [OOOAPIu] [OOOAPIt]. After this an array for the properties is created and the `property` `nodepath` with the corresponding value is saved in it [AFHMMPd].

Next an instance with the arguments `"com.sun.star.configuration.ConfigurationAccess"` additionally to the before created property array is created with the `method` `createInstanceWithArguments` of the `XMultiServiceFactory` [OOOAPIt]. With the `interface` `XNameAccess` and its `method` `getByName` the version of Open Office can be queried and saved in the `.scholz` directory [OOOAPIw] [OOOAPIv]. A directory is a collection with unique indexes which are representing names. In a directory the needed data can be accessed with an assigned name. A directory can be created with the command `name=.directory~new` [AFHMMPa]. The `.scholz` directory includes important information like the Open Office version, font height and font. This directory can be accessed everywhere in the program because it is a local variable. A local variable is created with the command `.local~variable` [AFHMMPb].

```
::routine query_OOo_Installation
   use strict arg xContext=.nil

     -- get the version of Open Office
   if xContext=.nil then
      xContext = UNO.connect()  -- connect to server and retrieve the XContext object
   XMcf = xContext~getServiceManager   -- retrieve XMultiComponentFactory

     -- get service object
   service="com.sun.star.configuration.ConfigurationProvider"
   sProvider=XMcf~createInstanceWithContext(service, xContext)
   xProvider=sProvider~XMultiServiceFactory  -- get its service manager

     -- create property array to query version
   propArr=bsf.createArray(.uno~propertyValue, 1)
   prop=.uno~propertyValue~new
   prop~name ="nodepath"
   prop~value="/org.openoffice.Setup/Product"
   propArr[1]=prop
     -- get a read-only ConfigurationAccess from the sProvider's service manager
   service="com.sun.star.configuration.ConfigurationAccess"
   aSettings=xProvider~createInstanceWithArguments(service, propArr)
   xSettings=aSettings~xNameAccess           -- get interface to access by name

     -- save name and version of product in .local
   .scholz~ooName        =xSettings~getByName("ooName")
   .scholz~ooSetupVersion=xSettings~getByName("ooSetupVersion")
```

Code 4: Get current Open Office version

The following figure shows the footer of the API-Viewer. It contains the UNO type name, the queried IDL-name, the date and time of creation, the used Open Office version, the actual page number and the total page number of the document.

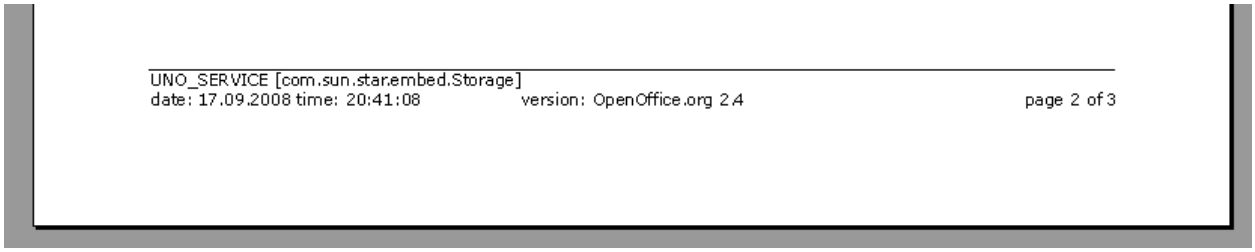


Figure 9: Footer

3.1.6 Process UNO IDL

After creating the table of content and the footer the request about the UNO IDL is sent. Here one distinguishes between calling IDL's and requesting UNO `services`. If an UNO `service` is passed to the `analyzeAndCreateReference` routine the array `arrQueryIDL` has more than one item. An array is a collection of indexes. There are the same numbers of indexes like items in the array. The indexes are used to refer to a item in the array. To get the amount of added elements the command `~items` is used [AFHMMPd]. If there are more items, the following code is executed. First in the command window the information "First processing aggregate service definitions …" is displayed so the user notices that a `service` is passed to the program. Afterwards a reflection object from the UNO definition follows. We can obtain the definition object with the following command: `unoObject~uno.getDefinition` [RGF08a].

When a `service` object is passed to the routine, sections in the Writer document are created. These sections help the user to distinguish what are the delivered `services` and what are the definitions of these `services`. Additionally section titles are set. Here the first section is called "SECTION A: SERVICE OBJECT" concatenated with the requested `service` name. All current services included in the passed service are listed. To show the user what is currently processed, the text "The Uno service object implements multiple services, namely:" plus the included `services` are displayed in the command window. To denote all `services` a loop over the array `arrQueryIDL` is implemented. This loop includes

the text mentioned before, a tabulator which is created with the code "0a09"x and the service.

```
   sectiontitle = "SECTION A: SERVICE OBJECT " refObj~fullyQualifiedName
   lfTab="0a09"x
   hintText="The UNO service object implements multiple services, namely:" "0a"x
   do service over arrQueryIDL   -- iterate over service
      hintText=hintText || lfTab || service
   end
```

Code 5: Service object loop

The following figure shows the text output in the command window during the query of UNO service object "com.sun.star.document.OfficeDocument". First the services included in the service object are listed. Afterwards the elements of the IDL are displayed successive when they are processed.

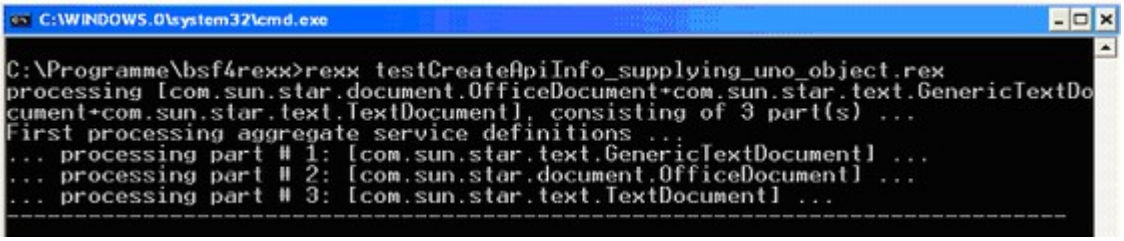


Figure 10: Command window: output UNO service object query

The next figure displays a command window with the text output printed in case an UNO service object is queried and the second layer is shown. First the processed service objects are listed and then the aggregate service definitions are displayed. Finally the member definitions included in the services are shown.

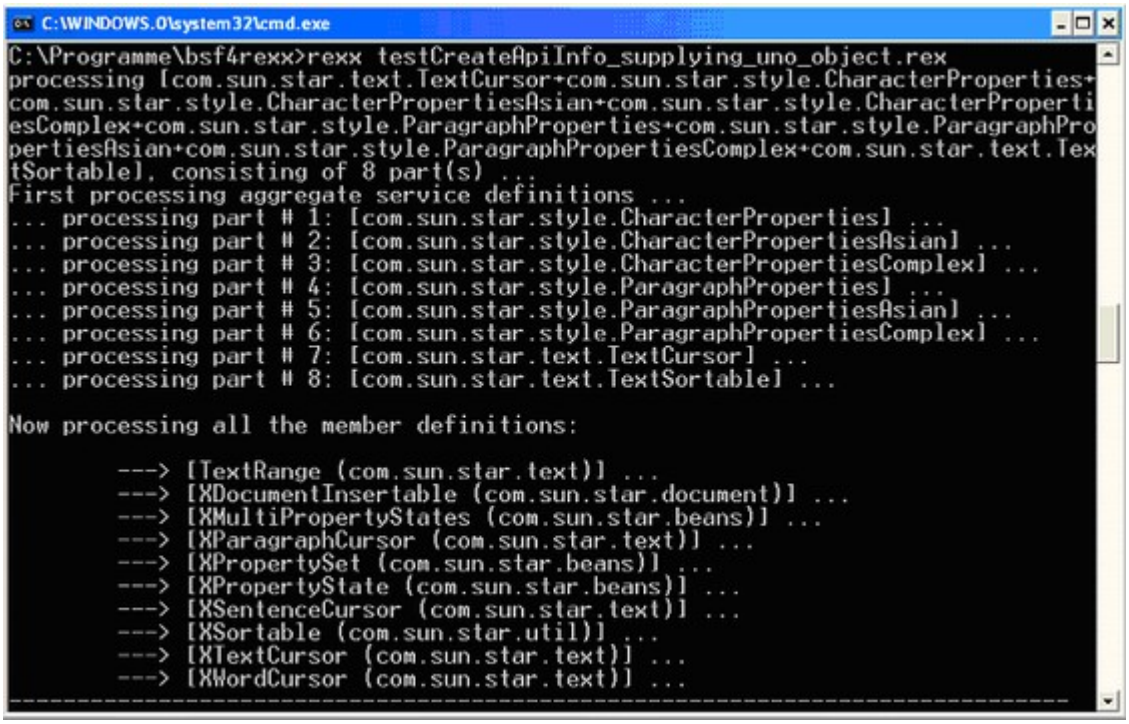


Figure 11: Command window output `UNO service object` query with second layer

The UNO IDL of every members are processed. Therefore the definition of the UNO IDL is accessed with the command `uno.getDefinition`. This command returns a string with all UNO IDL information [RGF08a]. Then an output line for the command window is created with the text "…processing part #" plus the UNO IDL. Afterwards it is checked, if the UNO object has members. If it has no members, the hint text "No members defined" is set. Later this text is inserted into the Writer document. Following the section title is set, if an `UNO service object` is queried. Then two different tables and the graphical display are created. How the tables and the graphic are created is described in the chapters 3.1.8, 3.1.10 and 3.1.11.

The following picture illustrates the `"com.sun.star.awt.PrinterException"` `exception` which has no members defined.

Figure 12: No defined members: `"com.sun.star.awt.PrinterException"`

The next figures show the output, given when processing the queried UNO IDL `"com.sun.star.embed.Storage"` and the display in the Writer document. On the first page of the Writer document the table of content is set, then two tables with different sorting methods are illustrated and on the last page the graphical display is shown.

Figure 13: Command window: output of the query `"com.sun.star.embed.Storage"`

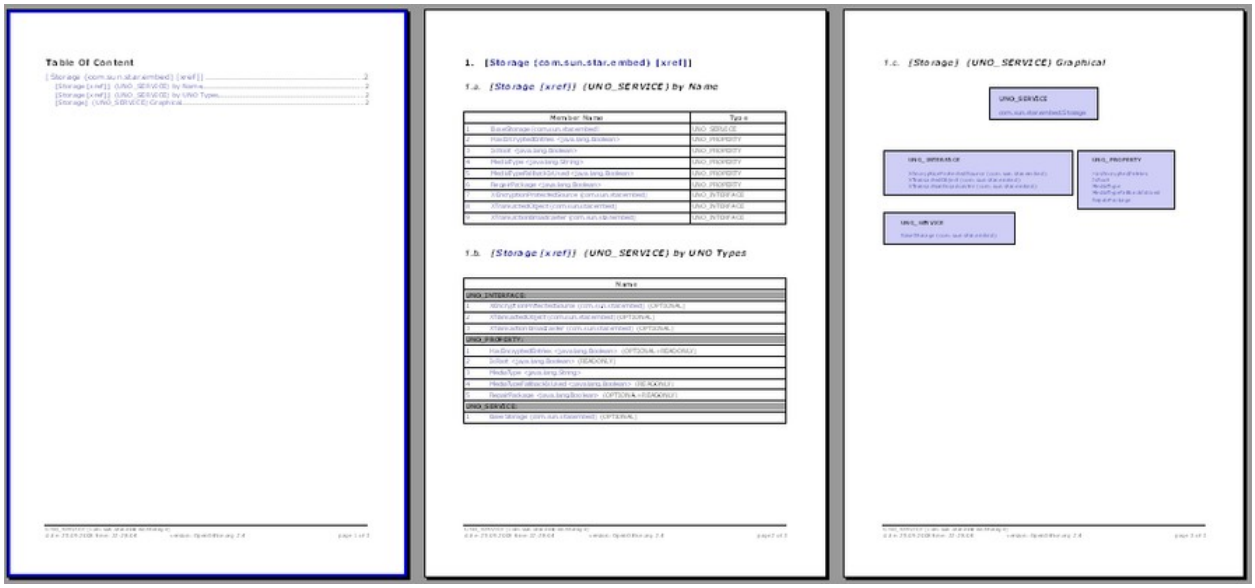


Figure 14: Created Writer document of the `service Storage`

3.1.7 Sections

The created sWriter document is divided into sections. A section is created when the second layer or a `service object` is displayed. When the second layer is displayed in the first section, the elements included in the requested UNO IDL are shown. In the second section the elements of the second layer are displayed. These sections should make the document easier to read and more structured.

If a second layer is displayed or a `service object` is passed to the `createApiInfo.rex` program, then the background color of the inserted headline is set in violet color with the `property ParaBackColor`. With this `property` the whole line is colored, not only the text. The color code is saved in the `.scholz` directory and can be accessed with the word `fillcolor` [OOOAPIz]. The `service object` is set to value one in the `createMainSectionHeading` routine when the `servicetitle` is not empty. If the “no second layer property“ is shown or no `service object` is requested, then the background color is set to white again. Therefore a box with the color code `ffffff` for white is added to the `ParaBackColor property`. With the next code lines font, character height and character weight are changed. Hence the properties `CharFontName`, `CharHeight` and `CharWeight` of the `service` `“com.sun.star.style.CharacterProperties”` have to be modified. The assigned values are saved in the `.scholz` directory also [OOOAPIaa]. A page break is added before the title is inserted in the Writer document every time a new section starts. Therefore the `enum BreakType` has to be imported with the following code: `call bsf.import “com.sun.star.style.BreakType”, “pageBreakType”.` From now on a page break can be set with the `setProperyValue method` with the arguments `“BreakType”, .pageBreakType~”PAGE BEFORE”.` `PAGE BEFORE` is used to achieve that the page break is created before the text is inserted, if `PAGE AFTER` is selected, the page break is applied after the object it belongs to [OOOAPIab] [OOOCSBe] [OOOD08c].

```
if cfLayer = 2 | serviceobject = 1 then
do
    -- set different background color for the section heading
    props~setPropertyValue("ParaBackColor", .scholz~fillcolor)
end
else
     -- set the background color white again
   props~setPropertyValue("ParaBackColor", box("int", "ffffff"x ~c2d))
   props~setPropertyValue("CharFontName", .scholz~font)
   props~setPropertyValue("CharHeight",   .scholz~entry(charHeight)) -- set font size 14
   props~setPropertyValue("CharWeight",   .scholz~bold)
   if (place = "first") then
   do
```

```
        props~setPropertyValue("BreakType",.pageBreakType~"PAGE_BEFORE")
    end
```

Code 6: Set text properties

The following figure shows the section headline. Its background color is set to violet so that it attracts attention to indicate the different sections to the user.

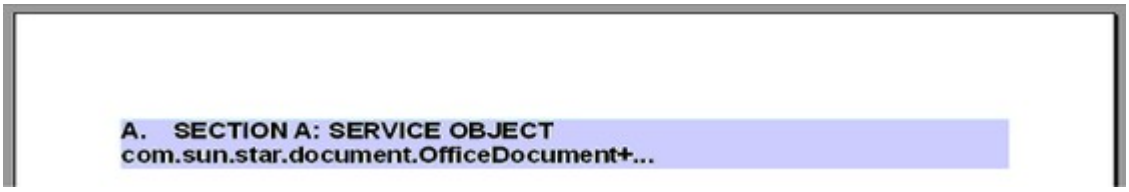


Figure 15: Section title

The next figure shows the headline of a new section when the second layer of an UNO IDL is displayed. Before the new section starts, a page break is set so that the section begins at the top of the page.

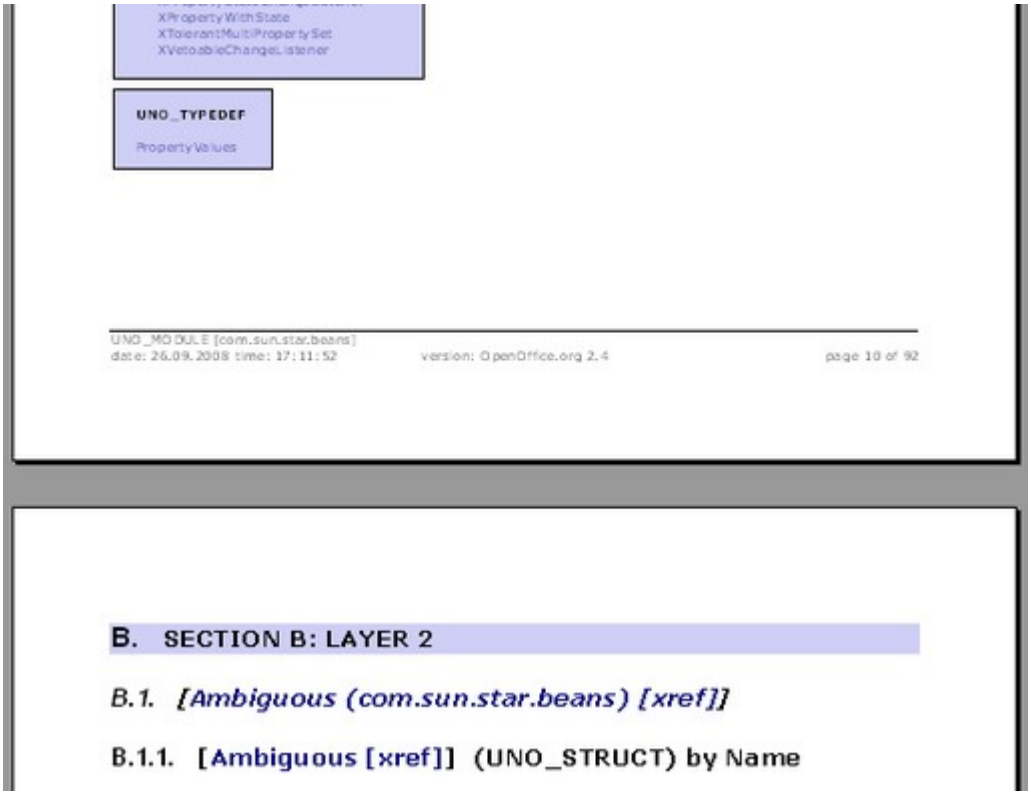


Figure 16: Page break before display of second layer

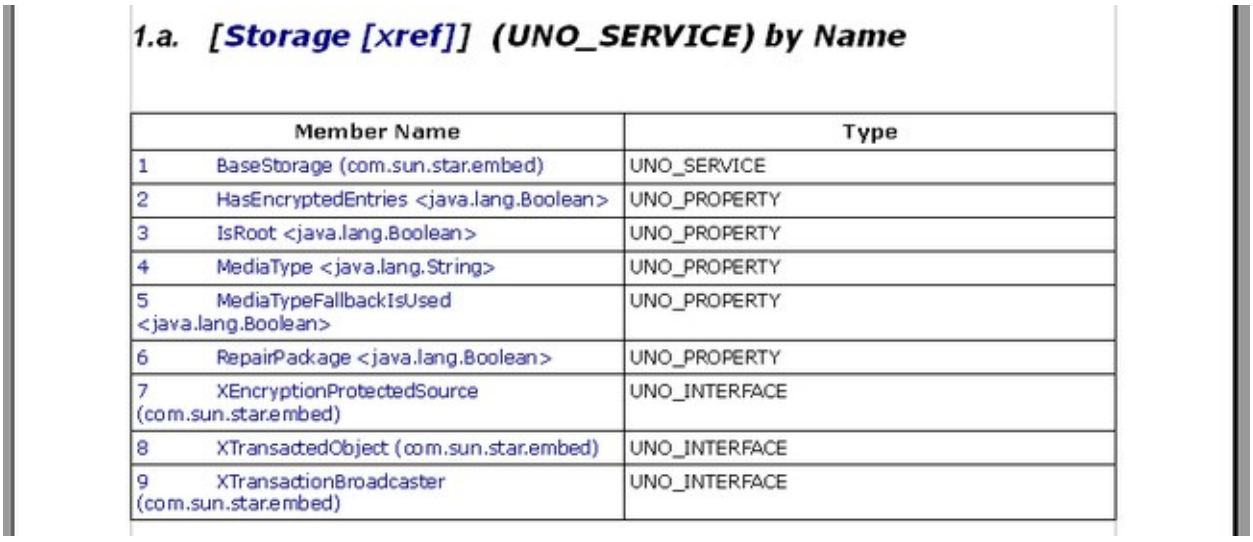
3.1.8 Table Overview

One main part in the created sWriter document is the table with all members of the UNO IDL. These members are sorted before they are added to the table. The created table has two columns, so that each added member gets its UNO type name assigned in the second column. Users can search for the UNO IDL member easily because they are sorted alphabetically and people see which to which type each member belongs.

The name of the UNO IDL members is longer than the UNO type name but the cells of the created table have the same size. That the cells are well arranged, the cell width has to be changed, so that the first column is larger than the second one. How the cell width can be adjusted can be read in the chapter 4.4.

Hyperlinks are added to the members of the UNO IDL, which link to the Open Office API. So the members are colored blue, so that they can be recognized as hyperlink. How the hyperlink is created can be found in the chapter 3.1.9.

The next picture shows a table in Writer where the column width is set automatically. The two columns have both the same size regardless how long the inserted text is.



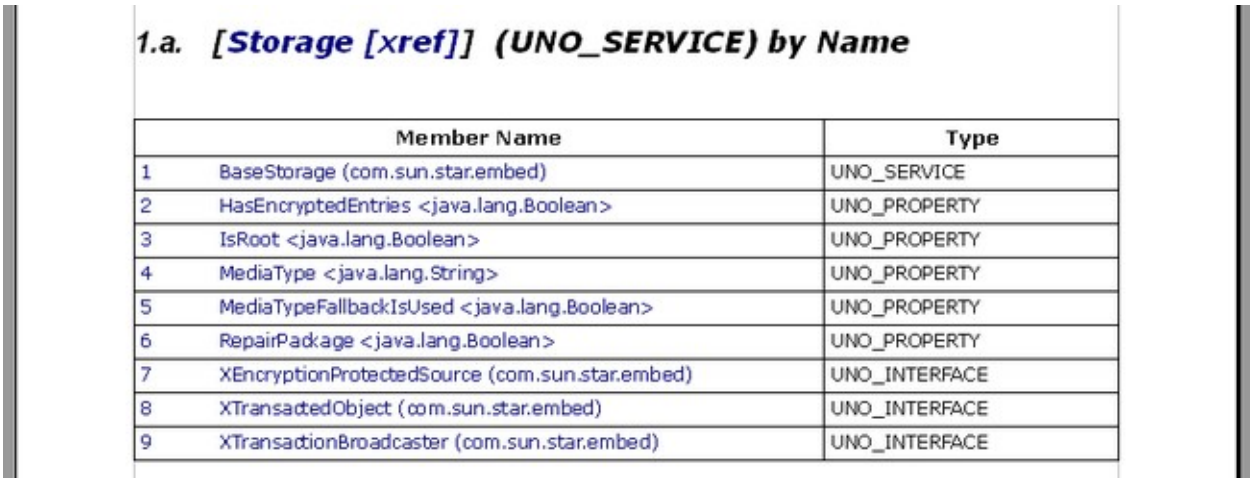
1.a. [Storage [xref]] (UNO_SERVICE) by Name

	Member Name	Type
1	BaseStorage (com.sun.star.embed)	UNO_SERVICE
2	HasEncryptedEntries <java.lang.Boolean>	UNO_PROPERTY
3	IsRoot <java.lang.Boolean>	UNO_PROPERTY
4	MediaType <java.lang.String>	UNO_PROPERTY
5	MediaTypeFallbackIsUsed <java.lang.Boolean>	UNO_PROPERTY
6	RepairPackage <java.lang.Boolean>	UNO_PROPERTY
7	XEncryptionProtectedSource (com.sun.star.embed)	UNO_INTERFACE
8	XTransactedObject (com.sun.star.embed)	UNO_INTERFACE
9	XTransactionBroadcaster (com.sun.star.embed)	UNO_INTERFACE

Figure 17: Table with normal column width

The next figure displays a table with changed column width so that the second column is smaller than the first one because the text in the first one is longer than in the second.

The UNO IDL members are sorted alphabetically and to each member the UNO type name is given in the second column.



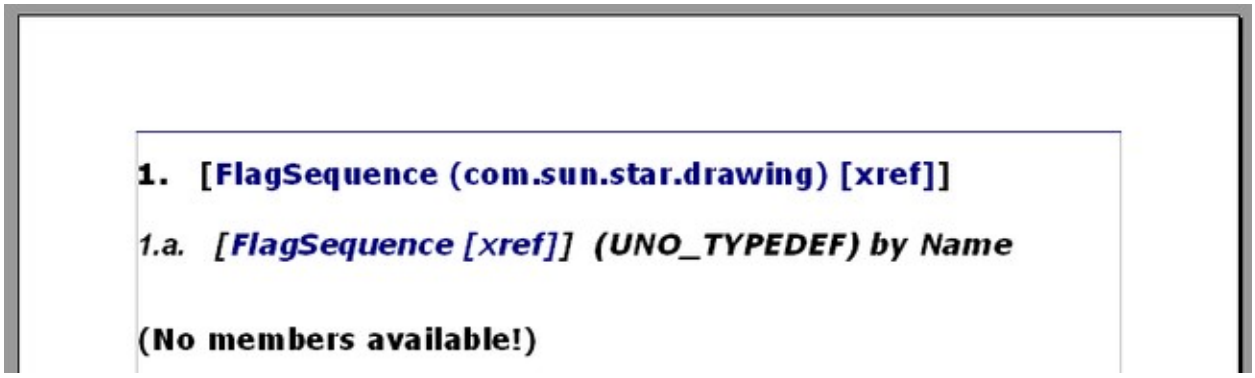
1.a. [Storage [xref]] (UNO_SERVICE) by Name

	Member Name	Type
1	BaseStorage (com.sun.star.embed)	UNO_SERVICE
2	HasEncryptedEntries <java.lang.Boolean>	UNO_PROPERTY
3	IsRoot <java.lang.Boolean>	UNO_PROPERTY
4	MediaType <java.lang.String>	UNO_PROPERTY
5	MediaTypeFallbackIsUsed <java.lang.Boolean>	UNO_PROPERTY
6	RepairPackage <java.lang.Boolean>	UNO_PROPERTY
7	XEncryptionProtectedSource (com.sun.star.embed)	UNO_INTERFACE
8	XTransactedObject (com.sun.star.embed)	UNO_INTERFACE
9	XTransactionBroadcaster (com.sun.star.embed)	UNO_INTERFACE

Figure 18: Table with changed column width

If the UNO IDL has no members then the text “(No members available!)” is displayed instead of a table. In this case three paragraph breaks are inserted and the next headline is added after some space.

The next Figure shows the display of the `typedef FlagSequence`. This `typedef` has no members so only text and no table is displayed.



1. [FlagSequence (com.sun.star.drawing) [xref]]

1.a. [FlagSequence [xref]] (UNO_TYPEDEF) by Name

(No members available!)

Figure 19: UNO IDL without member table overview

3.1.9 Hyperlink to the Open Office API

A hyperlink is added to the elements of the UNO IDL. This hyperlink is different depending on which UNO IDL is queried and if the internet or the local API of Open Office

is used. The hyperlink is important, because the user can read up the whole definition of the UNO IDL in the Open Office API and which arguments it includes. The possibility to choose between the internet API or the local API has the advantage that the hyperlinks to the local API are also working if no internet is available. Users are independent to the availability of an the internet connection. If the internet API is used, it has the advantage that the document can be opened in another environment with an internet connection and the hyperlinks to the more detailed information are working. If the local API is used and it is not installed on another computer or installed in a different directory the hyperlinks will not work.

How a hyperlink can be set is described in chapter 4.5 and how it can be set within a table is shown in chapter 4.6.

The appearance and the presentation of the created hyperlinks are different depending on the used Open Office API, the environment in which the document is generated and which UNO IDL is hyperlinked. But it always consists of three elements:

- the link to the Open Office API location (the internet link or the local directory),
- the words `docs/common/ref`,
- and the UNO IDL type name.

These elements have to be added depending on the used API and the displayed UNO IDL.

If the internet API is selected the link `http://api.openoffice.org/docs/common/ref/` is added to the `.scholz` directory as base URL. Otherwise the local directory of the Open Office SDK has to be set as base URL. Hence the location of it has to be queried. Therefore it is checked which operation system is currently in use. This information can be accessed with the command `.java.lang.system~getProperty("os.name")`. After saving the operating system into the variable with the command `left`, the first letter of the variable is checked [AFHMMPj]. If the first letter of the variable `a` is a `"W"`, then the location is found with the command `value`. This command can be used to manipulate the environment. Therefore the second argument has to be “ENVIRONMENT”. The result is

the path where the programs are located. The result is saved in the `.scholz` directory with the remaining path of the hyperlink. [AFHMMPi]. If the first letter is not `"W"`, then the base URL gets the `locallink` passed. The `locallink` is specified from the user in the `frontend.rex` program or has to be passed to the `analyzeAndCreateReference` routine. After creating the first part of the link the UNO type name is added to it with a `"#"` before the type name like `"#Services".`

```
.local~selectedUrl = cfURL
if .selectedUrl = 1 then
    -- link into the OOo's Internet base URL for IDL reference documentation
    .scholz~ooo.baseUrl="http://api.openoffice.org/docs/common/ref/"
else
do
   -- link into the OOo's local directory base URL for IDL reference
   -- documentation
  a=.java.lang.system~getProperty("os.name")
  if a~left(1)="W" then                        /
  do
     cp=value("ProgramFiles", , "ENVIRONMENT")
     cplink = cp~changestr("\","/")
     .scholz~ooo.baseUrl=cplink"/"locallink||"docs/common/ref/"
  end
  else
     .scholz~ooo.baseUrl=locallink||"docs/common/ref/"
end
```

Code 7: System environment

The following two figures show how a hyperlink for the `service Storage` is composed and how the target of the link looks like. To get the target of the hyperlink, the hyperlinked text has to be clicked with the control key. Then a browser window or a new tab in an already existing browser window will be opened with the target homepage. The link, which is displayed in the figure below, consists of the Open Office Homepage `http://api.openoffice.org`, the text `docs/common/ref/`, which was stored with the base URL, and the queried UNO IDL where the points are replaced with slashes.

Figure 20: Hyperlink to the internet API

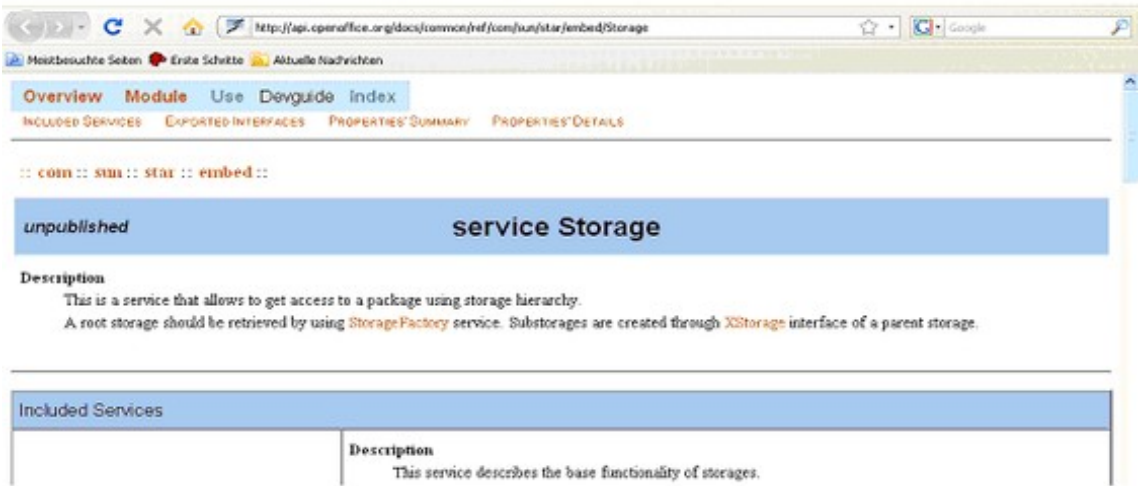


Figure 21: Open Office API `service Storage`

The next figure displays a hyperlink leading to the `service` storage of the local Open Office API. The hyperlink consists of the location of the Open Office SDK "`C:/Programme/OpenOffice.org_2.4_SDK`", the text `docs/common/ref`, which is added to the Open Office SDK location URL, and the queried UNO IDL where the points are replaced again with slashes.

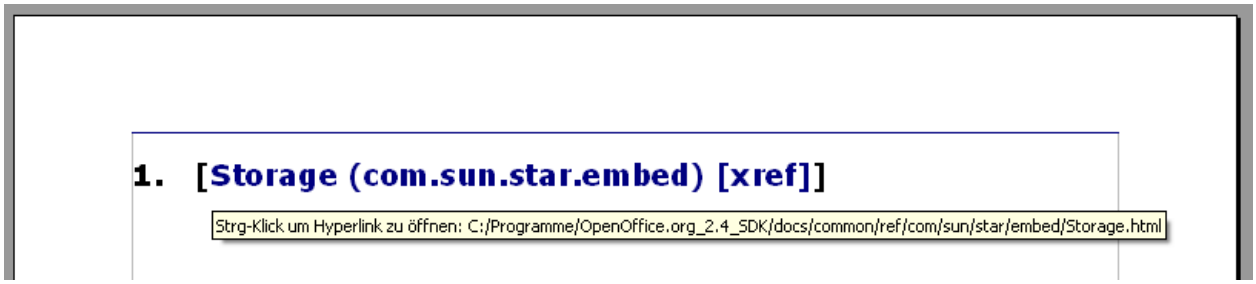


Figure 22: Hyperlink to the local Open Office API

After building the normal hyperlinks, the `xref` hyperlinks are created. They are only generated when `xref` exists for the UNO IDL. The `xref` is a list of elements which are contained in the requested UNO IDL. These elements can be other `services` which are included, `singletons` which instantiate the `service` and references to the Open Office Developers Guide.

The `xref` hyperlink contains the base URL and the originally created URL. Only the ending `"-xref.html"` is different to the other hyperlinks. After building the `xref` hyperlink the text “[xref]” is inserted into the Writer document after the headline.

The next two figures show the `xref` hyperlink to the `service Storage` and how the target of the `xref` link looks like. The other elements are the same as in the other hyperlinks. The target of this hyperlink is the Open Office API with all listed usages of the UNO IDL, which is shown in the second figure. Here are all `services` which include this `service` listed and the references to the Developers Guide are mentioned. Furthermore all `singletons`, which instantiate this `service`, are listed too. In the second figure the `service BaseStorage` is included in the `service Storage` and `FileSystemStorage`.

Figure 23: Hyperlink to the `xref` in the internet API

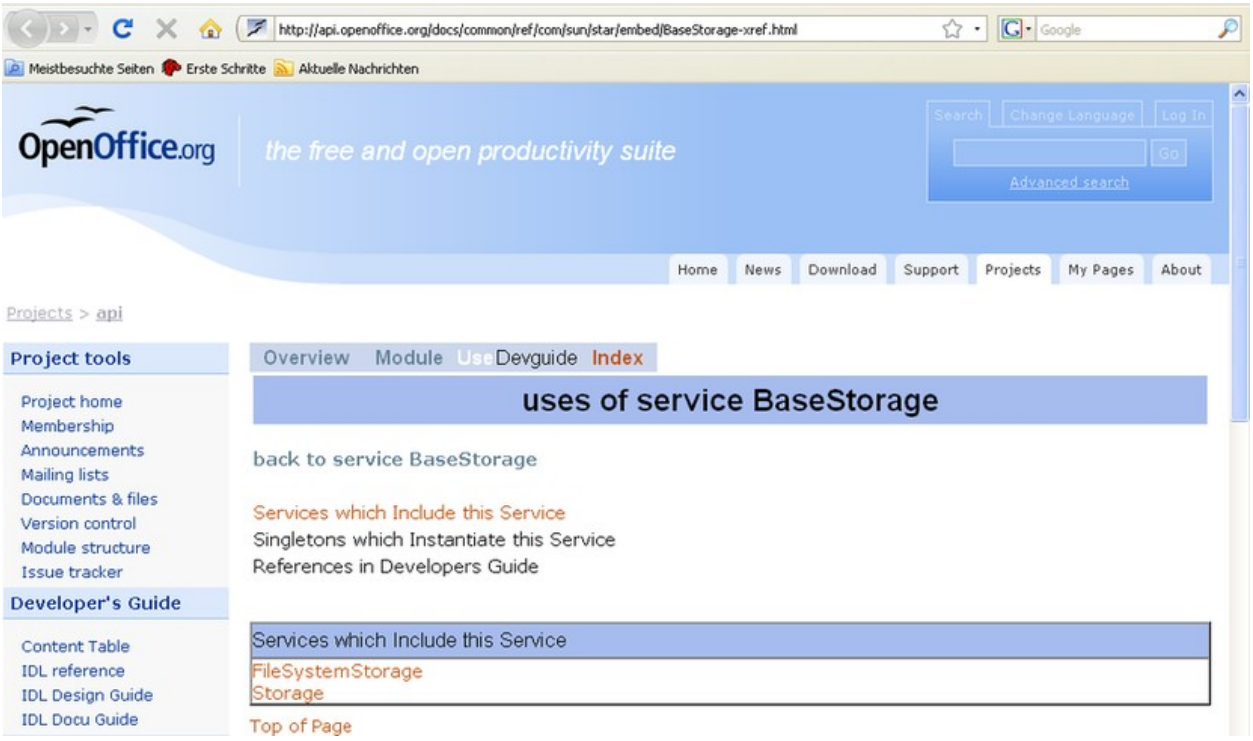


Figure 24:Open Office API - uses of service `BaseStorage`

3.1.10 Table ordered by Types

After the first table with the UNO IDL members in alphabetical order, a second table is added to the sWriter document. In this table the UNO IDL members are again added but they are ordered by UNO type names. So it can be easily seen which member belongs to which UNO type. In this table also additional information to each UNO type member is displayed. This table provides more information than the first one. The information is shown in different ways depending on the UNO type it belongs to.

First it has to be checked, if the UNO IDL has members. This can be done with the command `isA()` and the query `.uno_definitionWithMembers` from the `UNO_ReflectionUtil.cls`. With the command `isA` an instance of the method class can be queried. Therefore the `method` which should be included in the class has to be passed to the `isA` command. If the `method` `.uno_definitionWithMembers` is no `method` of the `refObj` class, the Nil object is returned [AFHMMPI]. The `refObj` is the object which contains the UNO IDL. If no Nil object is returned, a relation with the items associated with the UNO type names is created. After that a table using the `"com.sun.star.text.TextTable"` `service` is instantiated [OOOAPIae]. The table rows are calculated with the numbers of items of the member relation, the items of the unique types relation plus one for the headline. How many columns are needed depends on the result, if `UNO_CONSTANTS` and `UNO_ENUM` are displayed. If these two UNO types are added to the table, two columns are needed because additional information to the UNO IDL names is displayed. Otherwise only one column is initialized for the table. To find out if a string is similar to a word, the command `equals` can be used. This returns `.true` if both strings are equal [AFHMMPo].

```
    -- has members show them ordered in a table
if refObj~isA(.uno_definitionWithMembers) then
do
      -- get a relation where the items are associated with the UNO type names
     memberRel=makeUNOTypeRel(refObj)
     uniqueTypes=getUniqueItemTypes(memberRel)

      /* create the table */
     xTextTable = xDMsf~createInstance("com.sun.star.text.TextTable")~XTextTable

     nrRows=memberRel~items+uniqueTypes~items+1
     if refObj~unoTypeName~equals("UNO_CONSTANTS") | refObj~unoTypeName~equals("UNO_ENUM") then
     do
            xTextTable~initialize(nrRows, 2)       -- initialize the table
     end
     else
     do
            -- add one more row per UNO type (for heading) and one for title cell
            xTextTable~initialize(nrRows, 1)       -- initialize the table
```

```
    end
end
```

Code 8: Table ordered by types

The title of the first cell is set to “Name”. If the UNO type name is `UNO_CONSTANTS` or `UNO_ENUM`, for the second column the title “Value” is set too. Here again the command `equals` is used to find out if the two strings are similar [AFHMMPo]. To insert the title the `setCellText` routine is called. Then all elements are added to the table. In this table the elements are listed in sections. These sections are the UNO type names. So for example all elements with type name `UNO_SERVICE` are listed and then the elements with `UNO_TYPEDEF` follow. Within these sections the elements are alphabetically ordered. To find out which element belongs to the needed type name, the command `hasIndex` is used. This command returns the value one, which stands for true, if the array contains an item associated with the index given to the command. In the code below the UNO type is given as index [AFHMMPm]. In this code cut it is queried, if the given index is not included in the array, because then the loop should `iterate` [AFHMMPn]. In this loop the UNO type name is highlighted with a grey background which is set with the `setCellBackground` routine. After adding the UNO type name all elements of this type are inserted and then the next UNO type name with its members is added.

```
/* insert TextTable in the Text */
xText~insertTextContent(xTextCursor, xTextTable, .false)

call setCellText xTextCursor,"A1", "Name", xTextTable, 10, "BOLD"
if refObj~unoTypeName~equals("UNO_CONSTANTS") | refObj~unoTypeName~equals("UNO_ENUM") then
do
    call setCellText xTextCursor,"B1", "Value", xTextTable, 10, "BOLD"
end
row=2
do type over uniqueTypes      -- iterate over all found types
    if memberRel~hasIndex(type)=.false then iterate

call setCellText xTextCursor, "A"row, type":", xTextTable, 9, "BOLD"
call setCellBackground "A"row, xTextTable
```

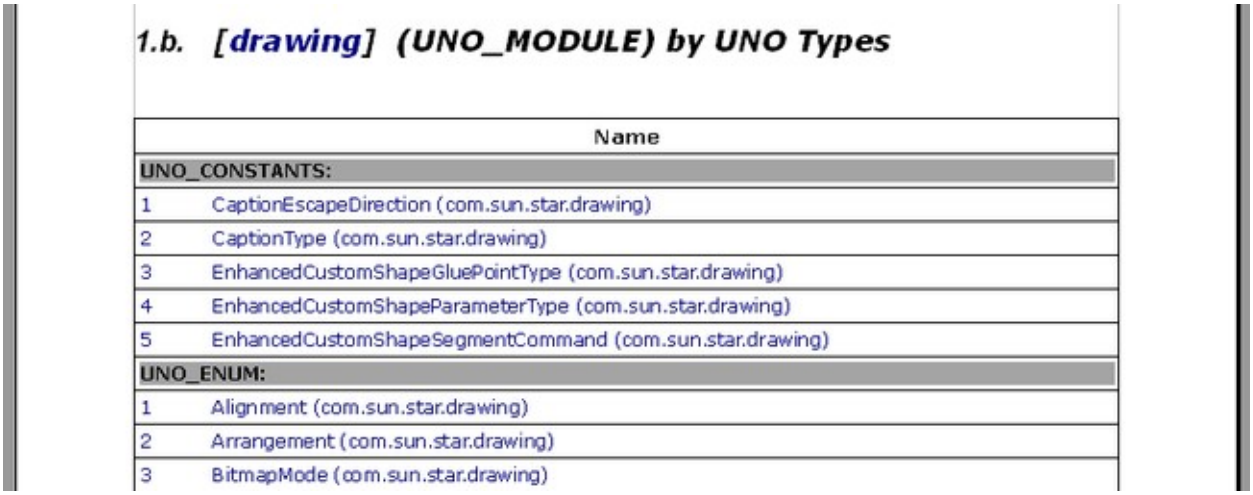
Code 9: Add UNO IDL members to table

The routine `setCellBackground` needs two arguments: the cell and the table. First in this routine the text cursor of the selected cell is created with the `method createCursorByCellName`. This `method` gets the chosen cell as argument [OOOAPIag]. Then the `property BackColor` of the cell is set to grey. This color is saved in the `.scholz` directory at the begin of the program [OOOAPIaa].

```
  -- routine to set the background color of a table cell
::routine setCellBackground
  use arg cell, xTextTable
  xTextTableCursor = xTextTable~createCursorByCellName(cell)
  xTextTableCursor~XPropertySet~setPropertyValue("BackColor", .scholz~backColor)
  return
```

Code 10: Set table cell background

The following figure shows a table with an UNO type and its members. The cell UNO type name is colored to be highlighted. The elements are shown after the type name they belong to and within they are ordered alphabetically and numbered.



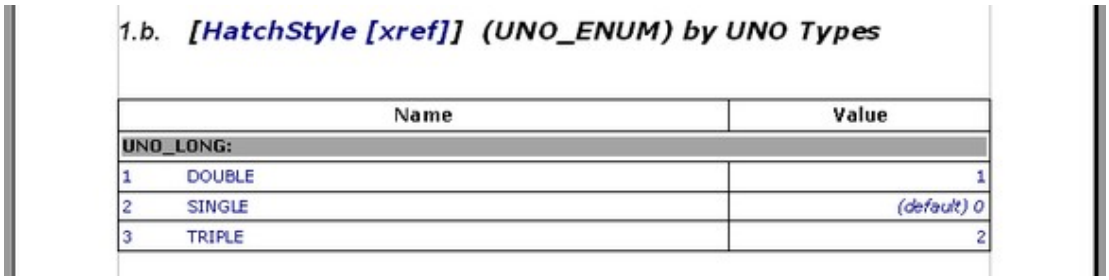
1.b. [drawing] (UNO_MODULE) by UNO Types

	Name
UNO_CONSTANTS:	
1	CaptionEscapeDirection (com.sun.star.drawing)
2	CaptionType (com.sun.star.drawing)
3	EnhancedCustomShapeGluePointType (com.sun.star.drawing)
4	EnhancedCustomShapeParameterType (com.sun.star.drawing)
5	EnhancedCustomShapeSegmentCommand (com.sun.star.drawing)
UNO_ENUM:	
1	Alignment (com.sun.star.drawing)
2	Arrangement (com.sun.star.drawing)
3	BitmapMode (com.sun.star.drawing)

Figure 25: `CreateTableByTypes` UNO type name background color

As described before, if the entered types are `UNO_CONSTANTS` or `UNO_ENUM`, then a second column is added to the table for additional information. For the insertion of the type name the columns should be merged so that the correlation of the UNO IDL members to the UNO type name is clear. How table cells could be merged can be read in the chapter 4.1.

The following figure shows a table where the columns of the second table row are merged. The other rows have two columns and are not affected by the merge of the columns in the second row.



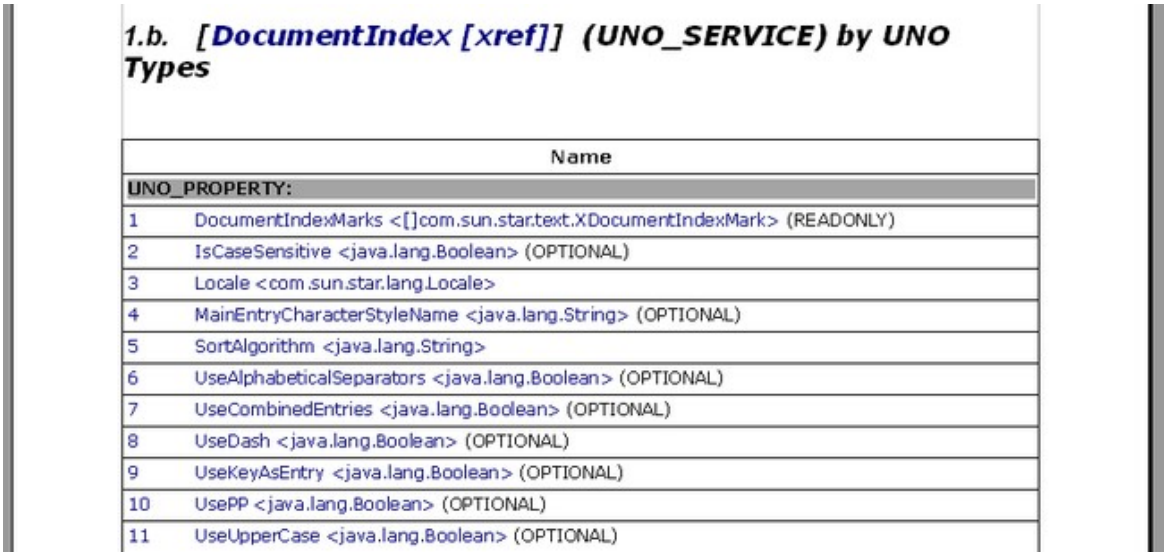
1.b. [HatchStyle [xref]] (UNO_ENUM) by UNO Types

	Name	Value
UNO_LONG:		
1	DOUBLE	1
2	SINGLE	(default) 0
3	TRIPLE	2

Figure 26: Table with merged columns

For the insertion of the UNO IDL members the different UNO types have to be distinguished, because different information is displayed to the different UNO types. If an `UNO_PROPERTY` is added to the table, it is checked, if there are modifiers provided. If modifiers are available, then they are inserted into the table column as well after the member name between two brackets. If the `UNO_PROPERTY` does not contain modifiers, only the member name is shown.

The next figure shows the different illustrations of the `UNO_PROPERTY` in the table sorted by UNO types. It displays the `UNO_PROPERTY` with attached modifiers and two times only the name of the properties is shown, because no modifiers are available for them.



1.b. [DocumentIndex [xref]] (UNO_SERVICE) by UNO Types

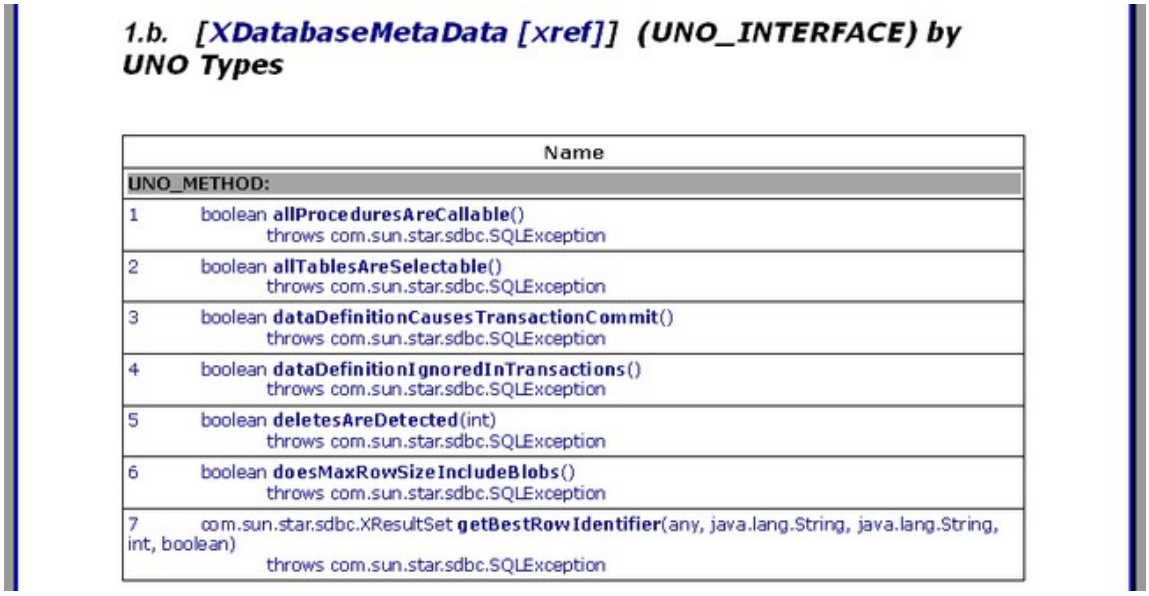
	Name
UNO_PROPERTY:	
1	DocumentIndexMarks <[]com.sun.star.text.XDocumentIndexMark> (READONLY)
2	IsCaseSensitive <java.lang.Boolean> (OPTIONAL)
3	Locale <com.sun.star.lang.Locale>
4	MainEntryCharacterStyleName <java.lang.String> (OPTIONAL)
5	SortAlgorithm <java.lang.String>
6	UseAlphabeticalSeparators <java.lang.Boolean> (OPTIONAL)
7	UseCombinedEntries <java.lang.Boolean> (OPTIONAL)
8	UseDash <java.lang.Boolean> (OPTIONAL)
9	UseKeyAsEntry <java.lang.Boolean> (OPTIONAL)
10	UsePP <java.lang.Boolean> (OPTIONAL)
11	UseUpperCase <java.lang.Boolean> (OPTIONAL)

Figure 27: `createTableByTypes` `UNO_PROPERTY` display modifiers

If elements of the UNO type `UNO_METHOD` are inserted into the table, the arguments and `exceptions` of these `methods` are listed. The member name should be highlighted and is

set to bold. This is important for the user to see the `method` name with one quick look. The `method` additional information is displayed as normal text not in bold.

The next figure shows the display of an `UNO_METHOD` with additional information. Only the name of the `methods` is marked bold.



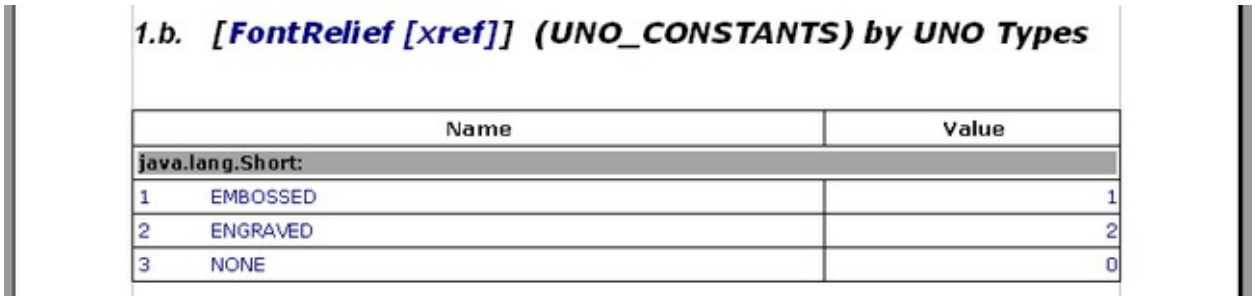
1.b. [XDatabaseMetaData [xref]] (UNO_INTERFACE) by UNO Types

Name
UNO_METHOD:
1 boolean **allProceduresAreCallable**() throws com.sun.star.sdbc.SQLException
2 boolean **allTablesAreSelectable**() throws com.sun.star.sdbc.SQLException
3 boolean **dataDefinitionCausesTransactionCommit**() throws com.sun.star.sdbc.SQLException
4 boolean **dataDefinitionIgnoredInTransactions**() throws com.sun.star.sdbc.SQLException
5 boolean **deletesAreDetected**(int) throws com.sun.star.sdbc.SQLException
6 boolean **doesMaxRowSizeIncludeBlobs**() throws com.sun.star.sdbc.SQLException
7 com.sun.star.sdbc.XResultSet **getBestRowIdentifier**(any, java.lang.String, java.lang.String, int, boolean) throws com.sun.star.sdbc.SQLException

Figure 28: `createTableByTypes` `UNO_METHOD` additional information

If an `UNO_CONSTANTS` or `UNO_ENUM` is added to the table, the value is displayed in a second column. This column is created at the begin of the table. The width of the two columns is adjusted so that the first column is larger than the second one. How the column width can be changed can be found in the chapter 4.4. An `UNO_ENUM` can have a default value. To find out the default value, the `method` `defaultValue` from the `UNO_ReflectionUtil.cls` is used. The default value is highlighted with the word (default) written in italic that it can be easily seen which value is used, if nothing is specified. The text within the value column is adjusted to the right. How to adjust a text within a table column is described in the chapter 4.2.

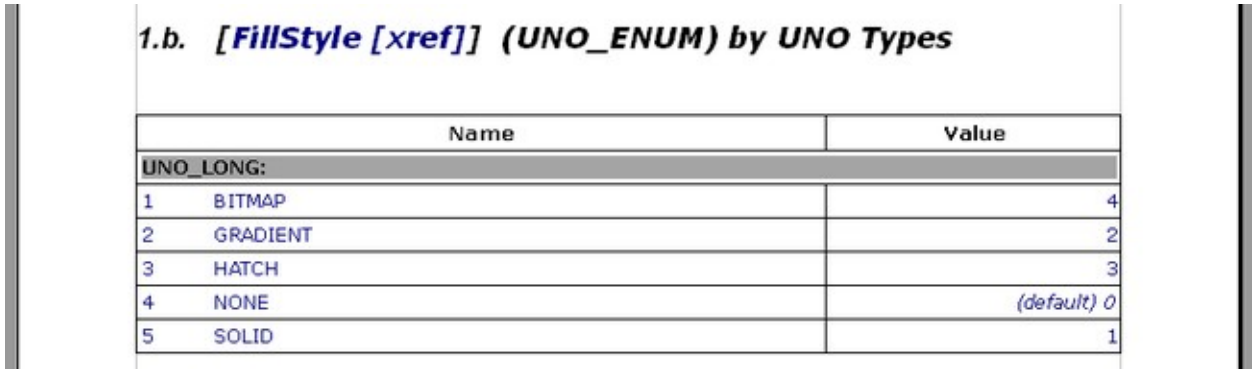
The following two figures illustrate a table with the elements of `UNO_CONSTANTS` and `UNO_ENUM`. A second column is added to the table and the values of each element are displayed. If the row entry is the default value of an `UNO_ENUM` the text (default) is set next to the value so the user can see which of the values is used if nothing is specified.



1.b. [FontRelief [xref]] (UNO_CONSTANTS) by UNO Types

	Name	Value
java.lang.Short:		
1	EMBOSSED	1
2	ENGRAVED	2
3	NONE	0

Figure 29: `createTableByTypes UNO_CONSTANTS`



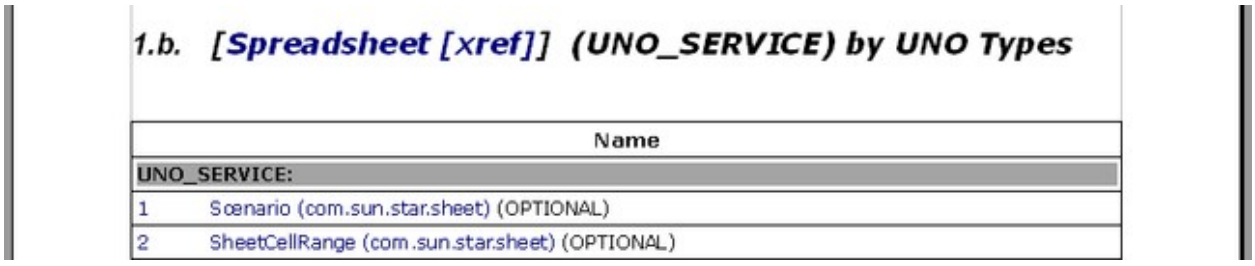
1.b. [FillStyle [xref]] (UNO_ENUM) by UNO Types

	Name	Value
UNO_LONG:		
1	BITMAP	4
2	GRADIENT	2
3	HATCH	3
4	NONE	*(default) 0*
5	SOLID	1

Figure 30:`createTableByTypes UNO_ENUM`

The last two UNO types of which additional information is provided in the table are the `UNO_SERVICE` and `UNO_INTERFACE`. If they are not mandatory, the text “(OPTIONAL)” is displayed after the name. To find out if they are optional the member name is parsed with the command `parse var` and `split` [AFHMMPq].

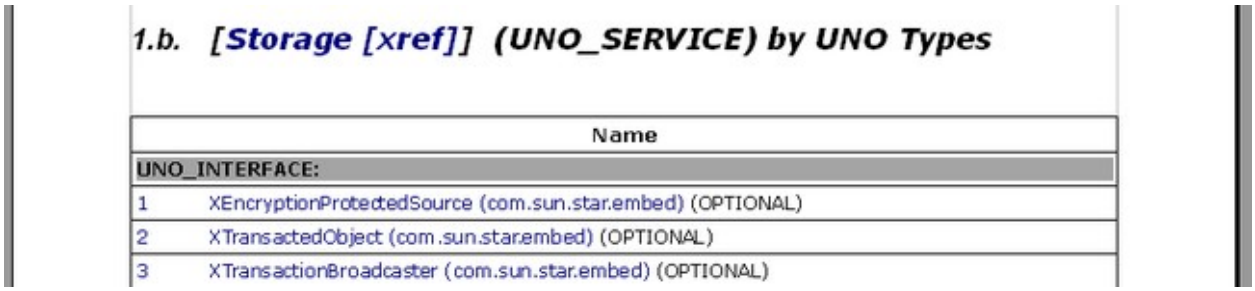
The following two figures show optional `services` and `interfaces` which are marked with the word “OPTIONAL”:



1.b. [Spreadsheet [xref]] (UNO_SERVICE) by UNO Types

	Name
UNO_SERVICE:	
1	Scenario (com.sun.star.sheet) (OPTIONAL)
2	SheetCellRange (com.sun.star.sheet) (OPTIONAL)

Figure 31: `createTableByTypes UNO_SERVICE`



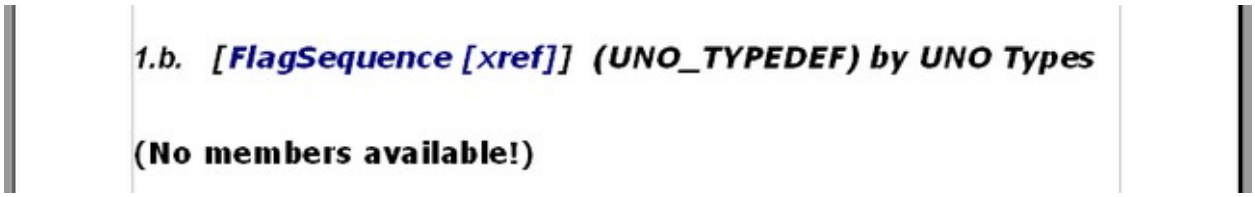
1.b. [Storage [xref]] (UNO_SERVICE) by UNO Types

	Name
UNO_INTERFACE:	
1	XEncryptionProtectedSource (com.sun.star.embed) (OPTIONAL)
2	XTransactedObject (com.sun.star.embed) (OPTIONAL)
3	XTransactionBroadcaster (com.sun.star.embed) (OPTIONAL)

Figure 32: `createTableByTypes UNO_INTERFACE`

If the UNO IDL has no members, the message “(No members available!)” is displayed instead of a table.

The following figure shows the display of the `typedef FlagSequence`, if an UNO IDL without members is passed to the routine.



1.b. [FlagSequence [xref]] (UNO_TYPEDEF) by UNO Types

(No members available!)

Figure 33: UNO IDL without members display table by types

After creating the table with the IDL members ordered by type, a paragraph break is set. If the table has more than three rows, a page break is set so that the graphic, which is displayed after the table, has enough space.

The next two figures show the difference between the graphical display directly after the table and after a page break. A page break is set if the UNO IDL has more than three members, as the second figure shows.

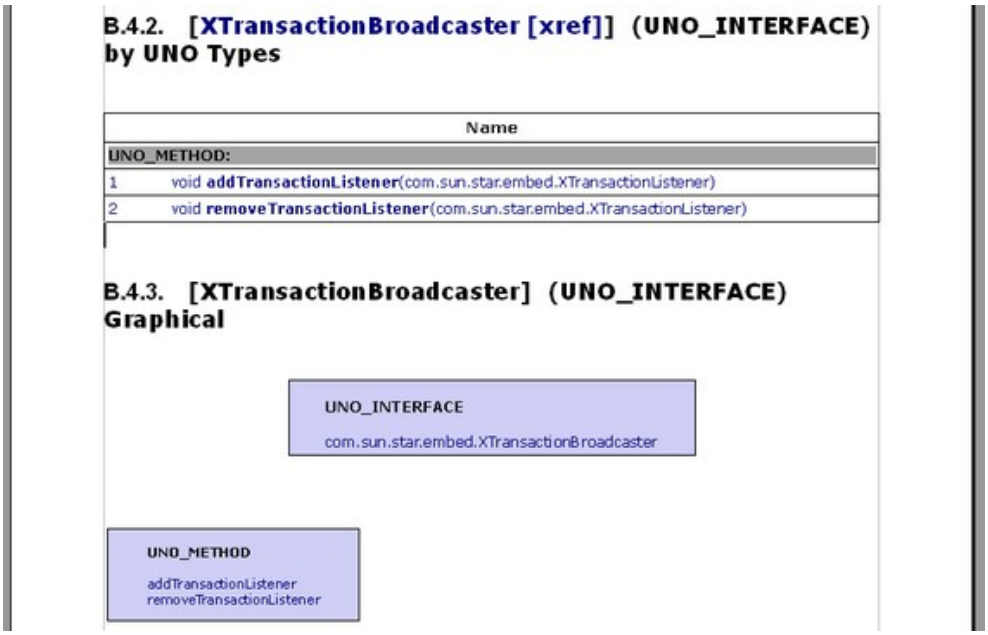


Figure 34: Graphical display directly after table

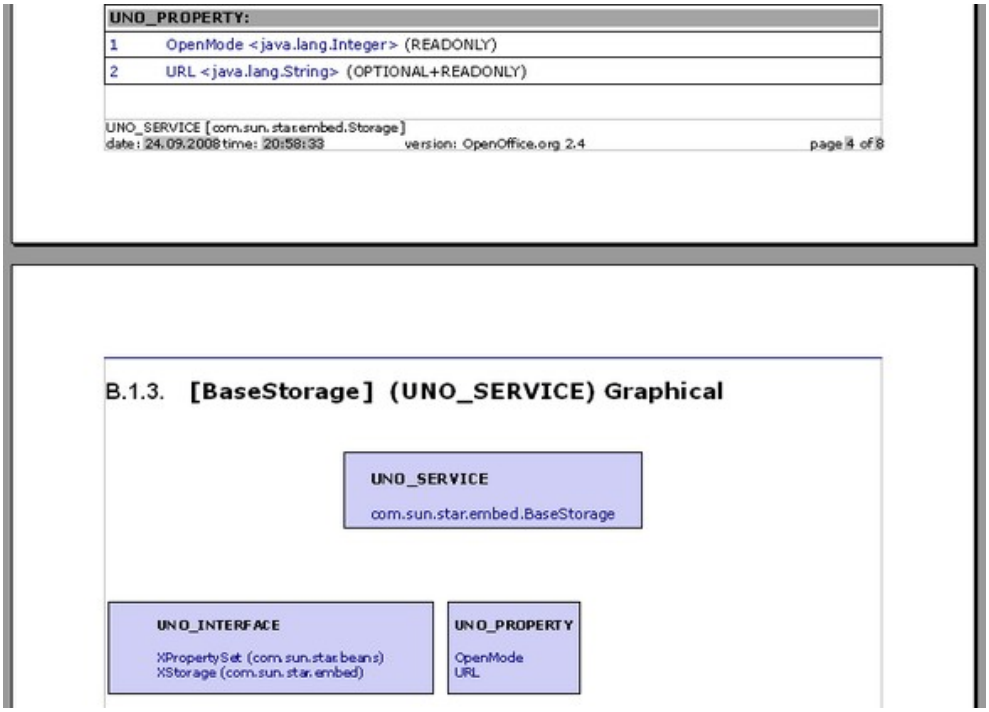


Figure 35: Graphical display after page break

3.1.11 Graphic Overview

After the two tables with the UNO IDL members the graphical display of these members is added to the sWriter document. The graphic consists of shapes with the UNO IDL

members as content. For each UNO type name a shape is created and the UNO IDL members belonging to it are inserted alphabetically into the shape. The chosen shape for the graphical display is a rectangle. In the first rectangle the UNO IDL name is inserted. The position of the first rectangle is always in the center of the document that it can be seen which UNO IDL is displayed in the graphic.

The following picture illustrates the first rectangle with the queried IDL and the UNO type name which it belongs to.

Figure 36: Rectangle with the queried IDL

After inserting the first rectangle, the text which should be added into the other rectangles is processed. Therefore a relation is created where the items are associated with the UNO type names. For `UNO_PROPERTY`, `UNO_ATTRIBUTE` and `UNO_MODULE` only the name is saved, for the others the fully qualified name. After each element a line break is inserted and then the next element is attached. Then the shape with the respective UNO elements is inserted into the Writer document.

If no elements are included in the UNO IDL the message “(No members available!)” is displayed. If the routine is processed, the position, width and height of the rectangle are set to the start value.

The next figure shows how the graphic looks like if no members are available for the UNO IDL. Only the UNO IDL name and its UNO type name are displayed and the message “(No members available!)” is shown.

Figure 37: Graphical display no members available

To create a rectangle some elements have to be defined:

- width
- height
- position.

These three elements have to be calculated automatically because each UNO IDL has a different amount of members and the member's names length varies from member to member. Another point is, that the members should be ordered by UNO type names so that for every type name one rectangle is created. Within the rectangle the members are sorted alphabetically.

First the width of the rectangle is calculated. Therefore the maximum length of the UNO IDL elements has to be evaluated with the command `length`. This command returns the length of the received string [AFHMMPt]. The maximum can be calculated with the command `max` which returns the largest number from the received arguments [AFHMMPu]. After calculating the maximum width it has to be multiplied to get the width of the rectangle. The height is calculated by counting the lines of the string with the UNO IDL members with the command `countStr`. This command counts how often the received argument is included in the string [AFHMMPv]. The received value is also multiplied to get the needed height.

After calculating the height and the width, the rectangle can be created by instantiating the `"com.sun.star.drawing.RectangleShape"` `service` [OOOAPIak]. The size is set with the `method` `setSize` of the `interface` `XShape` [OOOAPIal]. For the shape the anchor

`AT_PARAGRAPH` is set [OOOAPIam]. The text within the shape is adjusted to the top of the rectangle with the `enum "com.sun.star.drawing.TextVerticalAdjust"` so that the text always starts at the top [OOOAPIan] [OOOD08f].

```
   -- create and insert a shape object
shape = xDMsf~createInstance("com.sun.star.drawing.RectangleShape")~xShape
shape~setSize(.awtSize~new(rectWidth,rectHeight))
   -- set the anchor at_paragraph so the rectangle can be moved to another position
shape~xPropertySet~setPropertyValue("AnchorType", .anchorType~"AT_PARAGRAPH")
shape~xPropertySet~setPropertyValue("TextVerticalAdjust",                    -
      bsf.getConstant("com.sun.star.drawing.TextVerticalAdjust", "TOP"))
```

Code 11: Create shape

Each UNO IDL member gets a hyperlink added, which links to the Open Office API. How the link is composed can be read in the 3.1.9 chapter. The hyperlinked text is marked blue that it is recognized as a hyperlink. How hyperlinks can be set within a rectangle is described in the chapter 4.7.

The next figure shows a rectangle with all `UNO_INTERFACES` of an UNO IDL. The inserted `interfaces` have a hyperlink which links to the Open Office API.

Figure 38: Rectangle with `UNO_INTERFACES`

The first rectangle with the information which UNO IDL is displayed in the graphic, is always placed in the middle of the document. For the other rectangles the position has to be calculated, depending on the rectangle size, how many rectangles are inserted before and, if there is space on the current page left. If the width of the created rectangle is not larger than the place left of the document width, the rectangle is added next to the other rectangles.

The following figure shows a cut of the graphical display of the `module "com.sun.star.sdbc"`. The rectangles are arranged next to another.

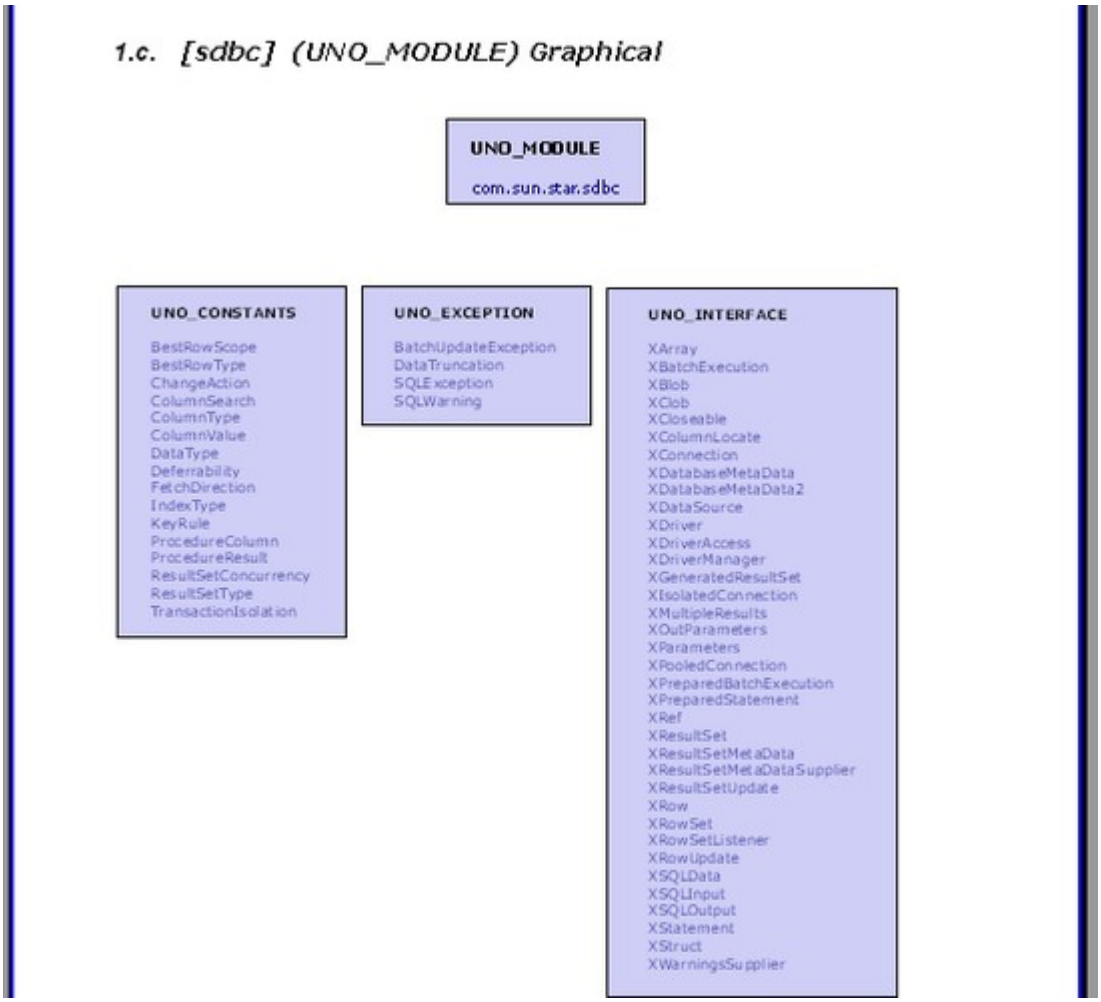


Figure 39: Division of the shapes with the IDL elements example 1 – arrange rectangles next to another

If the width of the document is fully used, then the next rectangle has to be set under the other rectangles. Every time when there is no place left next to a rectangle, the rectangle is set below the other rectangles. There is only one exception when the height of the page is full too. Then a page break is set before the next rectangle is inserted to the document.

The next figure illustrates the graphical display of the `service "com.sun.star.embed.Storage".` The rectangles are arranged next and below the other shapes.

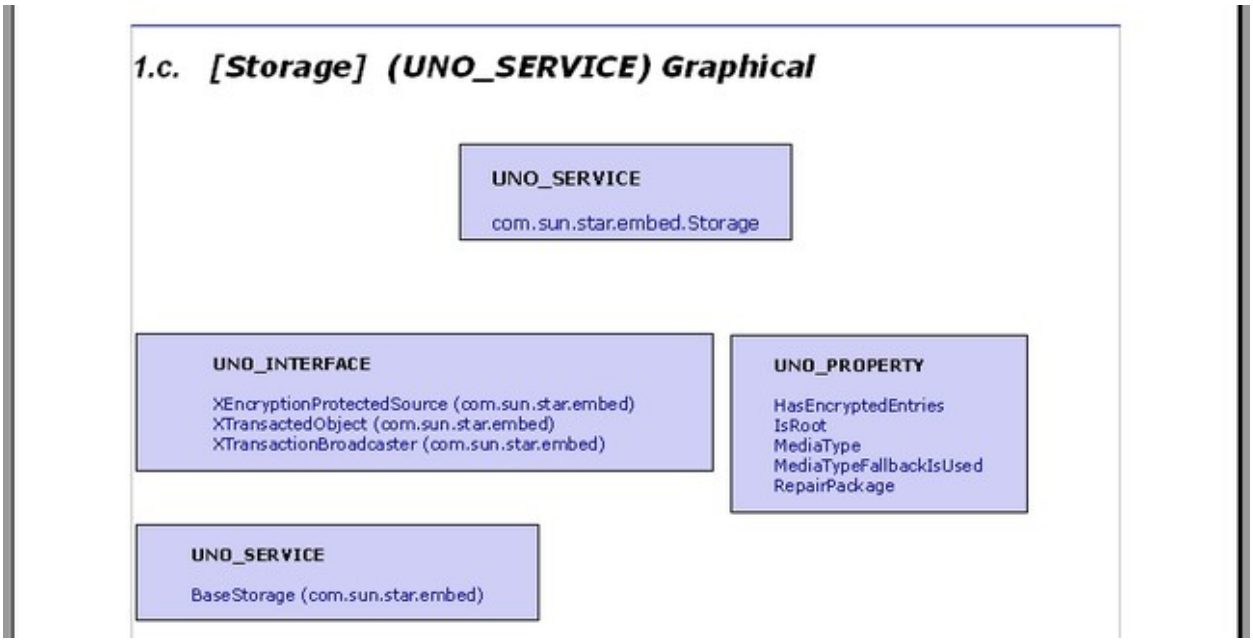


Figure 40: Division of the shapes with the IDL elements example 2 – arrange rectangles below another

- The following picture shows a cut of graphical display of the `module` "`com.sun.star.util`". A page break is needed because there is not enough space for the high rectangle on the first side.

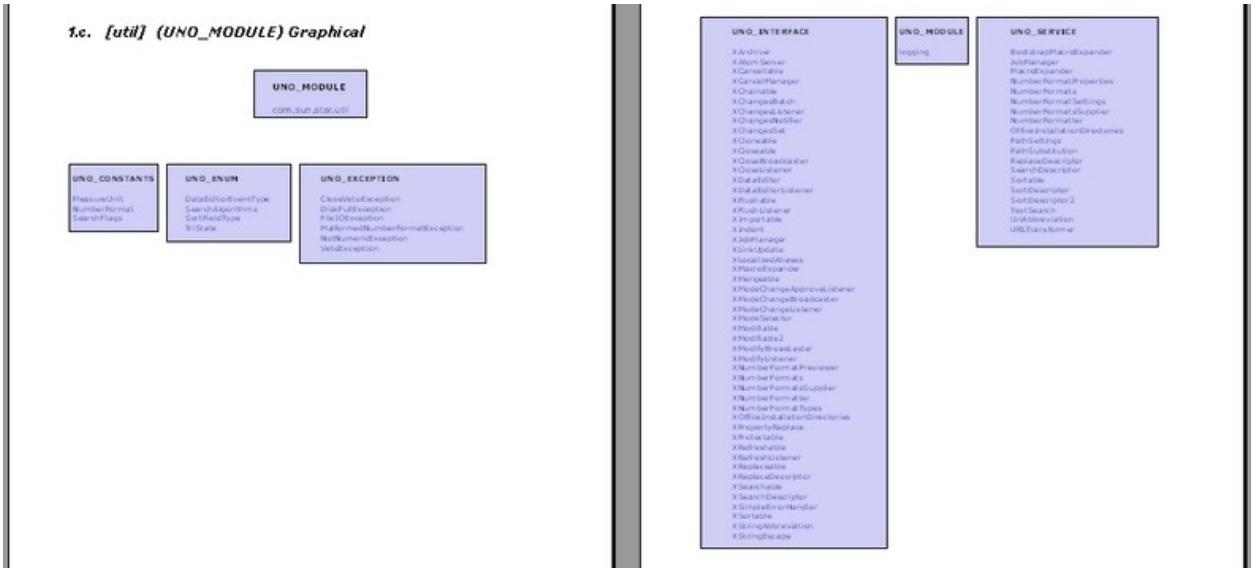


Figure 41: Division of the shapes with the IDL elements example 3 – page break

The last possibility of inserting a rectangle is when the rectangle is too high for one page. Then it has to be split into two or three rectangles. If there is enough space left the split

rectangle is added next to the other ones, otherwise a new page is added to the document and the divided rectangles are inserted there.

The following figure displays the split of the elements of the UNO type `UNO_SERVICE` of the `module` `"com.sun.star.text"` because there are too many members for one rectangle.

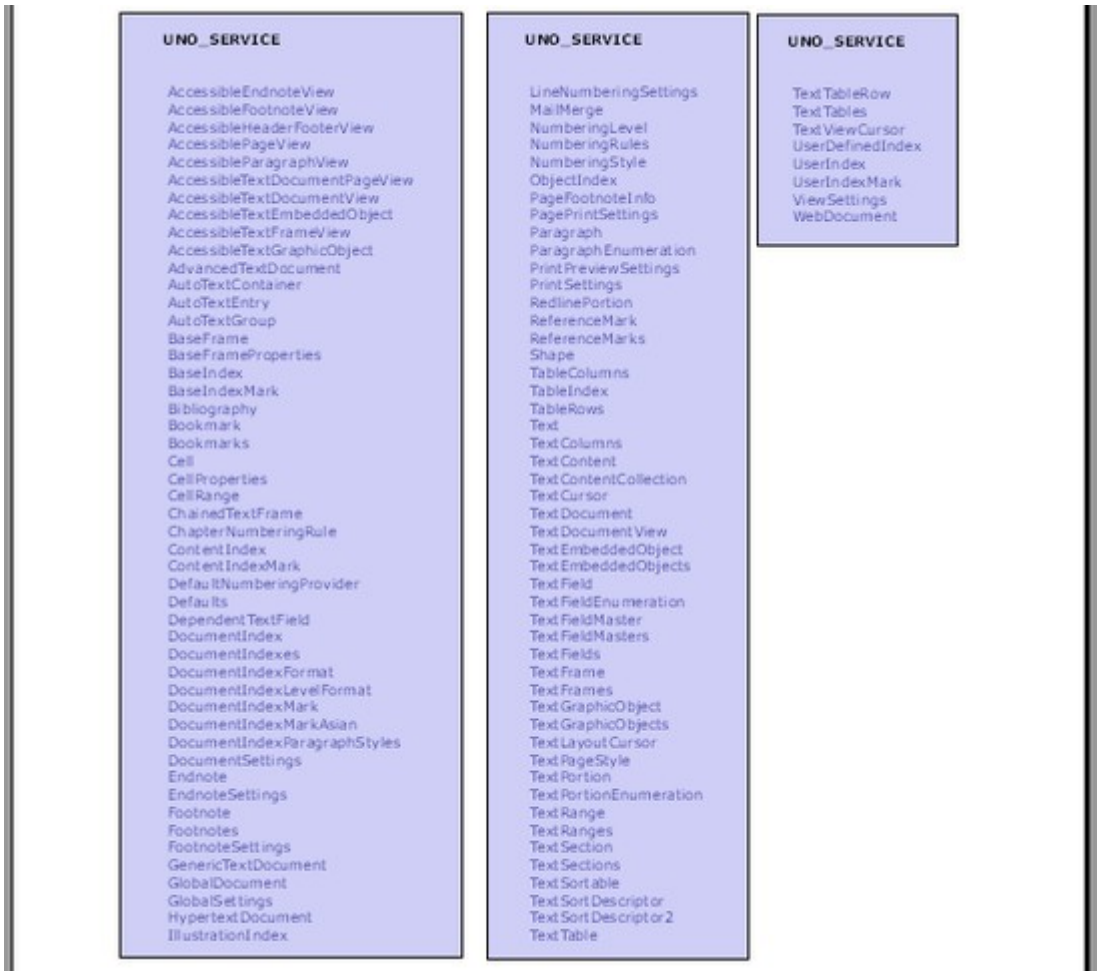


Figure 42: Split of too high shapes

3.1.12 Second Layer

The user has the possibility to display the second layer. In this case the UNO IDL members included in the primary request are processed and their members are displayed in the document too. This gives the user a better overview about the connections between the different UNO IDL's.

If a second layer is shown, the message “Now processing all the member definitions:“ concatenated with the member IDL names are shown in the command window. Before all member definitions are added to the sWriter document a page break is set and a new section headline is added for the user to see where the display of the second layer starts. For each processed IDL member, two tables and one graphic is created that the user always gets the same information to each member, regardless which layer or which IDL is displayed.

The next two figures show the query process of the `service "com.sun.star.embed.Storage"` with the second layer. In the command window the text output during the query is shown and in the Writer document the display of the UNO IDL and its members.

Figure 43: Command window output of the query `"com.sun.star.embed.Storage"` with second layer

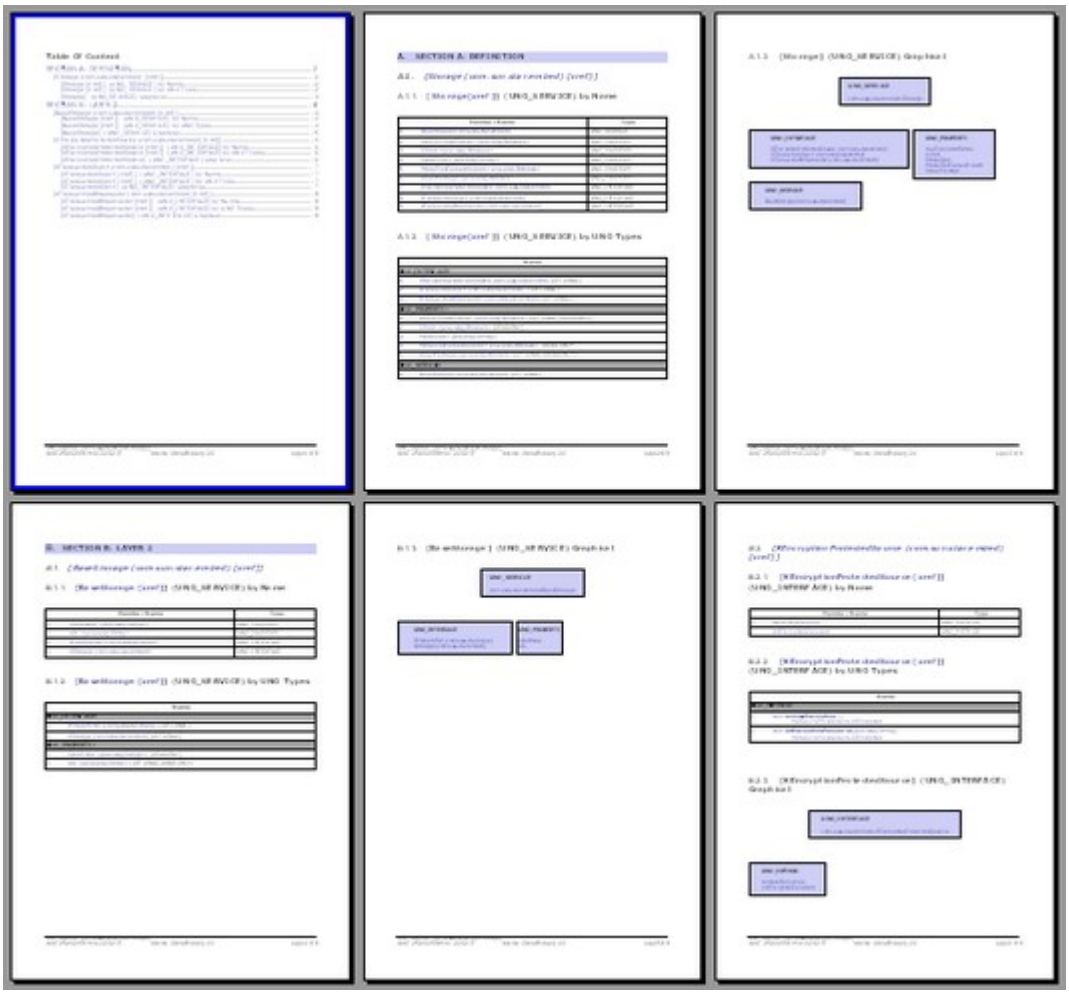


Figure 44: Created Writer document of the `service Storage` with second layer

The following figures show requests about different UNO IDL's and their output. The created Writer document always looks different but they have all the same structure. On the first page there is the table of content. Then two tables are displayed. The first table contains the UNO IDL members sorted alphabetically. In the second table the elements are sorted after the different UNO type names and within the same type name they are sorted alphabetically. After the second table the UNO IDL members are displayed graphically. For each UNO type one rectangle is created and the type members are added into this shape. The layout of the Writer document differs depending on the amount of UNO IDL members and how many layers are shown.

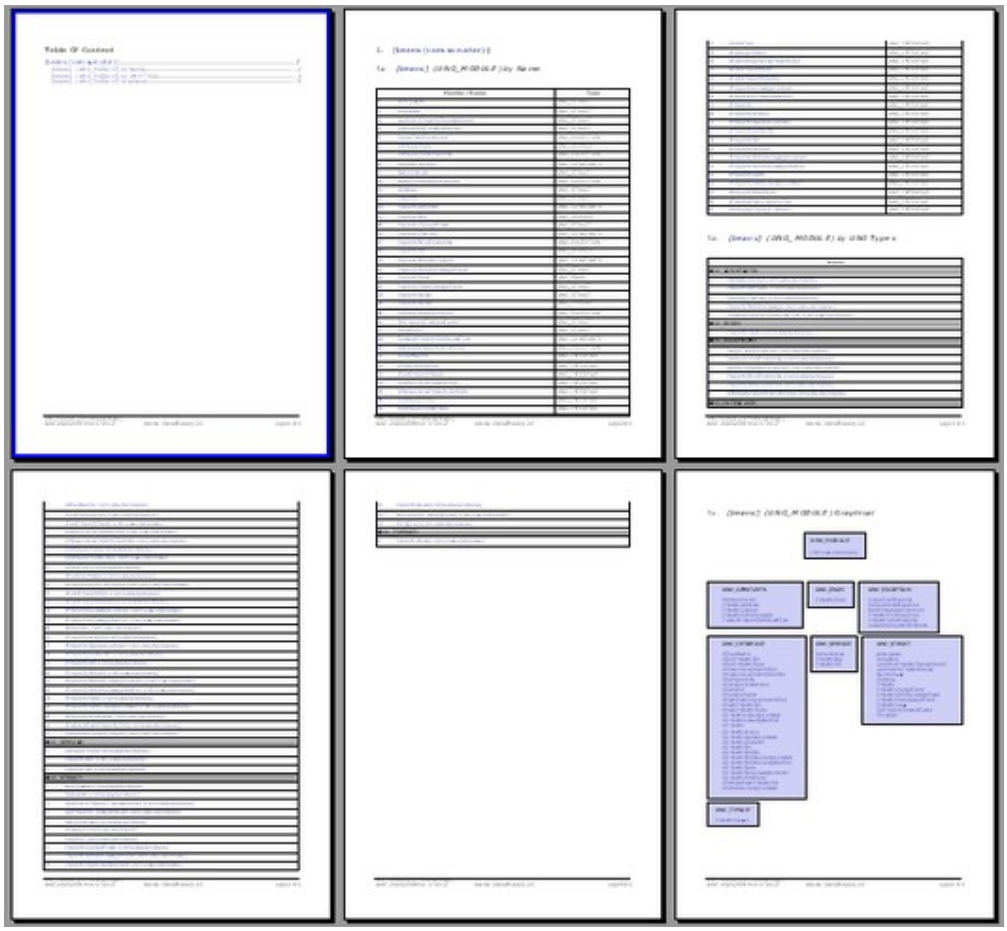


Figure 45: API-Viewer created Writer document – example 1

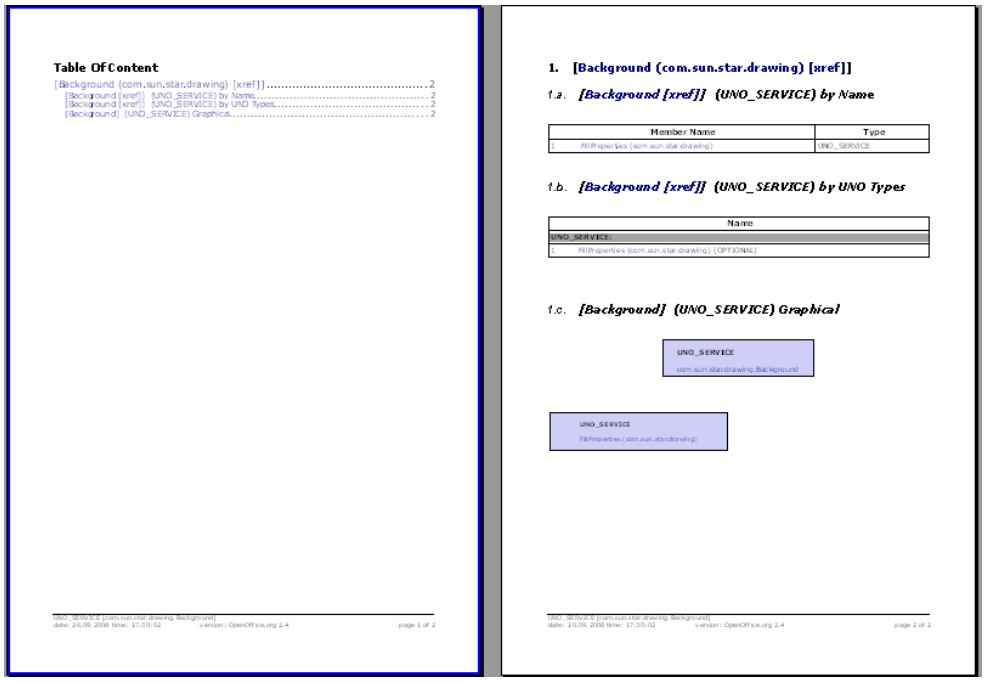


Figure 46: API-Viewer created Writer document – example 2

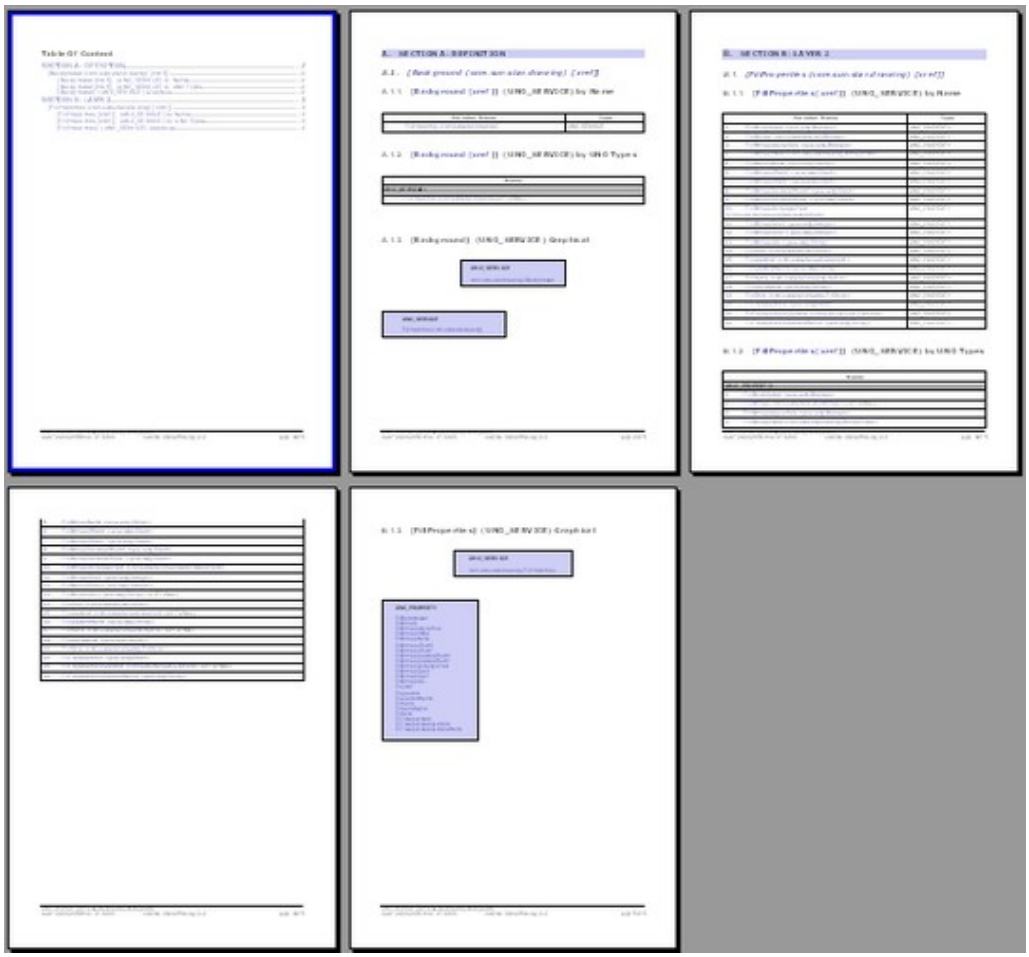


Figure 47: API-Viewer created Writer document – example 3

3.2 frontend.rex

`frontend.rex` is the graphical interface from the API-Viewer. It can be started through the command line with the following command: `rexx frontend.rex`.

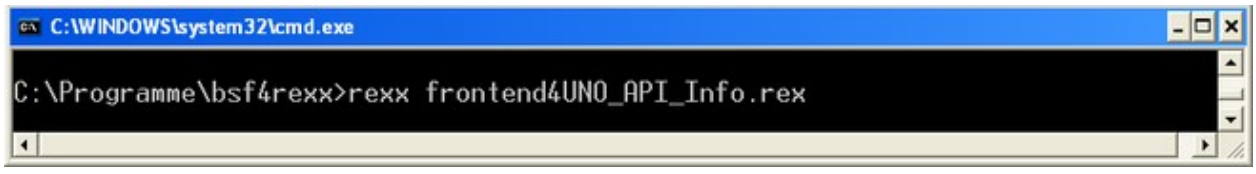


Figure 48: Starting `frontend.rex` from windows

After confirming with enter, the program starts and the following graphical interface is displayed on the monitor:

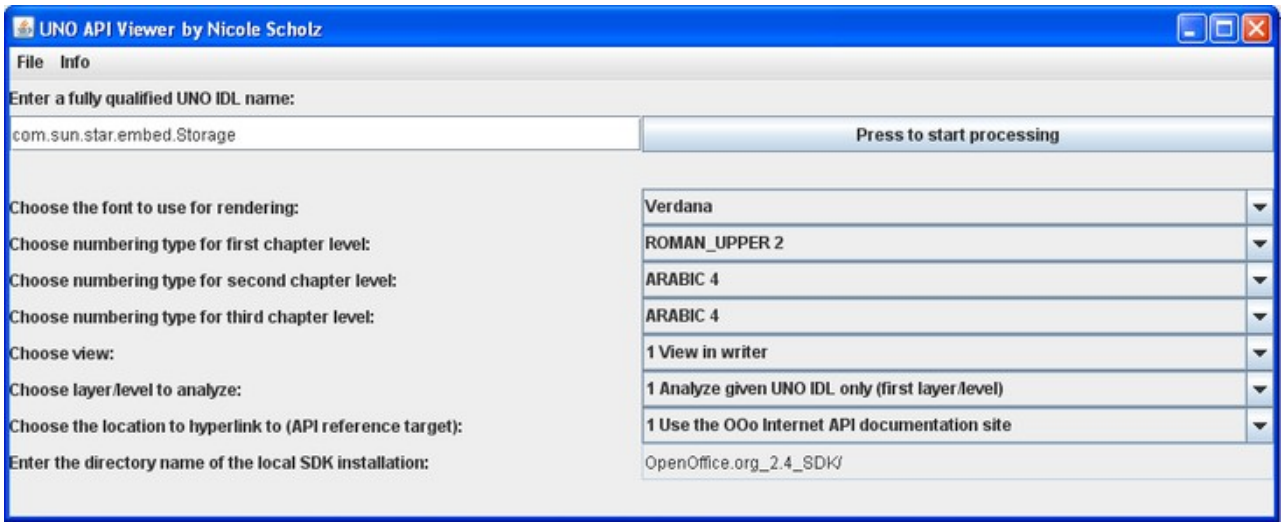


Figure 49: API Viewer graphical interface

`frontend.rex` is programmed with Java Swing so that it can be accessed from the Open Office side too. `frontend.rex` requires two elements to be executed:

```
::requires "bsf.cls"       -- get the Java support
::requires createApiInfo.rex   -- loads also "rgf_util2.rex"
```

Code 12: `frontend.rex` required elements

It needs the `bsf.cls` to get Java support of BSF4Rexx so that all elements Java provides can be used via ooRexx. It also requires `createApiInfo.rex`. These requirements have to be set after the normal program code before the routines section starts.

To create a program display with Java, some components are needed: a frame where the elements of the interface should be included, labels and the chosen layout style. Every Java class, which is used very often, can be imported with BSF4Rexx and can be accessed with one word and not the whole hierarchical class name has to be called every time. To import a Java class `bsf.import` has to be called. This `method` has two arguments the first argument is the Java class, which has to be imported, and the second is the variable with which the class is accessed. The following code imports two Java classes: `javax.swing.Jlabel` and `java.awt.Cursor`.

```
/* we need a swing-Label quite often, hence import it */
/* the ooRexx proxy class is named "swingLabel" */
call bsf.import "javax.swing.JLabel", "swingLabel"
call bsf.import "java.awt.Cursor", "awtCursor"
```

Code 13: `frontend.rex` import java classes

First a frame for the graphical display has to be created. It is possible to add a title to the frame. Here the title is set to “UNO API Viewer by Nicole Scholz”. To close the frame, an event listener is added to the frame. This event listener closes the window when the user is clicking the red marked cross on the right top of the frame.

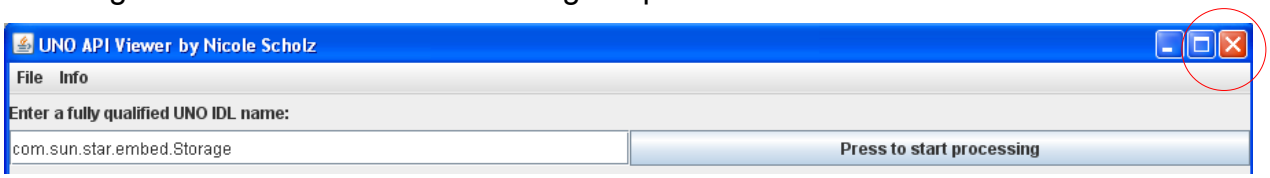


Figure 50: Closing `frontend.rex`

The following code shows how a frame is created and how the event listener is added to the frame. The frame is created with the `java.swing.Jframe` class [Java03a]. With the Rexx command `.bsf~new` a new Java class is created. The event listener is added to the frame with the message `~bsf.addEventListener`. The “~”-symbol is used to send messages to an object.

```
/*create the frame for the program*/
frame = .bsf~new("javax.swing.JFrame", "UNO API Viewer by Nicole Scholz")
frame~bsf.addEventListener('window', 'windowClosing', 'call BSF "exit"')
```

Code 14: Create program frame

Next a menu is added to the frame. The menu bar consists of two menu items - the “File” and “Information” section - which are added and grouped to the menu bar. The menu bar itself is added to the program frame. The program code below shows how the menu bar and the menus are created. The Java classes are created with the `.bsf~new` command again and the menus are added to the menu bar with `~add()`. This command always contains the object which should be added to the other object between the brackets [Java03b][Java03c].

```
/*create a menu bar*/
menubar= .bsf~new("javax.swing.JMenuBar")
menu= .bsf~new("javax.swing.JMenu", "File")
menubar~add(menu) -add the first menu to the menu bar
menu2= .bsf~new("javax.swing.JMenu", "Info")
menubar~add(menu2) -add the second menu to the menu bar
frame~setJMenuBar(menubar) -add the menu bar to the frame
```

Code 15: Create a menu bar

The following figure shows the menu of `frontend.rex`.

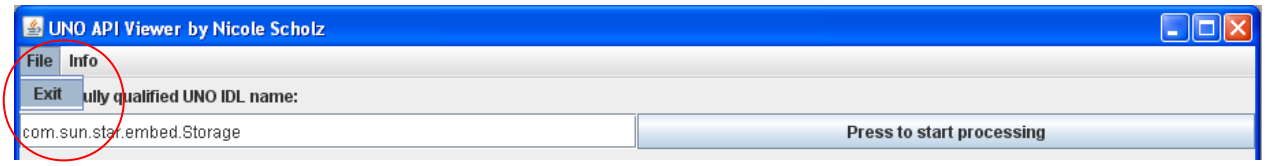


Figure 51: `frontend.rex`: menu

The first menu contains the exit item. With clicking on the menu item exit, the program is closed. The following program code shows that an event listener is added to the menu item to close the window when the user clicks on the exit menu entry [Java03d].

```
/*menu exit*/
menuitem= .bsf~new("javax.swing.JMenuItem", "Exit")
menuitem~bsf.addEventListener('action', '', '.bsf~bsf.exit')
menu~add(menuitem)
```

Code 16: Menu item exit

If the information menu is opened and the information entry is clicked, another window is opened with some short information about the features of the API-Viewer. The code below shows the creation of the information menu entry and the event listener to open a new window [Java03d]. To achieve this, the routine `newWindow` has to be called.

```
/*menu information*/
menuinfoitem= .bsf~new("javax.swing.JMenuItem", "Information")
menuinfoitem~bsf.addEventListener('action', '',  'call newWindow')
menu2~add(menuinfoitem)
```

Code 17: Add menu item information

In Object Rexx a routine has to be defined after the “required programs” section. The following code shows the position of the `newWindow` routine and the creation of a new frame for the information window. Here a local variable is used for the second frame. This variable can be accessed in the whole program without passing it to the called methods. It is created with the following command: `.local~variablename`. To access the variable, its name has to be written with a point in front of the name like `.variablename` [AFHMMPb].

```
::requires "bsf.cls"       -- get the Java support
::requires createApiInfo.rex   -- loads also "rgf_util2.rex"

  -- routine for creating the information window
::routine newWindow
  .local~frame1 = .bsf~new("javax.swing.JFrame", "About this Program")
```

Code 18: Create new window

The text in the information window is created with HTML. This allows to create numerations and to format the text. The code below displays the whole information text formatted with line breaks which are defined in `HTML` with `<br>` and paragraph breaks which are created with `<p>`. Between the brackets `<h3>` the heading is given. The numeration is created with `<ul>` and the list points with `<li>`. The information text is written between the `HTML` commands.

```
type="text/html"      -- MIME type of text
txt="<h3>Program Information</h3>                                                    "  -
    "                                                                                "  -
    "This program queries the given, fully qualified UNO IDL string at runtime and   <br>  "  -
    "formats the results in an OpenOffice.org writer document, such that the user is <br>  "  -
    "able to learn about all of its documented capabilities. The UNO IDL definitions <br>  "  -
    "are hyperlinked either to the OpenOffice.org Internet documentation or to the   <br>  "  -
    "locally installed OpenOffice.org developer kit.                                 "  -
    "                                                                                "  -
    "<p>The user can determine:                                                      "  -
    "   <ul>                                                                         "  -
    "   <li> the fully qualified name of the UNO IDL definition to analyze,          "  -
    "   <li> which font should be used for rendering,                                "  -
    "   <li> which numbering types should be used for the chapter levels,            "  -
    "   <li> whether the rendered result should only be viewed or whether it should <br>  "  -
    "        be saved, printed and/or rendered to PDF,                               "  -
    "   <li> whether the members of the analyzed UNO IDL definition should be       <br>  "  -
    "        analyzed themselves (&quot;2 show first and second layer&quot;)          "  -
    "   <li> where the generated hyperlinks point to (Internet or local machine).    "  -
    "   </ul>                                                                        "  -
    "<p>                                                                             "  -
    "                                                                                "  -
      "Just experiment a little bit in order to find the settings of your liking.    "  -
    "                                                                                "  -
    "<p>Nicole Scholz, Vienna, Austria 2009                                          "
```

Code 19: Information window created with `HTML`

The following figure shows the information window which is accessed through the information menu entry.

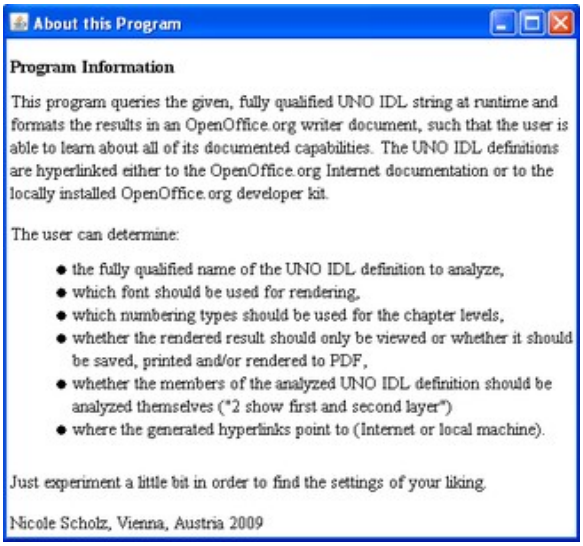


Figure 52: `frontend.rex`: information window

To make it possible to add the `HTML` text to the content pane, the Java class `javax.swing.JeditorPane` is needed. This class allows to access different text editors. Only the used text editor has to be passed to the `JEditorPane` and the formatted text [Java03e]. If the text should not be allowed to be changed the `setEditable` `method` can be set false [Java03f]. After creating the editor, it can be added to the content pane.

```
  jEdit=.bsf~new("javax.swing.JEditorPane", "text/html", txt)
  jEdit~setEditable(.false)
  frame1~getContentPane~add(jEdit)
```

Code 20: Add window to frame

The layout of the information window is set with the Java class `java.awt.GridLayout`. In a grid layout columns and rows can be defined. In the example below only one column and row are displayed [Java03g]. This window also gets a window close `method`. Here the `closeInfoWindow` `method` which is defined below is used. With the `.frame1~~pack~~show~~toFront` command the information window is packed and shown in the front of the other program window. This ensures that the new opened window is always shown in front of the other opened programs [Java03h] [Java03i] [Java03j].

```
  .frame1~getContentPane~setLayout(.bsf~new("java.awt.GridLayout", "1", "1"))
  .frame1~bsf.addEventListener('window', 'windowClosing', 'Call closeInfoWindow')
  .frame1~~pack~~show~~toFront
  return
```

Code 21: Set layout and add event listener for closing

The following code lines show the `closeInfoWindow` routine which closes the information window with the command `.frame1~~show~~dispose` [Java03i]. The `method` dispose destroys the window components and returns the used memory back to the operating system [Java03k].

```
  -- method for closing one frame
::routine closeInfoWindow
  .frame1~~show~~dispose
  return
```

Code 22: Close information window

After programming the menu, the other elements of `frontend.rex` are created. To achieve this, the content pane of the frame has to be accessed because the labels and the buttons are added to it. The content pane can be accessed with the command `frame~getContentPane` [Java03l]. For the `frontend.rex` program a Grid layout is chosen because the amount of columns and rows can be easily defined by the user. In this

program code two columns and ten rows are defined [Java03g]. The layout has to be set to the content pane of the frame with the `method setLayout` [Java03a].

```
frameContentPane=frame~getContentPane

/*layout is set to Gridlayout*/
frameContentPane~setLayout(.bsf~new("java.awt.GridLayout", "10", "2"))
```

Code 23: Set window layout

The following code displays the difference between accessing a Java class on the usual way and after importing the Java class. Up to the code pane below Java classes are created with `.bsf~new(java class, definition)`. If you import a Java class, an instance is created in another way. At the begin of the program code the `JLabel` has been imported with `call bsf.import "javax.swing.JLabel", "swingLabel"`. After this import the class can be accessed with `.swingLabel~new()`. Between the brackets the label text can be defined. To get an empty label only two apostrophes are added between the brackets [Java03m].

```
/*create textfields and buttons*/
labelBegriff=.swingLabel~new("Enter a fully qualified UNO IDL name:")
labelLeer   =.swingLabel~new("")
frameContentPane~~add("1", labelBegriff)      -
                  ~add("2", labelLeer)
```

Code 24: Create text field and button

The labels are added to the content pane with the command `add()`. As the labels are added to a grid layout it has to be specified on which place the labels should appear. Within the brackets, the field number has to be stated first and second the label which should be added. The following figure shows which number is assigned to which field in the grid layout. The numbering starts on the left top side. The second field is the field on the right top side and so on. The first label with the text “Enter a fully qualified UNO IDL name:” is attached at the first place on the left top of the content pane.

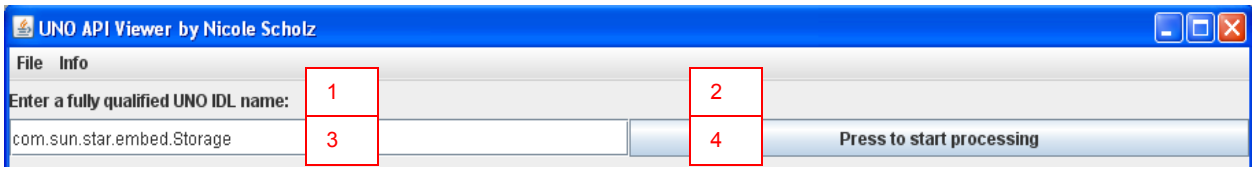


Figure 53: `frontend.rex`: grid layout numbering

Next a text field is added to the content pane. Hence an instance of `javax.swing.JTextField` has to be created. With the `method` `setText` a text can be added to the text field. Here the UNO IDL name "`com.sun.star.embed.Storage`" is added to the text field as an example how the IDL can be entered [Java03n]. Afterwards the text field is attached to the content pane.

```
tf_UNO_IDL_Name=.bsf~new("javax.swing.JTextField", "", 30)
tf_UNO_IDL_Name~setText("com.sun.star.embed.Storage")
frameContentPane~add("3", tf_UNO_IDL_Name)
```

Code 25: Create text field

Furthermore four choice boxes are created. This gives the user the possibility to choose some different ways of the document creation and design. So everyone can configure the result as required. First the user can determine which font type is to be used in the writer document. Therefore a combo box was created with the `javax.swing.JComboBox` class. Then the different font types have to be added to the box with the command `~addItem`. With the `method` `setSelectedItem` a preselected item can be specified. In the code below the font `Verdana` is predefined [Java03o].

```
/*font choice */
cffont = .bsf~new("javax.swing.JComboBox")
cffont~~addItem("Arial")                              -
       ~~addItem("Bitstream Vera Sans Mono")          -
       ~~addItem("Courier New")                       -
       ~~addItem("Cordia New")                        -
       ~~addItem("DejaVueSans Condensed")             -
       ~~addItem("DejaVueSans Mono")                  -
       ~~addItem("Franklin Gothic Book")              -
       ~~addItem("Gill Sans MT")                      -
       ~~addItem("Lucida Sans")                       -
       ~~addItem("Old English Text MT")               -
       ~~addItem("Times new Roman")                   -
       ~~addItem("Verdana")
cffont~setSelectedItem("Verdana")   -- select "Verdana"
label2=.swingLabel~new("Choose the font to use for rendering:")
frameContentPane~~add("9", .swingLabel~new)~~add("10", .swingLabel~new)
```

Code 26: Create choice box

The following figure shows the open font choice box.

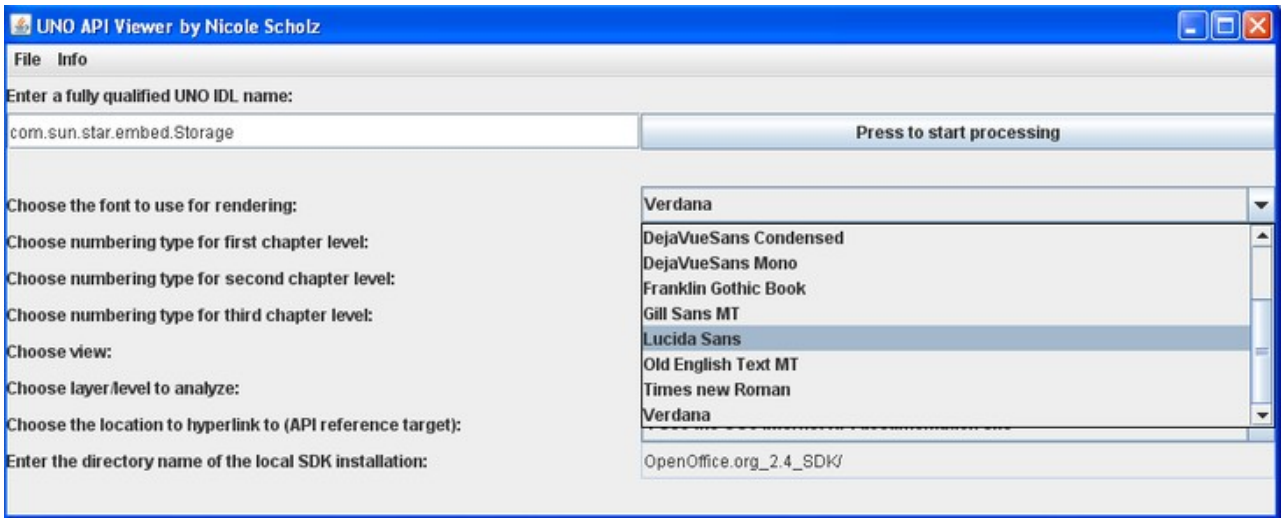


Figure 54: `frontend.rex`: font choice box

Next the user can choose which numbering type he would like to use for the different chapter levels. Therefore three combo boxes are available for each chapter level one. The numbering type names are not always easy to understand, so a tool tip text gives a short explanation how the numbering will look like. Each time another numbering type has been chosen, the tool tip text changes. The tool tip text can be set to an element with the command `setToolTipText`. The code below shows how it can be set on the combo box `cffirstlevel` [Java03q] [Java03o].

```
cffirstlevel~setToolTipText("Numbering is in Roman numbers" )
```

Code 27: Tool tip text

The following figure shows the three combo boxes, one for each chapter numbering level. For the first combo box the tool tip text is shown.

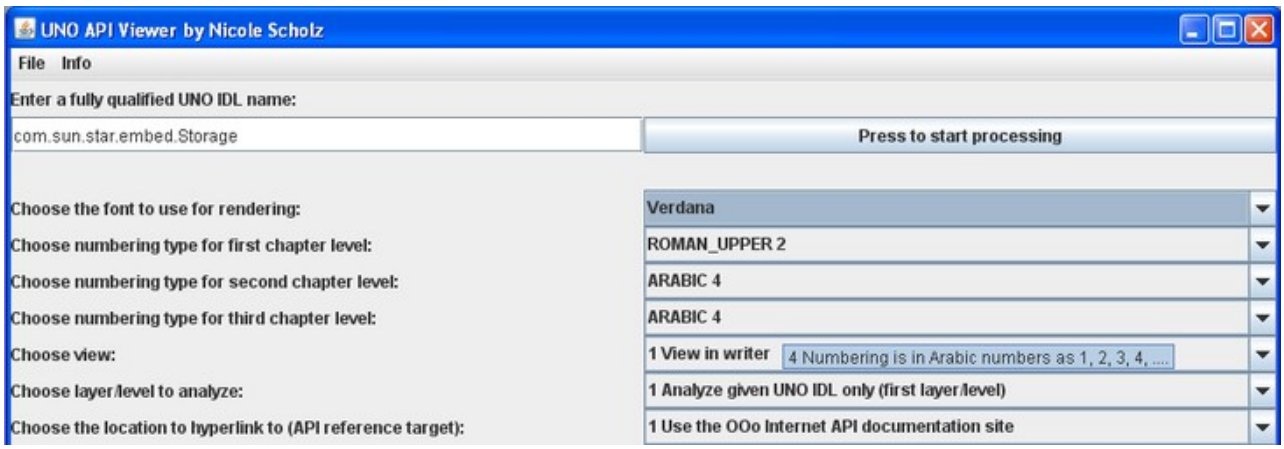


Figure 55: Tool tip text for the chapter numbering types

The next figure shows the opened combo box for the first chapter level numbering type. The combo box includes all current numbering types provided by Open Office. The numbering types are ordered alphabetically and after the numbering type the value of them is shown.

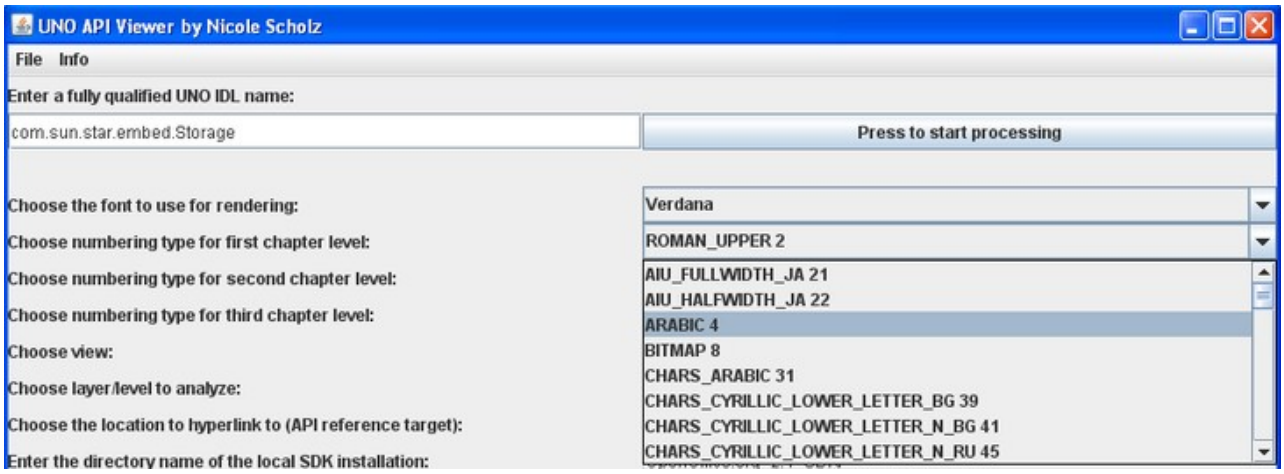


Figure 56: Chapter numbering types combo box

The next possibility for the user is to choose how the document should be displayed. Therefore again a combo box is used and different elements are added to it. First there is the possibility to show only the created writer document. The second alternative is to show the writer file and to store it automatically. With the third option the writer document is saved and/or printed and the last option is to save the writer document and to create a `pdf` file [Java03o]. The figure below shows the box with the different possibilities.

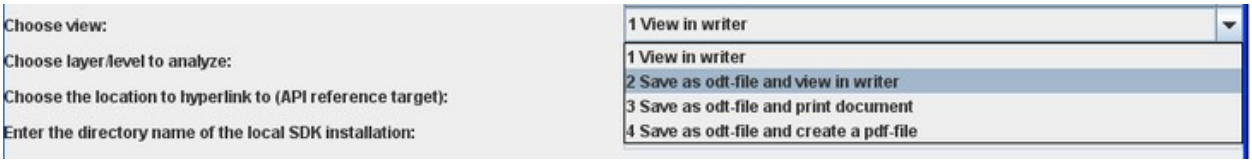


Figure 57: View choice

Another important feature of the API-Viewer is that the user can choose, if only the requested UNO IDL should be shown, or the members of this IDL too. Here again a combo box is offered where the user can choose the requested item [Java03o]. The next figure shows the layer choice box.

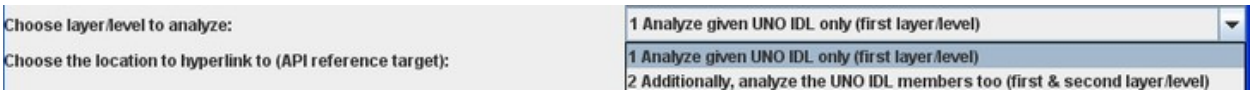


Figure 58: Layer choice

The API-Viewer offers the opportunity to look up the UNO IDL's in the Open Office API. Therefore hyperlinks are added to each UNO IDL. These hyperlinks are either a connection to the internet or, if there is no internet available, to a local installed API of Open Office. This API is included in the Open Office Software Development Kit (SDK) and can be downloaded from the Open Office homepage. After installing the Open Office SDK the local API can be used [OOOorgc]. The API-Viewer gives the user the possibility to choose which API to use. For the local API there are two different ways of displaying the hyperlink. If the operation system is windows, the hyperlinks are set without jump marks, otherwise they are currently not working, whereas in Unix the jump marks can be set. To provide these options a combo box is created and the three choice elements are added to it as the following code shows [Java03o]. The following figure shows the box with the different choices.

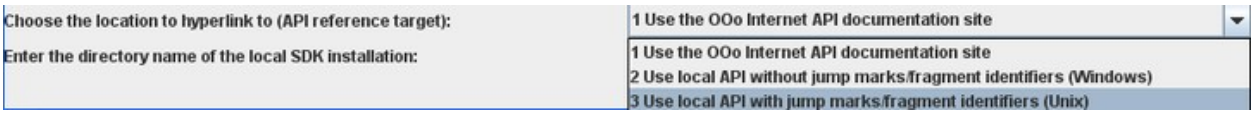


Figure 59: Open Office API location

If the local API of the Open Office SDK is used, the location where it has been installed to has to be specified. To denounce where the Open Office SDK is located, a text field is provided. This text field can only be edited when the local API is chosen. Therefore an event listener is added to the URL combo box which is shown in the following code. If the selected item of the box is not the first one, the text field is set editable.

```
cfURL~bsf.addEventListener('action', '',
'tf_local_URL~setEditable(cfURL~getSelectedItem~left(1)<>1)')
```

Code 28: Add event listener

The text field is created with the `javax.swing.JTextField` class and it is set to "not editable" until the local API is chosen [Java03n].

```
label5=.swingLabel~new("Enter the directory name of the local SDK installation: ")
tf_local_URL=.bsf~new("javax.swing.JTextField", "", 35)
```

```
tf_local_URL~setText("OpenOffice.org_2.4_SDK/")
tf_local_URL~setEditable(.false)     -- do not allow to edit, unless "cfURL<>1"
frameContentPane~~add("19", label5)~add("20", tf_local_URL)
```

Code 29: Create text field

The following figure shows a cut of the `frontend.rex` display. If the Open Office internet API is used, the text field below can not be edited. To mark that the text field can not be changed, it is colored in grey.

Figure 60: `frontend.rex`: local API location not editable

The next figure shows that after choosing the local API the text field with the Open Office SDK location can be edited.

Figure 61: `frontend.rex`: local API location editable

Now the possibilities of custom designing the output of the API-Viewer are explained. To start the query about the UNO IDL's, a button is needed. This button is created with the `javax.swing.JButton` class [Java03p]. To make sure the program can be started, an event listener has to be attached to the button. This event listener calls the `processUNO_IDL` routine from the `createApiInfo.rex` program. To this routine the selected elements are passed.

```
buttonStartProcessing= .bsf~new("javax.swing.JButton", "Press to start processing")
buttonStartProcessing~bsf.addEventListener("action","",                    -
                      'call processUNO_IDL '                               -
                      'frame,'                                             -
                      'tf_UNO_IDL_Name,'                                   -
                      'cfLayer,'                                           -
                      'cfView,'                                            -
                      'cfURL,'                                             -
                      'tf_local_URL,'                                      -
                      'cffont')
frameContentPane~add("4", buttonStartProcessing)
```

Code 30: Add button

After creating and adding all elements of the content pane the program frame can be packed and shown. The frame is shown in the Front this means it is visible after starting the program [Java03h] [Java03i] [Java03j].

```
/* paint frame in front and set size of the frame*/
frame~~pack~~show~toFront
```

Code 31: Show frame

In order to achieve that the program window is displayed until the user closes it, the following program code is needed. The program is shown until the result is “SHUTDOWN REXX!”.

```
do forever
        INTERPRET .bsf~bsf.pollEventText
        if result="SHUTDOWN REXX!" then leave
end
```

Code 32: Close `frontend.rex`

3.3 Starting API-Viewer within another program

It is possible to start the API-Viewer from another program. In the following chapter two alternatives are described to call the API Viewer. It can be called directly from a program with the needed attributes or an Open Office `service` can be passed to the `createApiInfo.rex` program.

The following code shows a Rexx program which calls the `createApiInfo.rex` directly. The first instruction calls the API-Viewer with the requested Open Office query but no other attributes specified. Here the default arguments are used which are defined in the `createApiInfo.rex` program. The second call specifies that the second layer should be shown too, so all member definitions of “`com.sun.star.embed.Storage`” are displayed. The other arguments also get the default value set in the `createApiInfo.rex` program. The default values are always used if one attribute is not defined in the method call.

```
/* little Rexx program that calls createApiInfo.rex directly*/
call createApiInfo 'com.sun.star.embed.Storage'
call createApiInfo 'com.sun.star.embed.Storage', 2  -- show member definitions as well
```

Code 33: Call `CreateApiInfo`

The second possibility to call a `service object` of Open Office, is displayed by the following code. First the `createApiInfo.rex` with no query is called. Then a connection to the Open Office Writer is set with the command `uno.createDesktop` [RGF08d]. This command returns the Open Office desktop object. Furthermore a Writer file is created. The last two instructions call Open Office `service objects`. Therefore the `analyzeAndCreateReference` routine of the `createApiInfo.rex` program is called and the `service objects`, which were created before, are passed to it. This will start the whole query procedure.

```
call createApiInfo.rex

desktop           = UNO.createDesktop()
xDesktop          = desktop~XDesktop
xComponentLoader  = xDesktop~XComponentLoader
writerComponent   = xcomponentLoader                                          -
                    ~loadComponentFromURL("private:factory/swriter", "_blank", 0, .UNO~noProps)
xTextDocument     = writerComponent~xTextDocument
xText             = xTextDocument~getText
xTextCursor       = xText~createTextCursor
xTextRange        = xText~getEnd
xTextRange~setString("Hello, this is ooRexx speaking on:" date("S") time())

call analyzeAndCreateReference writerComponent
call analyzeAndCreateReference xTextCursor, 2  -- show definition of members as well

::requires uno.cls
```

Code 34: Calling a `service object` of Open Office

4 Code Snippets

In the next chapters some helpful code snippets are described. A code snippet is a short program code example which illustrates how a function can be implemented. These code snippets explain important functions of Open Office automation which were needed for the creation of the API Viewer program. Most of these snippets are also available on the Open Office code snippet homepage, so that everyone can use them.

4.1 Merge Table Columns

This code snippet shows how table columns can be merged. The merged table columns can be in the middle of the table without any affect on the other cells. Only the selected table cells are merged.

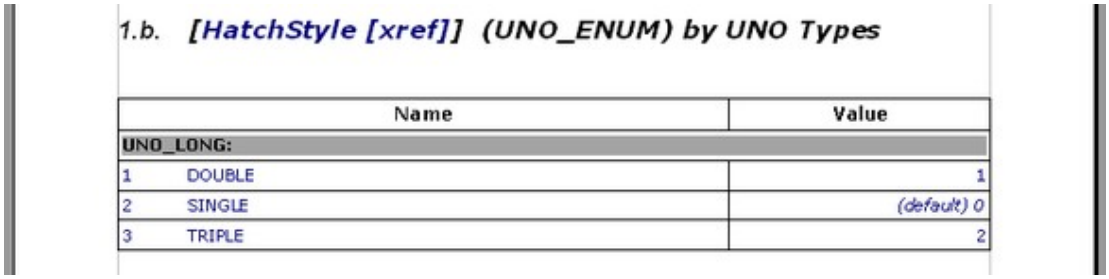
First a table in sWriter has to be created. Then a `TextCursor` for the targeted table row has to be created with the command `createCursorByCellName` of the `interface` `XTextTable` [OOOAPIag]. In the next step the cursor is moved to the second cell with the `method` `goToCellByName` of the `interface` `XTextTableCursor` in the same row and then the command `mergeRange` is send to the text cursor. The `method` `mergeRange` of the `interface` `XTextTableCursor` merges a selected range of cells [OOOAPIai] [OOOCSBg].

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()    -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                              -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet
xDMsf = xTextDocument~XMultiServiceFactory

/* create the table with 3 columns*/
xTextTable = xDMsf~createInstance("com.sun.star.text.TextTable")~XTextTable
xTextTable~initialize(1, 3) -- initialize the table

/* insert TextTable in the Text */
xText~insertTextContent(xTextCursor, xTextTable, .false)

/* insert text into the table */
call setCellText "A1", "merge column A1 and B1", xTextTable

  -- merge column A1 and B1
```

```
xTextTableCursor=xTextTable~createCursorByCellName("A1") -- create cursor
  -- select area up to and including this cell
xTextTableCursor~gotoCellByName("B1", .true)
xTextTableCursor~mergeRange                              -- merge selected cells
xTextTable~createCursorByCellName("A1") ~~gotoCellByName("B1", .true)

::requires UNO.cls

  -- routine to set the cell text
::routine setCellText
  use arg cell, text, xTextTable
  xCellText = xTextTable~getCellByName(cell)~XText
  xCellCursor = xCellText~createTextCursor()
  cursorProps = xCellCursor~XPropertySet
  xCellText~setString(text)
  return
```

Code 35: Merge table columns

The following figure shows a table where the columns of the second table row are merged. The other rows have two columns and are not affected by the merge of the columns in the second row.



1.b. [HatchStyle [xref]] (UNO_ENUM) by UNO Types

	Name	Value
UNO_LONG:		
1	DOUBLE	1
2	SINGLE	(default) 0
3	TRIPLE	2

Figure 62: `CreateTableByTypes` table cell merge

4.2 Adjust Table Cell Text to the Right

The following code snippet displays how the text can be adjusted to the right side of the column.

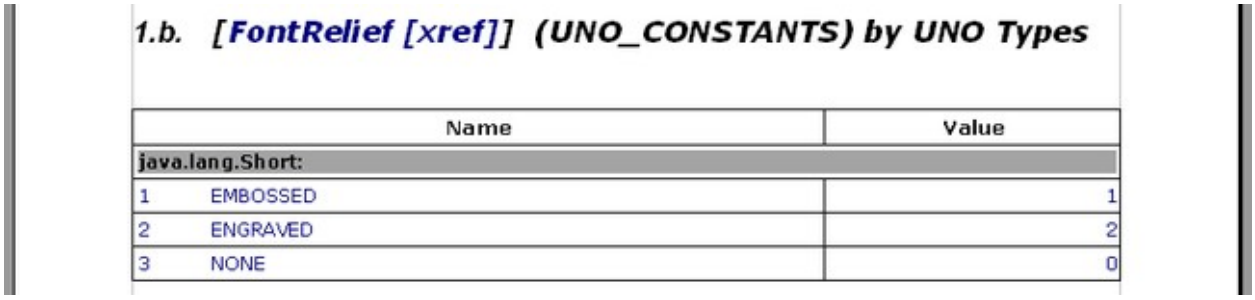
After creating a table in sWriter, the text can be inserted into the table. To adjust the before added text to the right side of the table column, the `property` `ParaAdjust` from the `service` `ParagraphProperties` has to be set to `right`. To be able to set the properties, a text cursor has to be created first and then the `XPropertySet` `interface` has to be accessed. Finally the paragraph `property` can be changed to right with the `method` `setPropertyValue` [OOOAPIz] [OOOCSBh].

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
```

```
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

-- import the enum
call bsf.import "com.sun.star.style.ParagraphAdjust", "paragraphAdjust"

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet
xDMsf = xTextDocument~XMultiServiceFactory

/* create the table */
xTextTable = xDMsf~createInstance("com.sun.star.text.TextTable")~XTextTable
xTextTable~initialize(1, 2) -- initialize the table

/* insert TextTable in the Text */
xText~insertTextContent(xTextCursor, xTextTable, .false)

/* insert text into the table */
call setCellText "A1", "text", xTextTable

::requires UNO.cls

  -- routine to set the cell text
::routine setCellText
  use arg cell, text, xTextTable
  xCellText = xTextTable~getCellByName(cell)~XText
  xCellCursor = xCellText~createTextCursor()
  cursorProps = xCellCursor~XPropertySet
  -- set the property of the cell text
  cursorProps~setPropertyValue("ParaAdjust", .paragraphAdjust~"RIGHT")
  xCellText~setString(text)
  return
```

Code 36: Adjust table cell text to the right

The following figure shows a table with two columns. The text in the second column is adjusted to the right.



1.b. [FontRelief [xref]] (UNO_CONSTANTS) by UNO Types

	Name	Value
java.lang.Short:		
1	EMBOSSED	1
2	ENGRAVED	2
3	NONE	0

Figure 63: Adjust table text to the right

4.3 Add Date to Footer

To create a footer an instance of the "com.sun.star.style.PageStyle" service has to be created. This service allows accessing the common properties of the page styles [OOOAPIk]. Here also the XStyleFamiliesSupplier has to be retrieved, so that the styles of the document can be accessed with the command getStyleFamilies [OOOAPIg]. Then the "Standard" style of the document and its properties have to be collected so that the footer can be activated. FooterIsOn property has to be set true [OOOAPIk].

After activating the footer, we want the date and the time of the document creation to be added to the footer. To achieve this, the service "com.sun.star.text.TextField.DateTime" is used. This service has the properties IsDate and IsFixed. To display the date the property IsDate has to be set true, and to show the time the same property has to be set false. It is important to set the property IsFixed true so that the date and the time are constant. Then it is possible to see when the document is created [OOOAPIo] [OOOCSBc].

Next the text is inserted to the footer. So the FooterText has to be accessed. Then a line is created and added to it. For the line creation an instance of the service "com.sun.star.drawing.LineShape" is needed [OOOAPIp]. Afterwards the position and the size of the line are defined. The position is set with the struct "com.sun.star.awt.Point" which has two elements: The first one is the X- and the second one the Y-axis [OOOAPIq]. For the size the struct "com.sun.star.awt.Size" is needed. This struct has also two elements: The width and the height. Width and height have both the type long [OOOAPIr]. Afterwards the line is inserted into the footer with the method insertTextContent of the interface XText. The method insertTextContent needs three arguments, the range where it should be inserted, what should be inserted and if the range should be replaced. Here the range is the footer text end, the element which should be added the line and the last argument is set false with the .false object. This object is the constant "0" representing a false result for logical operations [AFHMMPe] [OOOAPIj]. If only text should be inserted, the method setString can be used. This method only gets text which should be added to the document as argument [OOOCSBm].

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
```

```
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                                -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xWriterDocument = xWriterComponent~XTextDocument
xText = xWriterDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet
xDMsf = xWriterDocument~XMultiServiceFactory

  -- create the footer
xPageStyle = xDMsf~createInstance("com.sun.star.style.PageStyle")
xFamiliesSupplier = xWriterDocument~XStyleFamiliesSupplier
xStyle = xFamiliesSupplier~getStyleFamilies~getByName("PageStyles")~XNameContainer
xFooter = xStyle~getByName("Standard")
FooterProperty = xFooter~xPropertySet
FooterProperty~setPropertyValue("FooterIsOn", box("boolean", .true))
footerText = FooterProperty~getPropertyValue("FooterText")~xText

  -- Create the current date
datetime = xDMsf~createInstance("com.sun.star.text.TextField.DateTime")
datetimeProps = datetime ~XPropertySet()
datetimeProps~setPropertyValue("IsDate", box("boolean",.true))
datetimeProps~setPropertyValue("IsFixed", box("boolean",.true))
datetimeTC = datetime~XTextContent()
  -- Create the current time
datetime1 = xDMsf~createInstance("com.sun.star.text.TextField.DateTime")
datetimeProps1 = datetime1 ~XPropertySet()
datetimeProps1~setPropertyValue("IsDate", box("boolean",.false))
  -- set the date constant so it does not change when the document is opened
datetimeProps1~setPropertyValue("IsFixed", box("boolean",.true))
datetimeTC1 = datetime1~XTextContent()

  -- create a line
xTextCursorFooter = footerText~createTextCursor
Line = xDMsf~createInstance("com.sun.star.drawing.LineShape")~xShape
Line~setPosition(.bsf~new("com.sun.star.awt.Point", 1, 1))
Line~setSize(.bsf~new("com.sun.star.awt.Size", 17000, 0))
xTextContentShape = Line~xTextContent
  -- insert the line at the top of the footer
footerText~insertTextContent(footerText, xTextContentShape, .false)

  -- insert date and time to the footer
footerText~getEnd~setString(" date: ")
footerText~insertTextContent(footerText~getEnd, datetimeTC, .false)
footerText~getEnd~setString(" time: ")
footerText~insertTextContent(footerText~getEnd, datetimeTC1, .false)

::requires UNO.cls
```

Code 37: Add date to footer

The following Figure shows the footer of a created document with the added date and time.

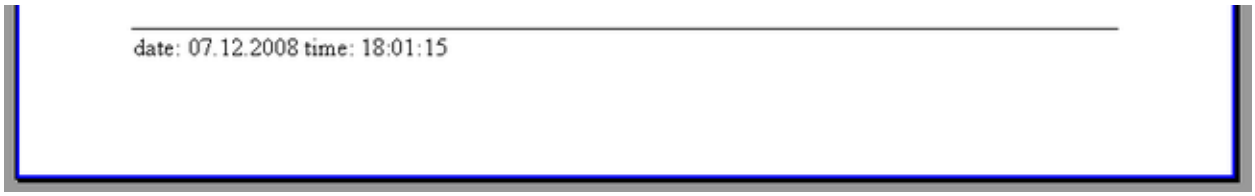


Figure 64: Footer

4.4 Change Table Column Width

In this chapter the way of how to change the column width in sWriter is described. This is especially helpful, when a table has more columns and the text in the columns has different length. So the width of the columns can be individually changed to fit perfectly to the entered text.

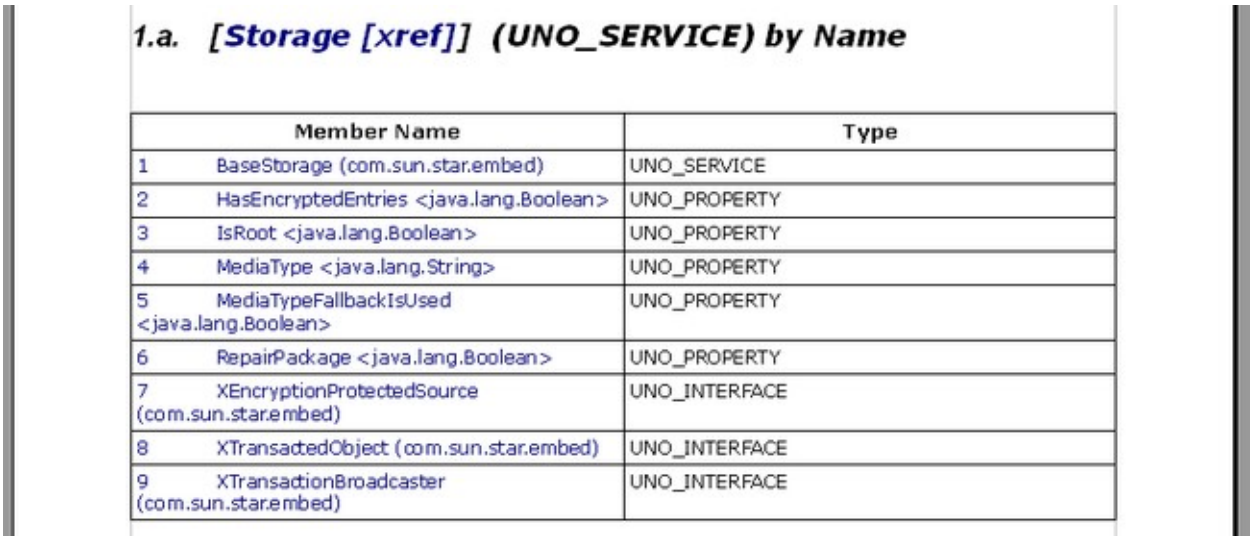
To change the column width the routine `setColumnWidth` is called. It needs the cell and the table as arguments. First the text cursor of the cell in which the text should be inserted and the table properties are accessed. To change the column width the `TableColumnSeperators` and the `TableColumnRelativeSum` of the `service` `TextTable` are needed. The `TableColumnSeperators` contains the table border description. The `TableColumnRelativeSum` includes the sum of the column width values used in the `TableColumnSeperators`. To change the column width the `TableColumnSeperators` have to be changed to a new relative sum. In this case the sum is changed to seventy per cent for the column separator. Then the new separator is added to the `property` `TableColumnSeperators` [OOOAPIae] [OOOCSBf].

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                              -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet
xDMsf = xTextDocument~XMultiServiceFactory

/* create the table */
xTextTable = xDMsf~createInstance("com.sun.star.text.TextTable")~XTextTable
xTextTable~initialize(1, 2) -- initialize the table
```

```
/* insert TextTable in the Text */
xText~insertTextContent(xTextCursor, xTextTable, .false)

/* insert text into the table */
call setCellText "A1", "table width", xTextTable
call setColumnWidth "A1", xTextTable

::requires UNO.cls

  -- routine to set the cell text
::routine setCellText
        use arg cell, text, xTextTable
        xCellText = xTextTable~getCellByName(cell)~XText
        xCellCursor = xCellText~createTextCursor()
        cursorProps = xCellCursor~XPropertySet
        xCellText~setString(text)
        return

  -- routine to change the column width of a table
::routine setColumnWidth
    use arg cell, xTextTable
  xTextTableCursor = xTextTable~createCursorByCellName(cell)
  xTableProps=xTextTable~xPropertySet
  colRelSum=xTableProps~getPropertyValue("TableColumnRelativeSum") -- usually: 10000
  colSeparators=xTableProps~getPropertyValue("TableColumnSeparators") -- total sum
  -- new position of the column separator
  colSeparators[1]~position=trunc(colRelSum*0.70)
  -- set the property with the calculated position of the separator
  xTableProps~setPropertyValue("TableColumnSeparators", colSeparators)
  return
```

Code 38: Change table column width

The next picture shows a table in Writer where the column width is set automatically. The two columns have both the same size regardless how long the inserted text is.

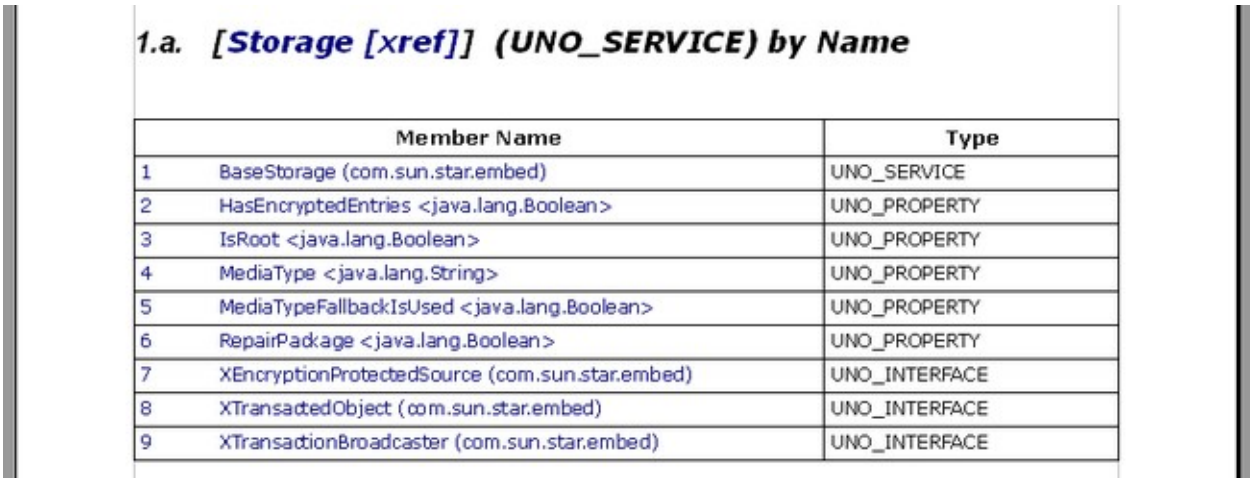


1.a. [Storage [xref]] (UNO_SERVICE) by Name

	Member Name	Type
1	BaseStorage (com.sun.star.embed)	UNO_SERVICE
2	HasEncryptedEntries <java.lang.Boolean>	UNO_PROPERTY
3	IsRoot <java.lang.Boolean>	UNO_PROPERTY
4	MediaType <java.lang.String>	UNO_PROPERTY
5	MediaTypeFallbackIsUsed <java.lang.Boolean>	UNO_PROPERTY
6	RepairPackage <java.lang.Boolean>	UNO_PROPERTY
7	XEncryptionProtectedSource (com.sun.star.embed)	UNO_INTERFACE
8	XTransactedObject (com.sun.star.embed)	UNO_INTERFACE
9	XTransactionBroadcaster (com.sun.star.embed)	UNO_INTERFACE

Figure 65: Table with normal column width

The next figure displays a table with changed column width so that the second column is smaller than the first one because the text in the first one is longer than in the second one.



1.a. [Storage [xref]] (UNO_SERVICE) by Name

	Member Name	Type
1	BaseStorage (com.sun.star.embed)	UNO_SERVICE
2	HasEncryptedEntries <java.lang.Boolean>	UNO_PROPERTY
3	IsRoot <java.lang.Boolean>	UNO_PROPERTY
4	MediaType <java.lang.String>	UNO_PROPERTY
5	MediaTypeFallbackIsUsed <java.lang.Boolean>	UNO_PROPERTY
6	RepairPackage <java.lang.Boolean>	UNO_PROPERTY
7	XEncryptionProtectedSource (com.sun.star.embed)	UNO_INTERFACE
8	XTransactedObject (com.sun.star.embed)	UNO_INTERFACE
9	XTransactionBroadcaster (com.sun.star.embed)	UNO_INTERFACE

Figure 66: Table with changed column width

4.5 Set Hyperlink

The following code snippet shows how a hyperlink can be set in sWriter. To add a hyperlink to a text, the `property` of the text has to be set with the `property` `HyperlinkUrl`. To this URL the hyperlink can be added. Then the text which should contain the hyperlink is added to the text document. This text is colored blue and underlined so it can be immediately identified as a hyperlinked text [OOOCSBi].

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()    -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                              -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
Textprops = xTextCursor~xPropertySet

  -- set a hyperlink
Textprops~setPropertyValue("HyperLinkURL", "http://api.openoffice.org/")
xText~insertString(xTextCursor, "module star", .false)

::requires UNO.cls
```

Code 39: Set hyperlink

The next figure displays text with a hyperlink. The hyperlink refers to the Open Office API star `module`.

Figure 67: Text with hyperlink

4.6 Set Hyperlink within a Table

Setting a hyperlink within a table looks similar to setting a hyperlink in a normal text like in chapter 4.5. In this example the `property` of the table cell text has to be set to the needed URL. After setting the hyperlink the text can be added to the table cell. This text is colored blue and underlined so it can be immediately identified as a hyperlinked text.

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                              -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet
xDMsf = xTextDocument~XMultiServiceFactory

/* create the table */
xTextTable = xDMsf~createInstance("com.sun.star.text.TextTable")~XTextTable
xTextTable~initialize(1, 2) -- initialize the table

/* insert TextTable in the Text */
xText~insertTextContent(xTextCursor, xTextTable, .false)

/* insert text into the table */
call setCellText "A1", "hyperlink", xTextTable

::requires UNO.cls

  -- routine to set the cell text
::routine setCellText
  use arg cell, text, xTextTable
  xCellText = xTextTable~getCellByName(cell)~XText
  xCellCursor = xCellText~createTextCursor()
  cursorProps = xCellCursor~XPropertySet
  cursorProps~setPropertyValue("HyperLinkURL", "http://api.openoffice.org/")
  xCellText~setString(text)
  return
```

Code 40: Hyperlink within a table

The following figure shows a table with a text added to the first cell. This text is hyperlinked to the Open Office API.

Figure 68: Hyperlink within a table

4.7 Shape with URL Text Fields

In this chapter it is described how to add an URL text field to a shape. This is a possibility to add a hyperlinked text to a shape.

First a rectangle has to be created using the `service` "`com.sun.star.drawing.RectangleShape`". After adding the rectangle to the sWriter document, a text field has to be created with the `service` "`com.sun.star.text.TextField.URL`". Then the properties of the text field have to be set. The `Representation property` includes the text which should be displayed. The `TargetFrame` specifies the name of the target frame in which the URL will be opened [OOOAPIap].

After creating the text field, it is attached to the shape with the command `insertTextContent`. After each text field a paragraph break is set so that the next text field is added below the other text fields [OOOAPIj] [OOOAPIaj].

```
call bsf.import "com.sun.star.text.ControlCharacter", "ctlChar"

oDesktop         = UNO.createDesktop()
xComponentLoader = oDesktop~XDesktop~XComponentLoader

url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                          -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument    = xWriterComponent~XTextDocument
xDocText         = xTextDocument~getText
xDocTextCursor   = xDocText~createTextCursorByRange(xDocText~getStart)
xDocFactory      = xTextDocument~XMultiServiceFactory

/* create a Rectangle Shape */
Shape = xDocFactory~createInstance("com.sun.star.drawing.RectangleShape")
```

```
xShape = Shape~xShape

size = .bsf~new("com.sun.star.awt.Size", 8000 /*width*/, 8000 /*height*/)
xShape~setSize(size)

xShapeProps=xShape~xPropertySet
at_para=bsf.getConstant("com.sun.star.text.TextContentAnchorType","AT_PARAGRAPH")
xShapeProps~setPropertyValue("AnchorType", at_para)
  --set the fill color of the shape
xShapeProps~setPropertyValue("FillColor", box("int", "C0 C0 C0"x ~c2d))

-- create a text field at the document factory
xTextFieldProps=xDocFactory~createInstance("com.sun.star.text.TextField.URL")           -
                ~XPropertySet

xTextFieldProps~setPropertyValue("Representation","OpenOffice.org API Project")
xTextFieldProps~setPropertyValue("TargetFrame", "_blank")
xTextFieldProps~setPropertyValue("URL", "http://api.openoffice.org")

-- get the XTextContent of the shape and the field
xShapeTextContent=xShape~XTextContent
xFieldTextContent=xTextFieldProps~XTextContent

-- the shape is inserted at the DOCUMENT text
xDocText~insertTextContent(xDocTextCursor, xShapeTextContent, .false)

-- access the text inside the shape,
-- and create a text cursor
xShapeText       = xShape~XText
xShapeTextCursor = xShapeText~createTextCursor

-- insert the field at the SHAPE text
paraBreak=.CtlChar~paragraph_break     -- gets used multiple times in this routine
xShapeText~insertTextContent(xShapeTextCursor, xFieldTextContent, .false)
xShapeText~insertControlCharacter(xShapeTextCursor, .CtlChar~paragraph_break, .false)

-- create another text field
xTextFieldProps=xDocFactory~createInstance("com.sun.star.text.TextField.URL")~XPropertySet
xTextFieldProps~setPropertyValue("Representation", "Open Office homepage")
xTextFieldProps~setPropertyValue("TargetFrame", "_blank")
xTextFieldProps~setPropertyValue("URL", "http://api.openoffice.org/")

-- add it to the same shape below the first one
xFieldTextContent=xTextFieldProps~XTextContent
xShapeText~insertTextContent(xShapeTextCursor, xFieldTextContent, .false)

::requires UNO.cls
```

Code 41: Shape with two URL text fields

The following figure shows a rectangle with two URL text fields added to it. The text within the URL textfields is colored blue but not underlined. There is no hyperlink information shown when the mouse cursor is pointing on the URL. To open the hyperlink target the control key has to be pushed and the hyperlinked text clicked.

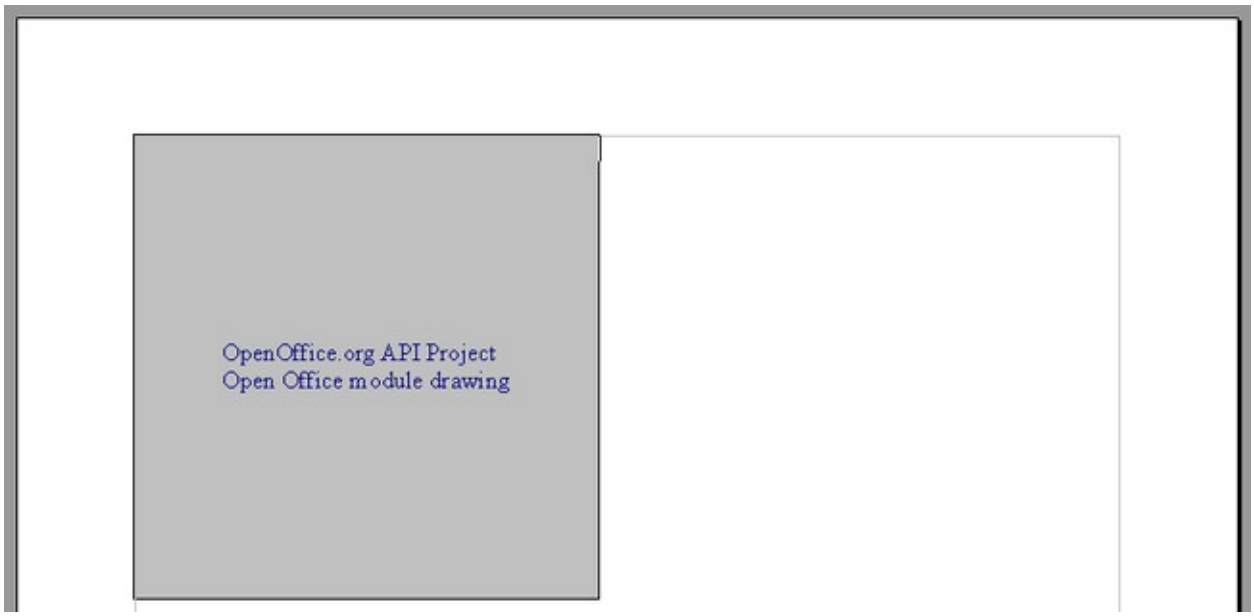


Figure 69: Hyperlinked text field within a rectangle

The next two figures illustrate a rectangle with text field. In the first picture the text fields are not separated with a paragraph break in the second they are.

Figure 70: Shape with URL text field without paragraph breaks between the text fields

Figure 71: Shape with URL text field with a paragraph break between the text fields

4.8 Deactivate Hyperlink Underlining

In the following code snippet the underlining of hyperlinks within the whole document is deactivated to make the layout of the document clear and better to read. To achieve this

the character styles of the document have to be accessed. Therefore the `XStyleFamiliesSupplier` with its `method` `getStyleFamilies` is needed. After getting the character styles, the properties of the internet links and the visited internet links are accessed with the style families and the character underlining is set to `NONE`. Therefore the `constant` `NONE` has to be accessed through the `"com.sun.star.awt.FontUnderline"` and added to the `property` `CharUnderline` of the `service` `"com.sun.star.style.CharacterProperties"` [OOOAPIaa] [OOOAPIah]. So the links before and after visiting are not underlined in the whole Writer document but they are still colored blue so that they are recognized as hyperlinks [OOOAPIg].

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()    -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                                    -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet

xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
Textprops = xTextCursor~xPropertySet

/* disable underlining of hyperlinks in the whole document */
xFamiliesSupplier = xTextDocument~XStyleFamiliesSupplier
xCharStyleFamily = xFamiliesSupplier~getStyleFamilies~getByName("CharacterStyles")~XNameContainer
xInternetLinkStyleProps= xCharStyleFamily~getByName("Internet link")
xVisitedInternetLinkStyleProps=xCharStyleFamily~getByName("Visited Internet Link")
xInternetLinkStyleProps~xPropertySet~setPropertyValue("CharUnderline",                 -
            box("short", bsf.getConstant("com.sun.star.awt.FontUnderline", "NONE")))
xVisitedInternetLinkStyleProps~xPropertySet~setPropertyValue("CharUnderline",          -
            box("short", bsf.getConstant("com.sun.star.awt.FontUnderline", "NONE")))

  -- set a hyperlink
Textprops~setPropertyValue("HyperLinkURL", "http://api.openoffice.org/")
xText~insertString(xTextCursor, "module star", .false)

::requires UNO.cls
```

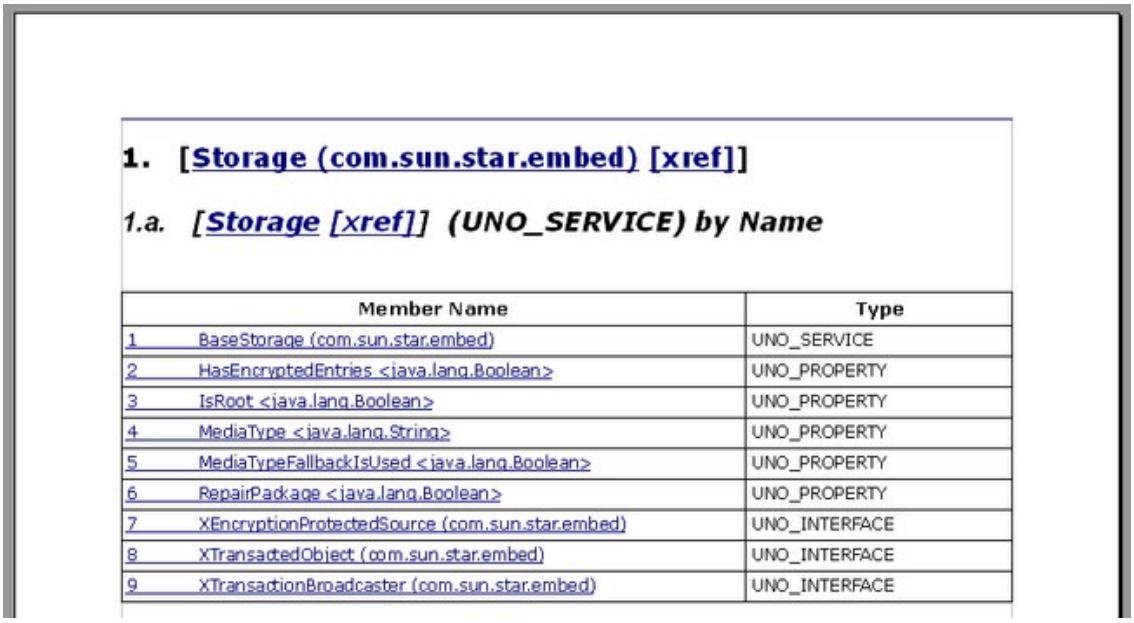
Code 42: Deactivate hyperlink underlining

The next two figures show the table of content with underlining of the hyperlinks and without. In the second picture the hyperlinks are still colored blue so that they are marked and recognized as hyperlinks.

Figure 72: Table of content underlined hyperlinks

Figure 73: Table of content hyperlinks not underlined

The following two figures show how the created document looks like with underlined hyperlinks and with no underlining. When there is no underlining of the hyperlinks the document is easier to read and well arranged. The hyperlinks are not underlined anymore but they are still colored in blue and executable.

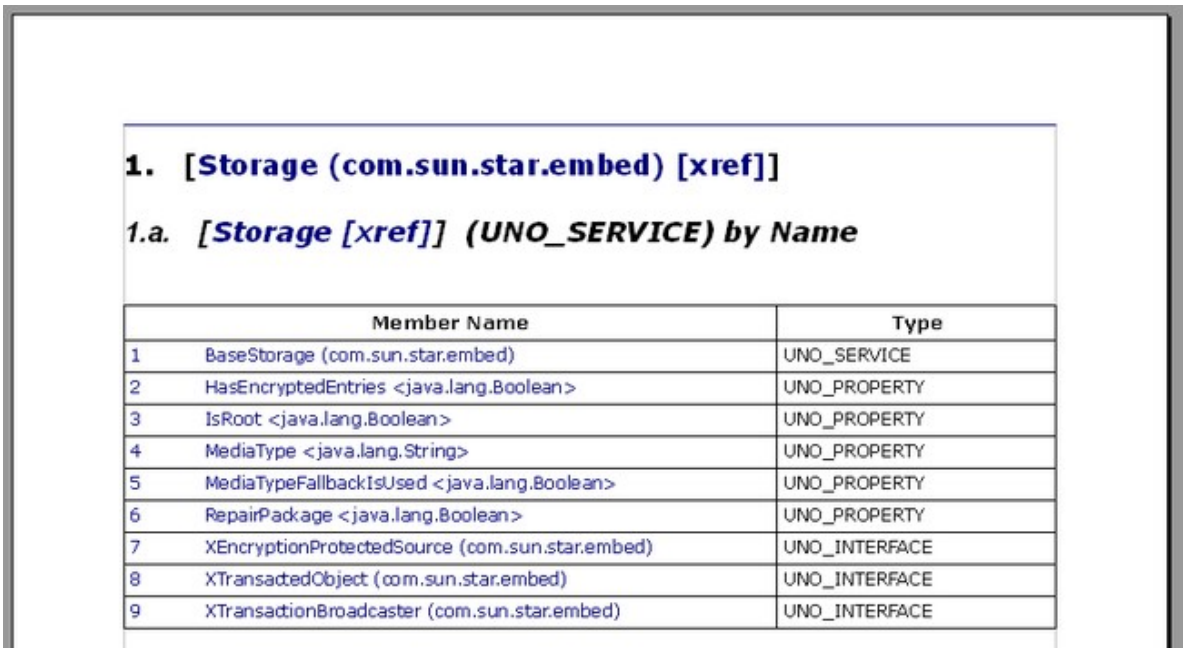


1. [Storage (com.sun.star.embed) [xref]]

1.a. [Storage [xref]] (UNO_SERVICE) by Name

	Member Name	Type
1	BaseStorage (com.sun.star.embed)	UNO_SERVICE
2	HasEncryptedEntries <java.lang.Boolean>	UNO_PROPERTY
3	IsRoot <java.lang.Boolean>	UNO_PROPERTY
4	MediaType <java.lang.String>	UNO_PROPERTY
5	MediaTypeFallbackIsUsed <java.lang.Boolean>	UNO_PROPERTY
6	RepairPackage <java.lang.Boolean>	UNO_PROPERTY
7	XEncryptionProtectedSource (com.sun.star.embed)	UNO_INTERFACE
8	XTransactedObject (com.sun.star.embed)	UNO_INTERFACE
9	XTransactionBroadcaster (com.sun.star.embed)	UNO_INTERFACE

Figure 74: Underlined hyperlinks



1. [Storage (com.sun.star.embed) [xref]]

1.a. [Storage [xref]] (UNO_SERVICE) by Name

	Member Name	Type
1	BaseStorage (com.sun.star.embed)	UNO_SERVICE
2	HasEncryptedEntries <java.lang.Boolean>	UNO_PROPERTY
3	IsRoot <java.lang.Boolean>	UNO_PROPERTY
4	MediaType <java.lang.String>	UNO_PROPERTY
5	MediaTypeFallbackIsUsed <java.lang.Boolean>	UNO_PROPERTY
6	RepairPackage <java.lang.Boolean>	UNO_PROPERTY
7	XEncryptionProtectedSource (com.sun.star.embed)	UNO_INTERFACE
8	XTransactedObject (com.sun.star.embed)	UNO_INTERFACE
9	XTransactionBroadcaster (com.sun.star.embed)	UNO_INTERFACE

Figure 75: Hyperlinks with no underlining

4.9 Table of Content

For the creation of a table of content the "`com.sun.star.text.ContentIndex`" `service` is needed. It can be instanced on the `MultiServiceFactory` with the `createInstance` command. Then the properties of the table of content can be accessed by setting `XPropertySet` to the before created content index [OOOAPIf] [OOOCSBa] [OOOCSBb].

Next the `XStyleFamiliesSupplier` has to be accessed so that the display-format of the table of content entries can be changed. After receiving the `XstyleFamiliesSupplier`, its `method` `getStyleFamilies` is called. This `method` passes the argument `ParagraphStyles` to access the paragraph styles. Next the heading style is changed with the `property` `ParaStyleName`.

After setting the paragraph styles, the properties of the content index can be changed. The code below level is set to three so only three levels are shown. If a paragraph style is set to "`Heading 4`" it will not be shown in the table of content. In this code snippet the level is set to two [OOOCSBb] [OOOAPIh].

Then the properties of a table of content are changed. First the `XPropertySet interface` is accessed. Then an array with all properties is created. This array is a three dimensional array with 5 slots. To set the properties, the command `uno.createProperty` is used. This command creates a `property value object` and returns its name and value [RGF08b]. The properties are set in the following order, so that the table of content gets the required format. First the `TokenHyperlinkStart` is set. This is important, because the text in the table of content should be hyperlinked with the headlines in the document. On the second place the `TokenEntryText` is set. After that the `TokenHyperlinkEnd` is set to announce that the remaining text has no hyperlink. The residual properties are the `TokenTabStop` and the `PageNumber`. The `TokenTabStop` has to be right aligned, so that the page numbers are displayed on the right frame of the document. Therefore the `property TabStopRightAligned` is set true. Furthermore the `TokenTabStop property TabStopFillCharacter` gets a "." assigned, so that the blank between the entry text and the page number is filled with points. After defining the properties, the level of the content index properties is set to three [AFHMMPd] [OOOD08b] [OOOCSBb].

In the next step the title is set to "Table Of Content" [OOOCSBa]. The `property Title` is contained in the `service BaseIndex` [OOOAPIh]. Furthermore the table of content is set protected with the `property IsProtected` from the `service BaseIndex` [OOOAPIh] [OOOCSBb].

It can also be specified, if the document index is created from outline. Here this `property` is set to true with `.java.true`. [OOOAPIg] To allow this, a `property` can be set with the command `.java.true`. It has to be defined as local variable at the begin of the program otherwise the command `box("Boolean", .true)` has to be used instead of `.java.true` [AFHMMPb].

After creating the content of the document, the table of content has to be updated with the command `update()` [OOOCSBa] [OOOCSBb].

```
/* creation of a table of content*/

/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                              -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)
```

```
xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet

  -- some text with Heading 1 style so the text is shown in the TOC
xPropertySet~setPropertyValue("ParaStyleName","Heading 1")
xText~insertString(xTextCursor,"Headline",.false)
xDMsf = xTextDocument~XMultiServiceFactory

 -- create TOC
contentInd = xDMsf~createInstance("com.sun.star.text.ContentIndex")
contentProps = contentInd~XPropertySet

 -- create two-dimensional array for the TOC properties
m1=5   /* three PropertyValue pairs */
m2=2   /* two PropertyValues        */
 /* create two-dimensional array of type PropertyValue */
propsToc = bsf.createArray(.UNO~PropertyValue, m1, m2)
do i1=1 to m1
   do i2=1 to m2
      propsToc[i1,i2]=.uno~propertyValue~new - create an assign PropertyValue object

      if i2=2 then  /* set companion PropertyValue to default value */
      do
         propsToc[i1,2]~name="CharacterStyleName"
         propsToc[i1,2]~value=""
      end
   end
end

 -- now set the property values
propsToc[1,1]~name  = "TokenType"
propsToc[1,1]~value = "TokenHyperlinkStart"
propsToc[2,1]~name  = "TokenType"
propsToc[2,1]~value = "TokenEntryText"
propsToc[3,1]~name  = "TokenType"
propsToc[3,1]~value = "TokenHyperlinkEnd"
propsToc[4,1]~name  = "TokenType"
propsToc[4,1]~value = "TokenTabStop"
propsToc[4,2]~name  = "TabStopRightAligned"
propsToc[4,2]~value = box("boolean", .true)
propsToc[5,1]~name  = "TokenType"
propsToc[5,1]~value = "TokenPageNumber"

contentProps~setPropertyValue("CreateFromOutline",box("boolean", .true))
contentProps~setPropertyValue("Title","Table of Content")
contentProps~setPropertyValue("IsProtected",box("boolean", .false))

 -- add the properties to the TOC
LevelFormat = contentProps~getPropertyValue("LevelFormat")
LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(1, propsToc)

 -- set text content to TOC and format it
contentTextContent = contentInd~xTextContent()
contentProps~setPropertyValue("Level",box("short", 2))

 -- insert TOC
xText~insertTextContent(xText, contentTextContent, .false )
 -- update TOC
contentTextContent~XDocumentIndex~update()

::requires UNO.cls
```

Code 43: “Table of Content”

4.10 Chapter Numbering

In this chapter it is described how chapter numbering can be automated. This makes chapter numbering easier because if a text is defined as a headline the numbering will be set automatically. The count will be updated every time a text is defined as headline. If a text is defined as headline between two headlines the numbering will be refreshed and increased automatically of all following headlines. Another advantage is, that the numbering type can be set for each headline level. So each level can have another numbering type.

In this example, a table of content like in the chapter 4.8 is created first. Afterwards the chapter numbering for the document is created. Hence the `interface` `XChapterNumberingSupplier` is accessed over the `xWriterDocument`. This `interface` includes the `method` `getChapterNumberingRules`. Then an array for the property values with three entries is created with the command `bsf.createArray()` [OOOAPIaf] [OOOD08e]. Then the `aChapterNumberingArray` array is filled with the required properties: the `HeadingStyleName` is set to the heading which should be changed for example "`Heading 1`", the `NumberingType` is set to the passed `numberingType` for example `chars_lower_letter`, and the `Suffix` is set to "`.`". After setting these `properties` the array replaces the requested index of the chapter numbering rules with the `method` `replaceByIndex` of the `interface` `xIndexReplace` [OOOAPIi] [OOOD08e]. The numbering type for the first heading gets as numbering type the `property` `chars_upper_letter`. The numbering is set to A,B,C… . The numbering type for the second and third heading is set to `Arabic` so Arabic numbers are used.

Next the `property` value of the table of content is set to customize the layout of the table of content.

```
call bsf.import "com.sun.star.text.ControlCharacter", "ctlChar"
call bsf.import "com.sun.star.style.NumberingType", "numTypes"
say "ROMAN_UPPER="pp(.numTypes~roman_upper)
say "---"

/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()    -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                              -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)
```

```
xDMsf = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet

short3=box("short", 3)  -- Java Short object representing the value "3"

xChapterNumberingSupplier = xTextDocument~XChapterNumberingSupplier
xChapterNumberingRules = xChapterNumberingSupplier~getChapterNumberingRules()

--  we need to use a Java array of type PropertyValue for three property values
aChapterNumberingArray=bsf.createArray(.uno~propertyValue,3)

aChapterNumberingArray[1]=uno.createProperty("HeadingStyleName", "Heading 1")
numberingType=.numTypes~roman_upper    -- set numbering type to roman_upper
aChapterNumberingArray[2]=uno.createProperty("NumberingType",    box("short",numberingType))
aChapterNumberingArray[3]=uno.createProperty("Suffix",           ". ")
xChapterNumberingRules~xIndexAccess~xIndexReplace~replaceByIndex(0,aChapterNumberingArray)
aChapterNumberingArray[1]=uno.createProperty("HeadingStyleName", "Heading 2")
numberingType=.numTypes~ARABIC    - set numbering type to ARABIC
aChapterNumberingArray[2]=uno.createProperty("NumberingType",    box("short",numberingType))
aChapterNumberingArray[3]=uno.createProperty("Suffix",           ". ")

xChapterNumberingRules~xIndexAccess~xIndexReplace~replaceByIndex(1,aChapterNumberingArray)

aChapterNumberingArray[1]=uno.createProperty("HeadingStyleName", "Heading 3")
numberingType=.numTypes~roman_upper    -- set numbering type to roman_upper
aChapterNumberingArray[2]=uno.createProperty("NumberingType",    box("short",numberingType))
aChapterNumberingArray[3]=uno.createProperty("Suffix",           ". ")
xChapterNumberingRules~xIndexAccess~xIndexReplace~replaceByIndex(2,aChapterNumberingArray)

  -- TOC
contentInd = xDMsf~createInstance("com.sun.star.text.ContentIndex")
contentProps = contentInd~XPropertySet

xFamiliesSupplier = xTextDocument~XStyleFamiliesSupplier
xParaStyleFamily = xFamiliesSupplier~getStyleFamilies~getByName("ParagraphStyles")~XNameContainer

contentProps~setPropertyValue("CreateFromOutline", box("Boolean", .true))
contentProps~setPropertyValue("IsProtected",       box("Boolean", .true))
contentProps~setPropertyValue("Level",             short3)
contentProps~setPropertyValue("Title",             "Table Of Content")


contentTextContent = contentInd~xTextContent()
contentIndProps    = contentTextContent~XPropertySet

  /* create two-dimensional array of type PropertyValue */
propsToc = bsf.createArray(.UNO~PropertyValue, 6, 3)

  /* now set the property values */
propsToc[1,1]=uno.createProperty("TokenType",           "TokenEntryNumber")
propsToc[1,2]=uno.createProperty("CharacterStyleName",  "")

propsToc[2,1]=uno.createProperty("TokenType",           "TokenHyperlinkStart")
propsToc[2,2]=uno.createProperty("CharacterStyleName",  "")

propsToc[3,1]=uno.createProperty("TokenType",           "TokenEntryText")
propsToc[3,2]=uno.createProperty("CharacterStyleName",  "")

propsToc[4,1]=uno.createProperty("TokenType",           "TokenHyperlinkEnd")
propsToc[4,2]=uno.createProperty("CharacterStyleName",  "")

propsToc[5,1]=uno.createProperty("TokenType",           "TokenTabStop")
propsToc[5,2]=uno.createProperty("TabStopRightAligned", box("Boolean", .true))
propsToc[5,3]=uno.createProperty("TabStopFillCharacter",".")

propsToc[6,1]=uno.createProperty("TokenType",           "TokenPageNumber")
propsToc[6,2]=uno.createProperty("CharacterStyleName",  "")

  -- set the level of the Toc
contentIndProps~setPropertyValue("Level",               short3)
  -- set the format with the above specified properties of the 3 levels
LevelFormat = contentProps~getPropertyValue("LevelFormat")
```

```
LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(1, propsToc)
LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(2, propsToc)
LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(3, propsToc)

  -- insert TOC
xText~insertTextContent(xText, contentTextContent, .false )

  -- insert headlines
xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)

xPropertySet~setPropertyValue("ParaStyleName","Heading 1")
xPropertySet~setPropertyValue("ParaBackColor", box("int", "CECEF6"x ~c2d))
xText~insertString(xText~getEnd, "Headline1",.true)
xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)
xPropertySet~setPropertyValue("ParaStyleName","Heading 2")
xText~insertString(xText~getEnd, "Headline2",.true)
xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)
xPropertySet~setPropertyValue("ParaStyleName","Heading 3")
xText~insertString(xText~getEnd, "Headline3",.true)

  -- update table of content
contentTextContent~XDocumentIndex~update()

::requires UNO.cls
```

Code 44: Chapter numbering

The following figure shows the chapter numbering of three layers. For the first headline level the numbering type `roman_upper` is chosen. For the second headline level the numbering type is changed to `ARABIC`. The third level also uses `ROMAN_UPPER` as numbering type.

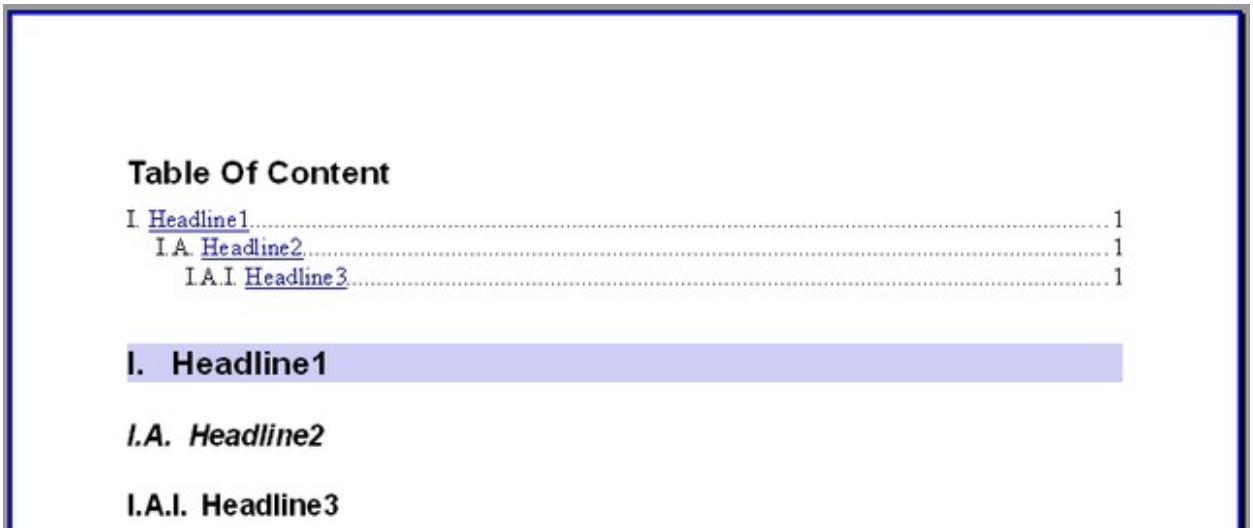


Figure 76: Chapter numbering

4.11 Table of content font change

The next code snippet points out how the font style of the table of content can be changed. If the font is not changed, the quoted headlines do not have the same font like the font which is used in the text. If the table of content should have another font it can be changed to the font which is required. Also the font height can be changed at the different headline levels.

First the table of content is created like in the chapter 4.8. Then the styles can be changed. Therefore the routine `ParaStyle` is created. This routine needs the following arguments: the `heading style, contentProps, xParaStyleFamily` and the `font height` and the `font` [OOOAPIg] [OOOD08b] [OOOCSBa]. In the `ParaStyle` routine first the heading style has to be accessed. This is achieved with the `method getPropertyValue` which gets the paragraph style `property` from the content index. Then the properties of this heading style can be set with the `setPropertyValue` command. This `method` needs two arguments: the `property` which should be changed and the value it should get for example `xHeadingStyleProps~setPropertyValue("CharFontName",“Verdana”)`. This code line changes the font to `Verdana` [OOOCSBd] [OOOAPIh] [OOOCSBb].

```
call bsf.import "com.sun.star.text.ControlCharacter", "ctlChar"
call bsf.import "com.sun.star.style.NumberingType", "numTypes"

/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                              -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)

xDMsf = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet

short3=box("short", 3)      -- Java Short object representing the value "3"

xChapterNumberingSupplier = xTextDocument~XChapterNumberingSupplier
xChapterNumberingRules = xChapterNumberingSupplier~getChapterNumberingRules()

--  we need to use a Java array of type PropertyValue for three property values
aChapterNumberingArray=bsf.createArray(.uno~propertyValue,3)

aChapterNumberingArray[1]=uno.createProperty("HeadingStyleName", "Heading 1")
numberingType=.numTypes~roman_upper
aChapterNumberingArray[2]=uno.createProperty("NumberingType",    box("short",numberingType))
aChapterNumberingArray[3]=uno.createProperty("Suffix",           ". ")
xChapterNumberingRules~xIndexAccess~xIndexReplace~replaceByIndex(0,aChapterNumberingArray)

aChapterNumberingArray[1]=uno.createProperty("HeadingStyleName", "Heading 2")
```

```
numberingType=.numTypes~roman_upper
aChapterNumberingArray[2]=uno.createProperty("NumberingType",    box("short",numberingType))
aChapterNumberingArray[3]=uno.createProperty("Suffix",           ". ")

xChapterNumberingRules~xIndexAccess~xIndexReplace~replaceByIndex(1,aChapterNumberingArray)

aChapterNumberingArray[1]=uno.createProperty("HeadingStyleName", "Heading 3")
numberingType=.numTypes~roman_upper
aChapterNumberingArray[2]=uno.createProperty("NumberingType",    box("short",numberingType))
aChapterNumberingArray[3]=uno.createProperty("Suffix",           ". ")
xChapterNumberingRules~xIndexAccess~xIndexReplace~replaceByIndex(2,aChapterNumberingArray)

  -- TOC
contentInd = xDMsf~createInstance("com.sun.star.text.ContentIndex")
contentProps = contentInd~XPropertySet

xFamiliesSupplier = xTextDocument~XStyleFamiliesSupplier
xParaStyleFamily = xFamiliesSupplier~getStyleFamilies~getByName("ParagraphStyles")~XNameContainer

  -- change the paragraph style
call ParaStyle "ParaStyleHeading", contentProps,xParaStyleFamily, 14, "Verdana"
call ParaStyle "ParaStyleLevel1", contentProps,xParaStyleFamily, 12, "Verdana"
call ParaStyle "ParaStyleLevel2", contentProps,xParaStyleFamily, 10 , "Verdana"
call ParaStyle "ParaStyleLevel3", contentProps,xParaStyleFamily, 8 , "Verdana"

contentProps~setPropertyValue("CreateFromOutline", box("Boolean", .true))
contentProps~setPropertyValue("IsProtected",       box("Boolean", .true))
contentProps~setPropertyValue("Level",             short3)
contentProps~setPropertyValue("Title",             "Table Of Content")


contentTextContent = contentInd~xTextContent()
contentIndProps    = contentTextContent~XPropertySet

  /* create two-dimensional array of type PropertyValue */
propsToc = bsf.createArray(.UNO~PropertyValue, 5, 3)

  /* now set the property values */
propsToc[1,1]=uno.createProperty("TokenType",           "TokenHyperlinkStart")
propsToc[1,2]=uno.createProperty("CharacterStyleName",  "")

propsToc[2,1]=uno.createProperty("TokenType",           "TokenEntryText")
propsToc[2,2]=uno.createProperty("CharacterStyleName",  "")

propsToc[3,1]=uno.createProperty("TokenType",           "TokenHyperlinkEnd")
propsToc[3,2]=uno.createProperty("CharacterStyleName",  "")

propsToc[4,1]=uno.createProperty("TokenType",           "TokenTabStop")
propsToc[4,2]=uno.createProperty("TabStopRightAligned", box("Boolean", .true))
propsToc[4,3]=uno.createProperty("TabStopFillCharacter",".")

propsToc[5,1]=uno.createProperty("TokenType",           "TokenPageNumber")
propsToc[5,2]=uno.createProperty("CharacterStyleName",  "")

  -- set the level of the Toc
contentIndProps~setPropertyValue("Level",               short3)
  -- set the format with the above specified properties of the 3 levels
LevelFormat = contentProps~getPropertyValue("LevelFormat")
LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(1, propsToc)
LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(2, propsToc)
LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(3, propsToc)

  -- insert TOC
xText~insertTextContent(xText, contentTextContent, .false )

  -- insert headlines
xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)

xPropertySet~setPropertyValue("ParaStyleName","Heading 1")
xPropertySet~setPropertyValue("ParaBackColor", box("int", "CECEF6"x ~c2d))
xText~insertString(xText~getEnd, "Headline1",.true)
xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)
xPropertySet~setPropertyValue("ParaStyleName","Heading 2")
xText~insertString(xText~getEnd, "Headline2",.true)
xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)
xPropertySet~setPropertyValue("ParaStyleName","Heading 3")
```

```
xText~insertString(xText~getEnd, "Headline3",.true)

  -- update table of content
contentTextContent~XDocumentIndex~update()

::requires UNO.cls

  -- routine for setting the paragraph style
::routine ParaStyle
  use arg HeadingStyle, contentProps, xParaStyleFamily, paraCharHeight, font
   sParaStyleHeading = contentProps~getPropertyValue(HeadingStyle)
   xHeadingStyleProps = xParaStyleFamily~getByName(sParaStyleHeading)~xPropertySet
   xHeadingStyleProps~setPropertyValue("CharFontName", font)
   xHeadingStyleProps~setPropertyValue("CharHeight", box("integer", paraCharHeight))
```

Code 45: “Table of Content” font change

The next figure shows a table of content with an individual defined font style and font height for each headline level.

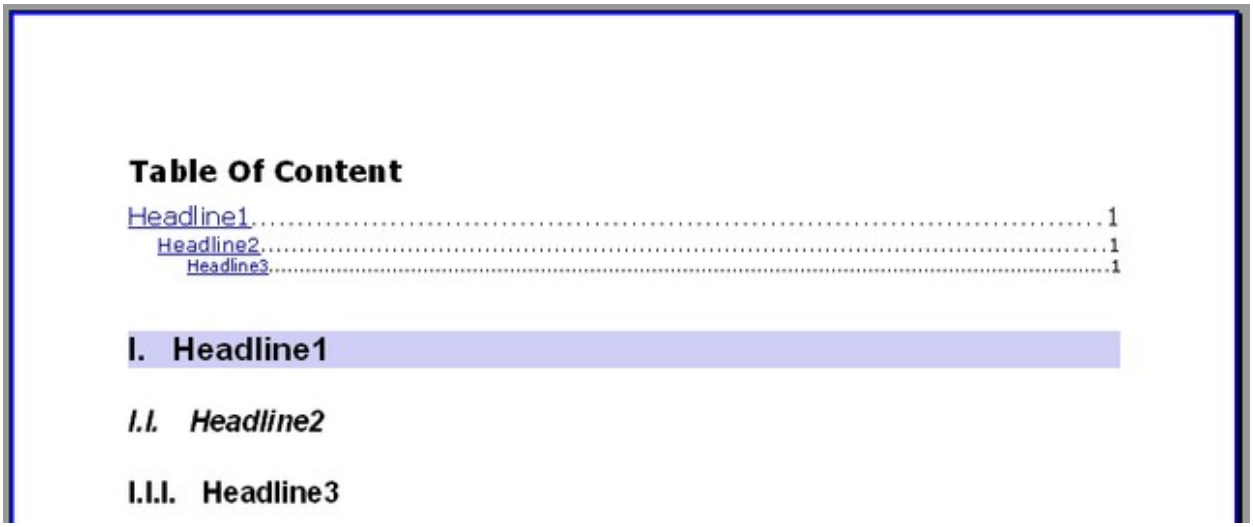


Figure 77: “Table of Content” font change

4.12 Document Indexes Demo

This program has been transcripted from Rony G. Flatscher from the `DocumentIndexesDemo.java` program which can be found in the achieve of the Open Office mailing list achieve [OOOOMLa]. The `DocumentIndexesDemo.java` program is provided by Ariel Constenla-Haile. The entire program code for this example can be found in the Attachment chapter 7.

The `DocumentIndexesDemo.java` program shows how a table of content is created step by step. After each step a pause is made for the user to have a look at the program code which code had been executed.

First some text sections with headlines of different heading levels are added to the document. The text is created automatically. The following figure shows one paragraph and two created headlines of different heading levels.

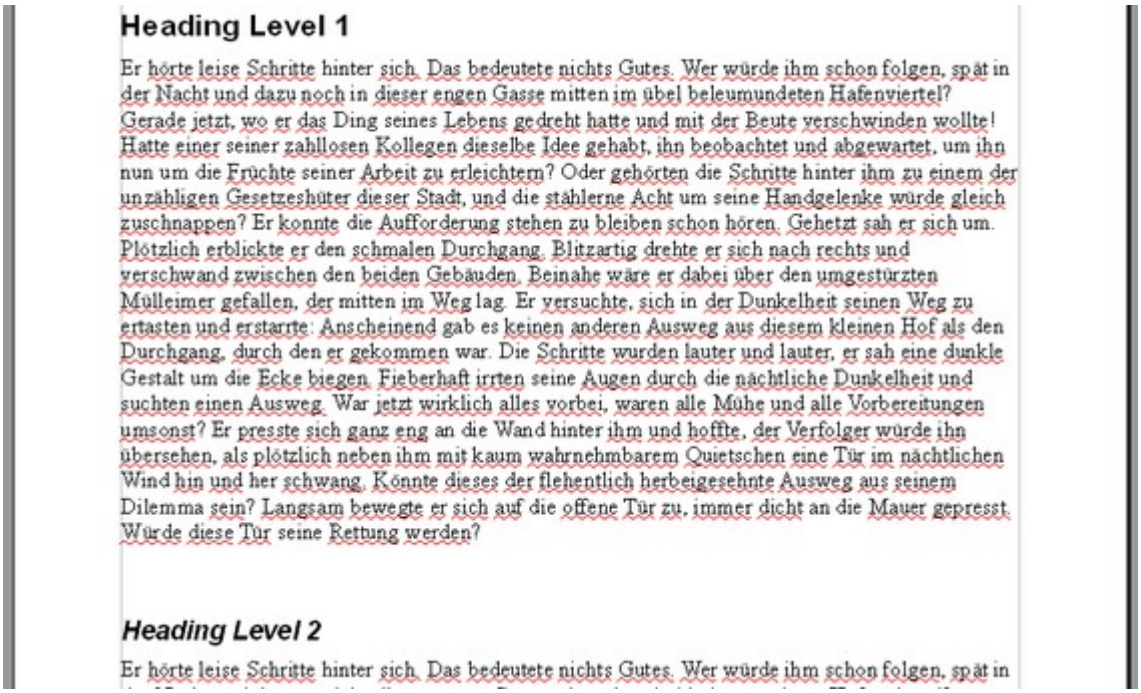


Figure 78: Text sections

Then a chapter numbering is added to the headlines. The following figure displays the before created document with the created chapter numbering.

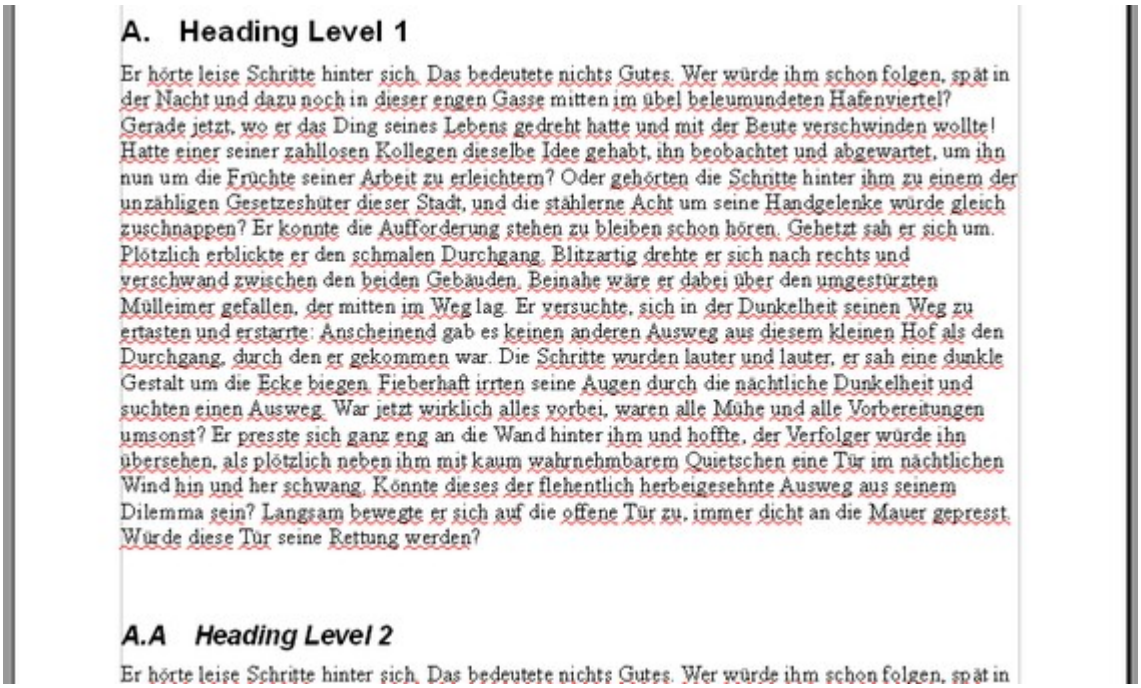


Figure 79: Chapter Numbering

The following program code shows how the chapter numbering has been added to the document.

```
xChapterNumberingSupplier = .docIdx~m_xTextDocument~XChapterNumberingSupplier
if xChapterNumberingSupplier = .nil then
    return

xChapterNumberingRules = xChapterNumberingSupplier~getChapterNumberingRules
aChapterNumberingList = uno.createArray(.uno~PropertyValue, 3)
NumberingType=uno.loadClass("com.sun.star.style.NumberingType")

-------------------------------------------------------------------
/*
 * HeadingStyleName
 * Contains the name of the paragraph style
 * that marks a paragraph as a chapter heading.
 */
aHeadingStyleName = uno.createProperty("HeadingStyleName", "Heading 1");
aChapterNumberingList[1]=aHeadingStyleName

/*
 * NumberingType
 *
 * Determines the type of the numbering defined
 * in com.sun.star.style.NumberingType.
 */
aNumberingType = uno.createProperty("NumberingType", NumberingType~ROMAN_UPPER)
aChapterNumberingList[2]=aNumberingType

/*
 * Suffix
 * the suffix of the numbering symbol.
 */
aSuffix = uno.createProperty("Suffix", ". ")
aChapterNumberingList[3]=aSuffix

xChapterNumberingRules~replaceByIndex( 0, aChapterNumberingList )

-------------------------------------------------------------------
-- create a new array of PropertyValues
```

```
  aChapterNumberingList = uno.createArray(.uno~PropertyValue, 4)

  -- HeadingStyleName
  aHeadingStyleName~Value = "Heading 2";
  aChapterNumberingList[1]=aHeadingStyleName

  -- NumberingType
  aNumberingType~Value = NumberingType~ARABIC;
  aChapterNumberingList[2]=aNumberingType

  -- Suffix
  aSuffix~Value = " ";
  aChapterNumberingList[3]=aSuffix

  /*
   * ParentNumbering
   * Determines if higher numbering levels are included in the numbering,
   * for example, 2.3.1.2.
   */
  aParentNumbering = uno.createProperty("ParentNumbering", box("short",2))
  aChapterNumberingList[4]=aParentNumbering

  xChapterNumberingRules~replaceByIndex( 1, aChapterNumberingList )

  -- we reuse the array!
  ------------------------------------------------------------------

  -- HeadingStyleName
  aHeadingStyleName~Value = "Heading 3"

  -- NumberingType
  aNumberingType~Value = NumberingType~ROMAN_LOWER

  -- Suffix
  aSuffix~Value = ") "

  -- ParentNumbering
  aParentNumbering~Value = box("short", 3)

  xChapterNumberingRules~replaceByIndex( 2, aChapterNumberingList )


say "in demoOutlineNumbering, *AFTER* defining header level formatting, look at the document ..."
call beep 1000,100
call sysSleep .05
call beep 1000,100
"pause"
```

Code 46: Create chapter numbering

Then the table of content is added to the document. It contains every headline of the document and every level of the table of content gets a different style. The following figure shows the table of content of the created document. The font weight of the second level is set to bold and the font color is set to pink of the third level.

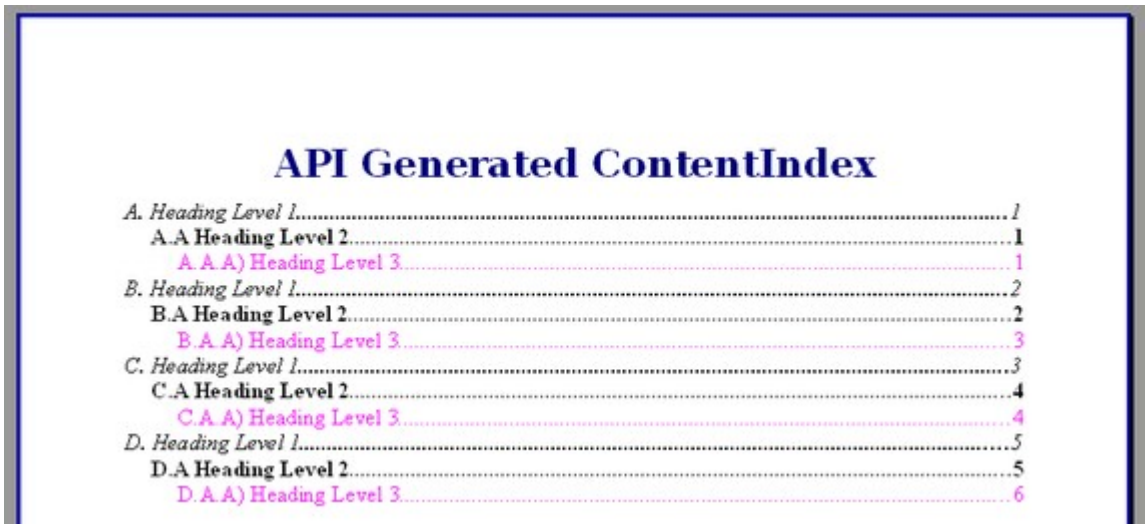


Figure 80: Table of Content

The following program code illustrates how the table of content displayed before was created. For each table of content level a different style is set. For the second level a custom style has been defined.

```
  xDocFactory = .docIdx~m_xTextDocument~XMultiServiceFactory
  oContentIndex = xDocFactory~createInstance("com.sun.star.text.ContentIndex")

  -- a unique name may help for later access
  xNamedIndex = oContentIndex~XNamed
  xNamedIndex~setName(.docIdx~CONTENT_INDEX1)

  -- access the set of properties
  xContentIndexProps = oContentIndex~XPropertySet

  -- "Title" is a property of every BaseIndex
  xContentIndexProps~setPropertyValue("Title", "API Generated ContentIndex")
  -- "IsProtected" is a property of every BaseIndex
  xContentIndexProps~setPropertyValue("IsProtected", box("bool",.true))
  -- "CreateFromOutline" is a ContentIndex property
  xContentIndexProps~setPropertyValue("CreateFromOutline", box("bool",.true))

  -- HOW TO CHANGE THE STYLES?
  ------------------------------------------------------------------/
  ------------------------------------------------------------------/
  --
  --  GETTER METHOD
  --  1. get the name of the style applied to the level we want
  --  2. get that style from the paragraph styles family
  --  3. modify it
  --
--*****************************************************************************
  xParaStyleFamily = .docIdx~m_xStyleFamilies~getByName("ParagraphStyles")~XNameContainer

  -- the service BaseIndex has properties to access the styles:
  --      ParaStyleHeading
  --      ParaStyleLevel1, ParaStyleLevel2 ... ParaStyleLevel10

  -- change the heading style
  sParaStyleHeading = xContentIndexProps~getPropertyValue("ParaStyleHeading")

  if xParaStyleFamily~hasByName(sParaStyleHeading) then
  do
      xHeadingStyleProps = xParaStyleFamily~getByName(sParaStyleHeading)~XPropertySet

      center=uno.loadClass("com.sun.star.style.ParagraphAdjust")~center
```

```
      xHeadingStyleProps~setPropertyValue("ParaAdjust", center)

      xHeadingStyleProps~setPropertyValue("CharHeight", box("int",20))
      xHeadingStyleProps~setPropertyValue("CharColor",  box("int",getRGBColor(0,0,125)))
      xHeadingStyleProps~setPropertyValue("CharFontName", "DejaVu Serif;Galatia SIL");
  end

  -- change the style for level 1
  sParaStyleLevel1 = xContentIndexProps~getPropertyValue("ParaStyleLevel1")

  if xParaStyleFamily~hasByName(sParaStyleLevel1) then
  do
      xParaStyleLevel1 = xParaStyleFamily~getByName(sParaStyleLevel1)~XPropertySet

      italic=uno.loadClass("com.sun.star.awt.FontSlant")~italic
      xParaStyleLevel1~setPropertyValue("CharPosture", italic)
  end

  -- change the style for level 3
  sParaStyleLevel3 = xContentIndexProps~getPropertyValue("ParaStyleLevel3")

  if xParaStyleFamily~hasByName(sParaStyleLevel3) then
  do
      xParaStyleLevel3 = xParaStyleFamily~getByName(sParaStyleLevel3)~XPropertySet
      xParaStyleLevel3~setPropertyValue("CharColor", box("int",getRGBColor(255,0,255)))
  end

  ------------------------------------------------------------------/
  --
  -- SETTER METHOD
  --  1. create a new custom style
  --  2. set that style at the index level you want
  -- create a new style for level 2
  sMyParaStyleName = "My Custom ContentIndex Style Level 2"

  if xParaStyleFamily~hasByName(sMyParaStyleName) then
      oNewParaStyleLevel2 = xParaStyleFamily~getByName(sMyParaStyleName)
  else
  do
      oNewParaStyleLevel2 = xDocFactory~createInstance("com.sun.star.style.ParagraphStyle")
      xParaStyleFamily~insertByName(sMyParaStyleName, oNewParaStyleLevel2)
  end

  xParaStyleLevel2 = oNewParaStyleLevel2~XPropertySet

  xParaStyleLevel2~setPropertyValue("ParaLeftMargin", box("int",500))
  bold=uno.loadClass("com.sun.star.awt.FontWeight")~bold
  xParaStyleLevel2~setPropertyValue("CharWeight", bold)

  -- SET IT AT THE INDEX LEVEL
  xContentIndexProps~setPropertyValue("ParaStyleLevel2", sMyParaStyleName)

  ------------------------------------------------------------------

  xDocumentIndex = oContentIndex~XDocumentIndex

  -- insert the index at the start of the document
  call insertTextContent .docIdx~m_xTextDocument~getText~getStart, xDocumentIndex
  xDocumentIndex~update


/* ***************************************************************************/
::routine insertTextContent
  use arg xTextRange, xContent
  xTextRange~getText~insertTextContent(xTextRange, xContent, .false)
```

Code 47: Defining table of content

4.13 Set document hidden

When a document is generated with a lot of content, the creation takes some time. To accelerate the creation time the document can be set hidden and only shown when the document is finished.

To set the document hidden, the properties of the Writer document has to be changed. Hence an array with all properties is created with the command `bsf.createArray()` [AFHMMPd]. This command gets two elements, the first is the `property` and the second the size. These `properties` are set with the following commands: `props[1]~Name` and `props[1]~Value`. To set the document invisible, the `property` "`Hidden`" should be set `true`. So during the building time it is set hidden and after finishing the creation it is shown. After setting the properties, the Writer document can be loaded from the stated URL with the defined properties. Hence the `loadComponentFromURL` command is used to load a component specified by an URL into a defined frame. Below, the whole program code for these steps is displayed [OOOAPIb].

Next the main `interface` `XTextDocument` of the text document is accessed [OOOAPIe]. Then the controllers of the `xWriterDocument` are locked, if they are not locked yet. With the `hasControllersLocked` `method` it can be checked, if the controllers are currently locked. If they are not, one can use the `lockControllers` `method` [OOOAPIc].

For setting the Writer document visible, a series of commands is needed. First the `method` `getCurrentController` of the `interface` `XModel` is accessed [OOOAPIc]. Then the frame has to be received via the `getFrame` `method` of the `XController` `interface` [OOOAPIau]. Afterwards the container window of the frame has to be retrieved with the `method` `getContainerWindow` of the `interface` `XFrame` [OOOAPIav]. Finally the Writer document can be set visible with the `method` `setVisible` which needs the argument `.true` of the `interface` `XWindow` [OOOAPIaw] [AFHMMPk]. After setting the Writer document visible, it is checked if the controllers are locked. If they are locked, they are set unlocked with the `method` `unLockControllers` of the `interface` `XModel` [OOOAPIc].

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface
```

```
/* open the blank *.odt - file */
url = "private:factory/swriter"
/* set document properties: document title and window hidden */
props = bsf.createArray(.UNO~PropertyValue, 1)
props[1] = .UNO~PropertyValue~new
-- set document hidden that the creation time is shorter (don't show Window)
props[1]~Name = "Hidden"
props[1]~Value =  box("boolean", .true)
xWriterComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0, props)

xWriterDocument = xWriterComponent~XTextDocument
-- do not update screen
if \xWriterDocument~hasControllersLocked then xWriterDocument~lockControllers

xText = xWriterDocument~getText()
xTextCursor = xText~createTextCursor()
xPropertySet = xTextCursor~xPropertySet
xDMsf = xWriterDocument~XMultiServiceFactory

xText~insertString(xTextCursor, "Table", .false)
/* create the table */
xTextTable = xDMsf~createInstance("com.sun.star.text.TextTable")~XTextTable
xTextTable~initialize(10, 2) -- initialize the table
/* insert TextTable in the Text */
xText~insertTextContent(xTextCursor, xTextTable, .false)

-- set WriterDocument visible after creation
xWriterDocument~XModel~getCurrentController()~XController~getFrame()   -
               ~Xframe~getContainerWindow()~XWindow~setVisible(.true)
-- now update screen again
if xWriterDocument~hasControllersLocked then xWriterDocument~unLockControllers

::requires UNO.cls
```

Code 48: Set document hidden

4.14 Store and print document

The document is saved as `.odt` file with the `storeAsURL` `method` of the `XStorable` `interface`. This `method` stores the created document to the specified URL. The URL is defined with the routine `makeURL` [OOOAPIar]. The routine `makeUrl` returns the current directory where the document should be saved. The current directory can be queried with the command `directory`. This command returns the current directory when no other directory is specified [AFHMMPx]. The URL is build with the directory command and the argument which is passed to the `makeUrl` routine.

Next the document is printed with the default printer. This is possible with the `method` `print` from the `XPrintable` `interface`. This `method` prints the specified document [OOOAPIas] [OOOCSBj].

Afterwards the current created Writer document is saved as `*.pdf` file. Therefore the `XStorable` `interface` has to be accessed [OOOAPIar]. To convert the Writer document into a `*.pdf` file, an array for the properties is needed. In this array the `FilterName` and

the `CompressMode` is set. Then the URL how the created file should be named is created and the created `pdf`-file is stored to the same directory as the `odt`-file [OOOCSBk].

```
/* Retrieve the Desktop object, we need its XComponentLoader
interface to load a new document*/
oDesktop         = UNO.createDesktop()     -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader -- get componentLoader interface
/* open the blank file */
url = "private:factory/swriter"
xWriterComponent = xComponentLoader                                                -
                   ~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)


xDocumentFactory = xWriterComponent~XMultiServiceFactory
xTextDocument = xWriterComponent~XTextDocument
xText = xTextDocument~getText()
xTextCursor = xText~createTextCursor()
Textprops = xTextCursor~xPropertySet

url=makeURL("createdDocument.odt")

-- save created document as odt-file
xWriterComponent~XStorable~storeAsURL(url, .UNO~noProps)

xPrintable = xWriterComponent~XPrintable
xPrintable~print(.UNO~noProps) -- print the file to default printer

-- create a pdf-file from the created odt file
xStorable = xWriterComponent~XStorable -- get xStorable interface

-- convert the document to a pdf-file
props = bsf.createArray(.UNO~propertyValue, 2)
props[1] = .UNO~PropertyValue~new
props[1]~Name = "FilterName"
props[1]~Value = "writer_pdf_Export"
props[2] = .UNO~PropertyValue~new
props[2]~Name = "CompressMode"
props[2]~Value = 2
storeURL = substr(url, 1, lastpos(".", url)) || "pdf" -- create output file name
xStorable~storeToUrl(storeURL, props) -- store the file with props to URL

::requires UNO.cls

::routine makeUrl
  return  ConvertToURL(directory() || "\" || arg(1))
```

Code 49: Store and print document

5 Roundup and Outlook

In this paper the program API-Viewer is described. The API-Viewer gives the user the possibility to start queries about UNO IDL's to get their members and so see the structure of the UNO IDL elements and how they are composed. To some UNO IDL's like `UNO_ENUM` or `UNO_CONSTANTS` additional important information is given like the values.

The information provided by the API-Viewer is helpful for Open Office users who want to automate Open Office or get an overview about the structure of Open Office. The provided information of the API-Viewer can be used to get a picture about how components in Open Office work together or to create information papers about the UNO IDL components. The API-Viewer can be started via a graphical display the `frontend.rex` program or it can be called within other programs. This is an important feature because users can use the result of the API-Viewer for their own objects and include it in their programs.

Very important for the creation of the diploma thesis were the sources provided from the Open Office homepage and the Open Office community. These sources were helpful to create the formatting and layout of the query output in Open Office Writer. The Open Office API and the Developer's Guide are important to get an overview how Open Office can be automated, because they give information and examples about the basic components. For questions about custom programs, implementation of different elements and problems by implementing them, the Open Office Mailing list is very helpful. The Open Office Mailing list members are from the Open Office community and they share their knowledge they gather during programming with Open Office. Another place to find examples about automation of Open Office, is the Open Office Code Snippet Base where users provide short examples how components can be used to get the requested output. These snippets are created with different programming languages but usually they can be implemented with other programming languages without problems.

The programs `frontend.rex` and `createApiInfo.rex` will be enhanced by professor Rony G. Flatscher. This development will further increase the applicability of the API-Viewer. It

is planned, that the enhanced program will be included in a regularly BSF4Rexx distribution and made available for everyone after installing BSF4Rexx.

Summarizing the API-Viewer is a great tool for programmers and helps to get an overview about the Open Office structure, its components and how they work together.

6 References

[AFHMMPa] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, The Directory Class page 229 – 235, 25 October 2007

[AFHMMPb] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, The Local Directory (.LOCAL) page 392, 25 October 2007

[AFHMMPc] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, The Nil Object (.NIL) page 391, 25 October 2007

[AFHMMPd] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, The Array Class page 208 – 219, 25 October 2007

[AFHMMPe] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, The FALSE Constant (.FALSE) page 391, 25 October 2007.

[AFHMMPf] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, isA page 118, 25 October 2007

[AFHMMPg] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, wordPos page 164, 25 October 2007

[AFHMMPh] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, CHANGESTR page 407, 25 October 2007

[AFHMMPi] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, VALUE page 450, 25 October 2007

[AFHMMPj] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, LEFT page 154, 25 October 2007

[AFHMMPk] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, The True Constant (.TRUE) page 391, 25 October 2007

[AFHMMPl] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, isA page 118, 25 October 2007

[AFHMMPm] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, hasIndex page 213, 25 October 2007

[AFHMMPn] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, ITERATE page 55, 25 October 2007

[AFHMMPo] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, equals page151, 25 October 2007

[AFHMMPp] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, toString page 218, 25 October 2007

[AFHMMPq] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, PARSE VAR page 62, 25 October 2007

[AFHMMPr] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, pos page 158, 25 October 2007

[AFHMMPs] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, The ENDOFLINE constant (.ENDOFLINE) page 391, 25 October 2007

[AFHMMPt] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, length page 154, 25 October 2007

[AFHMMPu] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, max page 156, 25 October 2007

[AFHMMPv] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, caselessCountStr 156, 25 October 2007

[AFHMMPw] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, DIRECTORY page 420, 25 October 2007

[AFHMMPx] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, USE page 78, 25 October 2007

[AFHMMPy] W. David Ashley, Rony G. Flatscher, Mark Hessling, Rick McGuire, Mark Miesfeld and Lee Peedin, Open Object Rexx Reference 3.2.0, putAll page 205, 25 October 2007

[Flat08a] Dr. Rony G. Flatscher, Automatisierung von Java Anwendungen (1) – Course slides, 2008 ,http://wi.wu-wien.ac.at/rgf/wu/lehre/autojava/material/foils/ , retrieved 31 August 2008

[Flat08b] Dr. Rony G. Flatscher, Automatisierungen von Java Anwendungen (7) – Course slides, 2008, http://wi.wu-wien.ac.at/rgf/wu/lehre/autojava/material/foils/ , retrieved 31 August 2008

[Java03a] Java API, JFrame, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JFrame.html, retrieved 12 September 2008

[Java03b] Java API, JMenuBar, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JMenuBar.html, retrieved 12 September 2008

[Java03c] Java API, JMenu, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JMenu.html, retrieved 12 September 2008

[Java03d] Java API, JMenuItem, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JMenuItem.html, retrieved 12 September 2008

[Java03e] Java API, JeditorPane, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JEditorPane.html, retrieved 12 September 2008

[Java03f] Java API, JTextComponent – setEditable, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/text/JTextComponent.html#setEditable(boolean), retrieved 12 September 2008

[Java03g] Java API, GridLayout, 2003, http://java.sun.com/j2se/1.4.2/docs/api/java/awt/GridLayout.html, retrieved 12 September 2008

[Java03h] Java API, Window – pack(), 2003 http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Window.html#pack(), retrieved 12 September 2008

[Java03i] Java API, Window – show(), 2003, http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Window.html#show(), retrieved 12 September 2008

[Java03j] JavaAPI, Window – toFront(), 2003, http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Window.html#toFront(), retrieved 12 September 2008

[Java03k] Java API, Window – dispose(), 2003, http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Window.html#dispose(), retrieved 12 September 2008

[Java03l] Java API, JFrame – getContentPane(), 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JFrame.html#getContentPane(), retrieved 12 September 2008

[Java03m] Java API, JLabel, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JLabel.html, retrieved 12 September 2008

[Java03n] Java API, JTextField, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JTextField.html, retrieved 12 September 2008

[Java03o] Java API, JComboBox, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JComboBox.html, retrieved 12 September 2008

[Java03p] Java API, JButton, 2003, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JButton.html, retrieved 12 September 2008

[Java03q] Java API, setToolTipText, http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JComponent.html#setToolTipText(java.lang.String), retrieved 07 January 2009

[JavaSun] Java-Downloads für alle Betriebssysteme, http://java.sun.com/javase/downloads/index.jsp, retrieved 23 September 2008

[JavaSuna] Java Documentation – Installation Notes, http://java.sun.com/j2se/1.4.2/install.html, retrieved 23 September 2008

[OOOorga] About US: Open Office.org, 2008, http://about.openoffice.org/index.html, retrieved 4 May 2008

[OOOorgb] Open Office.org 2 – Product Description, http://www.openoffice.org/product/, retrieved on 4 May 2008

[OOOorgc] Open Office.org – Software Development Kit, 2006, http://download.openoffice.org/2.4.0/sdk.html, retrieved on 13 September 2008

[OOOorgd] Open Office.org –quick download, 2008, http://de.openoffice.org/downloads/quick.html, retrieved 13 September 2008

[OOOAPI] Open Office.org, API, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html, retrieved 10 September 2008

[OOOAPIa] Open Office.org, API – service Desktop, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/frame/Desktop.html, retrieved 15 September 2008

[OOOAPIb] Open Office.org, API – interface XComponentLoader, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/frame/XComponentLoader.html#loadComponentFromURL, retrieved 15 September 2008

[OOOAPIc] Open Office.org, API – interface XModel, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/frame/XModel.html , retrieved 15 September 2008

[OOOAPId] Open Office.org, API – service ServiceManager, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/lang/ServiceManager.html, retrieved 15 September 2008

[OOOAPIe] Open Office.org, API - interface XTextDocument, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextDocument.html, retrieved 15 September 2008

[OOOAPIf] Open Office.org, API – service ContentIndex, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/ContentIndex.html, retrieved September 2008

[OOOAPIg] Open Office.org, API – interface XStyleFamiliesSupplier, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/style/XStyleFamiliesSupplier.html, retrieved 16 September 2008

[OOOAPIh] Open Office.org, API – service BaseIndex, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/BaseIndex.html, retrieved 16 September 2008

[OOOAPIi] Open Office.org, API – interface XIndexReplace, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/container/XIndexReplace.html, retrieved 16 September 2008

[OOOAPIj] Open Office.org, API – interface XText, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/XText.html, retrieved 16 September 2008

[OOOAPIk] Open Office.org, API – service PageStyle, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/style/PageStyle.html, retrieved 17 September 2008

[OOOAPIl] Open Office.org, API – constants group NumberingType, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/style/NumberingType.html, retrieved 17 September 2008

[OOOAPIm] Open Office.org, API – service PageCount, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/textfield/PageCount.html, retrieved 17 September 2008

[OOOAPIn] Open Office.org, API – service PageNumber, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/textfield/PageNumber.html, retrieved 17 September 2008

[OOOAPIo] Open Office.org, API – service DateTime, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/textfield/DateTime.html, retrieved 17 September 2008

[OOOAPIp] Open Office.org, API – service LineShape, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/LineShape.html, retrieved 17 September 2008

[OOOAPIq] Open Office.org, API – struct Point, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/awt/Point.html, retrieved 17 September 2008

[OOOAPIr] Open Office.org, API – struct Size, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/awt/Size.html, retrieved 17 September 2008

[OOOAPIs] Open Office.org, API – interface XMultiComponentFactory, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/lang/XMultiComponentFactory.html, retrieved 17 September 2008

[OOOAPIt] Open Office.org, API – interface XMultiServiceFactory, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/lang/XMultiServiceFactory.html, retrieved 17 September 2008

[OOOAPIu] Open Office.org, API – service ConfigurationProvider, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/configuration/ConfigurationProvider.html, retrieved 17 September 2008

[OOOAPIv] Open Office.org, API - service ConfigurationAccess, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/configuration/ConfigurationAccess.html, retrieved 17 September 2008

[OOOAPIw] Open Office.org, API – interface XNameAccess, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/container/XNameAccess.html, retrieved 17 September 2008

[OOOAPIx] Open Office.org, API – interface XTextCursor, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextCursor.html, retrieved 17 September 2008

[OOOAPIy] Open Office.org, API – interface XPropertySet, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/beans/XPropertySet.html, retrieved 18 September 2008

[OOOAPIz] Open Office.org, API – service ParagraphProperties, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/style/ParagraphProperties.html, retrieved 18 September 2008

[OOOAPIaa] Open Office.org, API – service CharacterProperties, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/style/CharacterProperties.html, retrieved 18 September 2008

[OOOAPIab] Open Office.org, API – enum Break Type, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/style/BreakType.html, retrieved 18 September 2008

[OOOAPIac] Open Office.org, API – constant group FontWeight, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/awt/FontWeight.html, retrieved 20 September 2008

[OOOAPIad] Open Office.org, API – interface XSimpleText, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/XSimpleText.html, retrieved 20 September 2008

[OOOAPIae] Open Office.org, API – service TextTable, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/TextTable.html, retrieved 20 September 2008

[OOOAPIaf] Open Office.org, API – interface XChapterNumberingSupplier, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/XChapterNumberingSupplier.html, retrieved 22 September 2008

[OOOAPIag] Open Office.org, API – interface XTextTable, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextTable.html, retrieved 22 September 2008

[OOOAPIah] Open Office.org, API – constants group FontUnderline, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/awt/FontUnderline.html, retrieved 23 September 2008

[OOOAPIai] Open Office.org, API – interface XTextTableCursor, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/XTextTableCursor.html, retrieved 25 September 2008

[OOOAPIaj] Open Office.org, API – constants group ControlCharacter, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/ControlCharacter.html, retrieved 25 September 2008

[OOOAPIak] Open Office.org, API – service RectangleShape, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/RectangleShape.html, retrieved 25 September 2008

[OOOAPIal] Open Office.org API – interface XShape, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/XShape.html, retrieved 25 September 2008

[OOOAPIam] Open Office.org, API – enum TextContentAnchorType, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/TextContentAnchorType.html, retrieved 25 September 2008

[OOOAPIan] Open Office.org, API – enum TextVerticalAdjust, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/TextVerticalAdjust.html, retrieved 25 September 2008

[OOOAPIao] Open Office.org, API – service FillProperties, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/drawing/FillProperties.html, retrieved 25 September 2008

[OOOAPIap] Open Office.org, API - service URL, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/textfield/URL.html, retrieved 25 September 2008

[OOOAPIaq] Open Office.org, API – interface XModifiable, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/util/XModifiable.html, retrieved 25 September 2008

[OOOAPIar] Open Office.org, API – interface XStorable, 2008,http://api.openoffice.org/docs/common/ref/com/sun/star/frame/XStorable.html, retrieved 25 September 2008

[OOOAPIas] Open Office.org, API – interface XPRintable, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/view/XPrintable.html, retrieved 25 September 2008

[OOOAPIat] Open Office.org, API – interface XDocumentIndex, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/text/XDocumentIndex.html, retrieved 26 September 2008

[OOOAPIau] Open Office.org, API – interface XController, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/frame/XController.html, retrieved 26 Speptember 2008

[OOOAPIav] Open Office.org, API – interface XFrame, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/frame/XFrame.html, retrieved 26 September 2008

[OOOAPIaw] Open Office.org, AP – XWindow, 2008, http://api.openoffice.org/docs/common/ref/com/sun/star/awt/XWindow.html, retrieved 26 September 2008

[OOOCSB] Open Office.org, Code Snippet Base, 2005, http://codesnippets.services.openoffice.org/index.xml, retrieved 26 September 2008

[OOOCSBa] Open Office.org, Code Snippet Base – oo-Snippets: Table of Contents, Matthias Prem, 10 August 2006, http://codesnippets.services.openoffice.org/Writer/Writer.TableOfContents.snip, retrieved 16 September 2008

[OOOCSBb] Open Office.org, Code Snippet Base – oo-Snippets Table of Contents, Nicole Scholz, 2009, http://codesnippets.services.openoffice.org/Writer/Writer.TableOfContent.snip, retrieved 2 February 2009

[OOOCSBc] Open Office.org, Code Snippet Base – oo-Snippets Add Page Number, 2008, http://codesnippets.services.openoffice.org/Writer/Writer.AddPageNumber.snip, retrieved 17 September 2008

[OOOCSBd] Open Office.org, Code Snippet Base – oo-Snippets: Change existing Text, 10 August 2008, http://codesnippets.services.openoffice.org/Writer/Writer.ChangeExistingText.snip, retrieved 18 September 2008

[OOOCSBe] Open Office.org, Code Snippet Base – ooSnippets: Page Break, 9 August 2006, http://codesnippets.services.openoffice.org/Writer/Writer.PageBreak.snip, retrieved 18 September 2008

[OOOCSBf] Open Office.org, Code Snippet Base – ooSnippets: change table column width, Nicole Scholz, 2009, http://codesnippets.services.openoffice.org/Writer/Writer.ChangeTableColumnWidth.snip, retrieved 2 February 2009

[OOOCSBg] Open Office.org, Code Snippet Base – oo Snippets: merge table cells, Nicole Scholz 2009, http://codesnippets.services.openoffice.org/Writer/Writer.MergeTableCells.snip, retrieved 2 February 2009

[OOOCSBh] Open Office.org, Code Snippet Base – ooSnippets: adjust table cell text to the right, Nicole Scholz, 2009, http://codesnippets.services.openoffice.org/Writer/Writer.AdjustTableCellTextToTheRight.snip, retrieved 2 February 2009

[OOOCSBi] Open Office.org, Code Snippet Base – ooSnippets: set hyperlink, Nicole Scholz http://codesnippets.services.openoffice.org/Writer/Writer.Hyperlink.snip, 2009, retrieved 2 February 2009

[OOOCSBj] Open Office.org, Code Snippet Base – ooSnippets: Print Existing Document, http://codesnippets.services.openoffice.org/Writer/Writer.PrintExistingDocument.snip, retrieved 25 September 2008

[OOOCSBk] Open Office.org, Code Snippet Base – ooSnippets: Store actual Document as *.pdf, 10 August 2007, http://codesnippets.services.openoffice.org/Writer/Writer.StoreActualDocumentAs__pdf.snip, retrieved 25 September 2008

[OOOCSBl] Open Office.org, Code Snippet Base – ooSnippets: Merge Table Cells, 30 September 2008, http://codesnippets.services.openoffice.org/Writer/Writer.MergeTableCells.snip, retrieved 5 December 2008

[OOOCSBm] Open Office.org, Code Snippet Base – ooSnippets: add date to footer, 30 September 2008, http://codesnippets.services.openoffice.org/Writer/Writer.AddDateToFooter.snip, retrieved 5 December 2008

[OOOD08] Open Office.org, Open Office.org - Developers Guide , 2008, http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/Text/Text_Documents, retrieved on 15 September 2008

[OOOD08a] Open Office.org, Open Office.org - Developers Guide, 2008, http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/ProUNO/Introduction, retrieved on 15 September 2008

[OOOD08b] Open Office.org, Open Office.org - Developers Guide: Indexes and Index Marks, 2008, http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/Text/Indexes_and_Index_Marks, retrieved 16 September 2008

[OOOD08c] Open Office.org, Open Office.org – Developers Guide Formatting, 2008, http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/Text/Formatting, retrieved 18 September 2008

[OOOD08d] Open Office.org, Open Office.org – Developers Guide Tables, 2008, http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/Text/Tables, retrieved 18 September 2008

[OOOD08e] Open Office.org, Open Office.org – Developers Guide Line Numbering and Outline Numbering, 2008, http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/Text/Line_Numbering_and_Outline_Numbering, retrieved 22 September 2008

[OOOD08f] Open Office.org, Open Office.org – Developers Guide Drawing Shapes, 2008, http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/Text/Drawing_Shapes, retrieved 25 September 2008

[OOOPML] Open Office.org, Project Mailing List, 2008, http://api.openoffice.org/servlets/ProjectMailingListList, retrieved 15 September 2008

[OOOOMLa] Open Office.org, Project Mailing List, 2008, DocumentIndexesDemo, http://api.openoffice.org/servlets/ReadMsg?listName=dev&msgNo=19906, retrieved 07 January 2009

[OOOUNO] Professional UNO, http://api.openoffice.org/docs/DevelopersGuide/ProfUNO/ProfUNO.xhtml, retrieved 2 February2009

[ooRex08] Open Object Rexx – About Open Object Rexx, 2008, http://www.oorexx.org/, retrieved 4 May 2008

[ooRexx08a] Open Object Rexx – Current Releases, 2008, http://www.oorexx.org/download.html, retrieved 23 September 08

[RGF08a] Rony G. Flatscher, refCardOOo – Public Routines uno.getDefiinition, 12 September 2008, http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/refcardOOo.pdf, retrieved 26 September 2008

[RGF08b] Rony G. Flatscher, refCardOOo – Public Routines uno.createProperty, 12 September 2008, http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/refcardOOo.pdf, retrieved 26 September 2008

[RGF08c] Rony G. Flatscher, refCardOOo – Public Routines uno.connect(), 12 September 2008, http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/refcardOOo.pdf, retrieved 26 September 2008

[RGF08d] Rony G. Flatscher, refCardOOo – Public Routines uno.createDesktop, 12 September 2008, http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/refcardOOo.pdf, retrieved 26 September 2008

[RGF08e] Rony G. Flatscher, refCardOOo – Module UNO.CLS (Open Office.org), 12 September 2008, http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/refcardOOo.pdf, retrieved 26 September 2008

[RGF08f] Rony G. Flatscher, refCardOOo – Public Routines uno.getTypeName,12 September 2008, http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/refcardOOo.pdf, retrieved 26 September 2008

[Vimorg] Vim the editor - Downloading Vim, http://www.vim.org/download.php, retrieved 26 September 2008

7 Attachment

```
/*   DocumentIndexesDemo.rex, 2008-10-21, --- rgf, 2008-12-22
   transcribed from Ariels DocumentIndexDemo.java as of 2008-08-31 18:21:29*/

.local~docIdx=.directory~new    -- save a directory object in .local environment, can
                                -- be simply retrieved by its environment symbol (i.e.
                                -- its name with a dot prepended)

.docIdx~content_index1="API Content Index 1"
.docIdx~content_index2="API Content Index 2"

         -- save paragraph_break for later use
.docIdx~paragraph_break=uno.loadClass("com.sun.star.text.ControlCharacter")       -
       ~PARAGRAPH_BREAK

if arg(1,"omitted") then
   .docIdx~m_xContext=uno.connect() -- create a connection, save context
else
   .docIdx~m_xContext=arg(1)        -- use supplied context

call run                            -- invoke the routine "run"

::requires uno.cls                  -- load the UNO support

/****************************************************************************** */
::routine run
  call initDoc
  if .docIdx~m_xTextDocument<>.nil then
  do
      call initData
      call demoOutlineNumbering
      call createIndexes
  end

/* ****************************************************************************** */
::routine initDoc
  xTextDocument = createNewTextDoc(.docIdx~m_xContext)
  if xTextDocument<>.nil then
  do
     .docIdx~m_xTextDocument = xTextDocument
     .docIdx~m_xStyleFamilies = xTextDocument~XStyleFamiliesSupplier~getStyleFamilies
     .docIdx~m_xDummyText = getDummyText(.docIdx~m_xContext)
  end

/* ****************************************************************************** */
::routine initData

  xText = .docIdx~m_xTextDocument~getText
  xTextCursor = xText~createTextCursor
  xTextCursorProps = xTextCursor~XPropertySet

  call insertParaBreak xTextCursor~getEnd
  call insertParaBreak xTextCursor~getEnd

  defaultPropertyValue="Default"
  headingText="Heading Level"
  if .sLang='de' then      -- if German, must use German property value!
  do
     defaultPropertyValue='Standard'
     headingText="Überschrift Ebene"
  end

  do 4  -- repeat block four times
        -- Heading Level 1
      xTextCursorProps~setPropertyValue("ParaStyleName", "Heading 1")
      call insertText xTextCursor, headingText "1"
```

```
      -- dummy text
      call insertParaBreak xTextCursor~getEnd
      xTextCursorProps~setPropertyValue("ParaStyleName", defaultPropertyValue)

      call insertDummyText xTextCursor
      call insertParaBreak xTextCursor~getEnd

      -- Heading Level 2
      xTextCursorProps~setPropertyValue("ParaStyleName", "Heading 2")
      call insertText xTextCursor, headingText "2"

      call insertParaBreak xTextCursor~getEnd
      xTextCursorProps~setPropertyValue("ParaStyleName", defaultPropertyValue)

      call insertDummyText xTextCursor
      call insertParaBreak xTextCursor~getEnd

      -- Heading Level 3
      xTextCursorProps~setPropertyValue("ParaStyleName", "Heading 3")
      call insertText xTextCursor, headingText "3"

      call insertParaBreak xTextCursor~getEnd
      xTextCursorProps~setPropertyValue("ParaStyleName", defaultPropertyValue)

      call insertDummyText xTextCursor
      call insertParaBreak xTextCursor~getEnd
  end

/* ************************************************************************** */
::routine insertDummyText
  use arg xTextRange
  if (.docIdx~m_xDummyText <> .nil) then
      .docIdx~m_xDummyText~applyTo(xTextRange)
  else
      call insertText xTextRange, "Bla, bla, bla."

/* ************************************************************************** */
::routine createIndexes

say "in createIndexes(), before defining TOC, look at the document ..."
call beep 750,100
"pause"

  xDocFactory = .docIdx~m_xTextDocument~XMultiServiceFactory

  oContentIndex = xDocFactory~createInstance("com.sun.star.text.ContentIndex")

  -- a unique name may help for later access
  xNamedIndex = oContentIndex~XNamed

  xNamedIndex~setName(.docIdx~CONTENT_INDEX1)

  -- access the set of properties
  xContentIndexProps = oContentIndex~XPropertySet

  -- "Title" is a property of every BaseIndex
  xContentIndexProps~setPropertyValue("Title", "API Generated ContentIndex")
  -- "IsProtected" is a property of every BaseIndex
  xContentIndexProps~setPropertyValue("IsProtected", box("bool",.true))
  -- "CreateFromOutline" is a ContentIndex property
  xContentIndexProps~setPropertyValue("CreateFromOutline", box("bool",.true))

  -- HOW TO CHANGE THE STYLES?
  ------------------------------------------------------------------/
  --
  --  GETTER METHOD
  --  1. get the name of the style applied to the level we want
  --  2. get that style from the paragraph styles family
  --  3. modify it
```

```
--*****************************************************************************

xParaStyleFamily = .docIdx~m_xStyleFamilies~getByName("ParagraphStyles")        -
                          ~XNameContainer

-- the service BaseIndex has properties to access the styles:
-- ParaStyleHeading
-- ParaStyleLevel1, ParaStyleLevel2 ... ParaStyleLevel10

-- change the heading style
sParaStyleHeading = xContentIndexProps~getPropertyValue("ParaStyleHeading")

if xParaStyleFamily~hasByName(sParaStyleHeading) then
do
    xHeadingStyleProps = xParaStyleFamily~getByName(sParaStyleHeading)~XPropertySet

    center=uno.loadClass("com.sun.star.style.ParagraphAdjust")~center
    xHeadingStyleProps~setPropertyValue("ParaAdjust", center)

    xHeadingStyleProps~setPropertyValue("CharHeight", box("int",20))
    xHeadingStyleProps~setPropertyValue("CharColor", box("int",getRGBColor(0,0,125)))
    xHeadingStyleProps~setPropertyValue("CharFontName", "DejaVu Serif;Galatia SIL");
end

-- change the style for level 1
sParaStyleLevel1 = xContentIndexProps~getPropertyValue("ParaStyleLevel1")

if xParaStyleFamily~hasByName(sParaStyleLevel1) then
do
    xParaStyleLevel1 = xParaStyleFamily~getByName(sParaStyleLevel1)~XPropertySet

    italic=uno.loadClass("com.sun.star.awt.FontSlant")~italic
    xParaStyleLevel1~setPropertyValue("CharPosture", italic)
end

-- change the style for level 3
sParaStyleLevel3 = xContentIndexProps~getPropertyValue("ParaStyleLevel3")

if xParaStyleFamily~hasByName(sParaStyleLevel3) then
do
    xParaStyleLevel3 = xParaStyleFamily~getByName(sParaStyleLevel3)~XPropertySet
    xParaStyleLevel3~setPropertyValue("CharColor", box("int",getRGBColor(255,0,255)))
end

-----------------------------------------------------------------/
--
-- SETTER METHOD
--  1. create a new custom style
--  2. set that style at the index level you want
--
-- create a new style for level 2
sMyParaStyleName = "My Custom ContentIndex Style Level 2"

if xParaStyleFamily~hasByName(sMyParaStyleName) then
    oNewParaStyleLevel2 = xParaStyleFamily~getByName(sMyParaStyleName)
else
do
    oNewParaStyleLevel2 =xDocFactory~createInstance("com.sun.star.style.ParagraphStyle")
    xParaStyleFamily~insertByName(sMyParaStyleName, oNewParaStyleLevel2)
end

xParaStyleLevel2 = oNewParaStyleLevel2~XPropertySet

xParaStyleLevel2~setPropertyValue("ParaLeftMargin", box("int",500))
bold=uno.loadClass("com.sun.star.awt.FontWeight")~bold
xParaStyleLevel2~setPropertyValue("CharWeight", bold)

-- SET IT AT THE INDEX LEVEL
xContentIndexProps~setPropertyValue("ParaStyleLevel2", sMyParaStyleName)
```

```
  ------------------------------------------------------------------

  xDocumentIndex =oContentIndex~XDocumentIndex

  -- insert the index at the start of the document
  call insertTextContent .docIdx~m_xTextDocument~getText~getStart, xDocumentIndex
  xDocumentIndex~update

say "in createIndexes(), *AFTER* defining TOC, look at the document ..."
call beep 750,100
call sysSleep .05
call beep 750,100
"pause"

/* ********************************************************************************** */
::routine demoOutlineNumbering

say "in demoOutlineNumbering, before defining header level formatting,"          -
    "look at the document ..."
call beep 1000,100
"pause"

  xChapterNumberingSupplier = .docIdx~m_xTextDocument~XChapterNumberingSupplier

  if xChapterNumberingSupplier = .nil then
      return

  xChapterNumberingRules = xChapterNumberingSupplier~getChapterNumberingRules

  aChapterNumberingList = uno.createArray(.uno~PropertyValue, 3)

  NumberingType=uno.loadClass("com.sun.star.style.NumberingType")

  ------------------------------------------------------------------
  /*
   * HeadingStyleName
   *Contains the name of the paragraph style
   * that marks a paragraph as a chapter heading.
   */
  aHeadingStyleName = uno.createProperty("HeadingStyleName", "Heading 1");
  aChapterNumberingList[1]=aHeadingStyleName

  /*
   * NumberingType
   * Determines the type of the numbering defined in com.sun.star.style.NumberingType.
   */
  aNumberingType = uno.createProperty("NumberingType", NumberingType~ROMAN_UPPER)
  aChapterNumberingList[2]=aNumberingType

  /*
   * Suffix
   * the suffix of the numbering symbol.
   */
  aSuffix = uno.createProperty("Suffix", ". ")
  aChapterNumberingList[3]=aSuffix

  xChapterNumberingRules~replaceByIndex( 0, aChapterNumberingList )

  ------------------------------------------------------------------
  -- create a new array of PropertyValues
  aChapterNumberingList = uno.createArray(.uno~PropertyValue, 4)

  -- HeadingStyleName
  aHeadingStyleName~Value = "Heading 2";
  aChapterNumberingList[1]=aHeadingStyleName

  -- NumberingType
  aNumberingType~Value = NumberingType~ARABIC;
  aChapterNumberingList[2]=aNumberingType

  -- Suffix
```

```
  aSuffix~Value = " ";
  aChapterNumberingList[3]=aSuffix

  /*
   * ParentNumbering
   * Determines if higher numbering levels are included in the numbering,
   * for example, 2.3.1.2.
   */
  aParentNumbering = uno.createProperty("ParentNumbering", box("short",2))
  aChapterNumberingList[4]=aParentNumbering

  xChapterNumberingRules~replaceByIndex( 1, aChapterNumberingList )

  -- we reuse the array!

  ------------------------------------------------------------------

  -- HeadingStyleName
  aHeadingStyleName~Value = "Heading 3"

  -- NumberingType
  aNumberingType~Value = NumberingType~ROMAN_LOWER

  -- Suffix
  aSuffix~Value = ") "

  -- ParentNumbering
  aParentNumbering~Value = box("short", 3)

  xChapterNumberingRules~replaceByIndex( 2, aChapterNumberingList )

say "in demoOutlineNumbering, *AFTER* defining header level formatting, "        -
    "look at the document ..."
call beep 1000,100
call sysSleep .05
call beep 1000,100
"pause"

/* ************************************************************************** */
::routine createNewTextDoc
  use arg xContext

  xTextDocument = .nil
  oDesktop = xContext~getServiceManager                                          -
             ~createInstanceWithContext("com.sun.star.frame.Desktop", xContext)
  xComponentLoader = oDesktop~XComponentLoader

  if xComponentLoader<>.nil then
  do
     tmp = xComponentLoader                                                      -
           ~loadComponentFromURL("private:factory/swriter","_default", 0, .uno~noProps)
     xTextDocument = tmp~XTextDocument
  end
  return xTextDocument


/* ************************************************************************** */
::routine insertText
  use arg xTextRange, string
  xTextRange~getText~insertString(xTextRange, string, .false)

/* ************************************************************************** */
::routine insertParaBreak
  use arg xTextRange
  xTextRange~getText~insertControlCharacter(xTextRange, .docIdx~paragraph_break, .false)

/* ************************************************************************** */
::routine insertTextContent
  use arg xTextRange, xContent
  xTextRange~getText~insertTextContent(xTextRange, xContent, .false)
```

```
/* ********************************************************************************** */
::routine getDummyText
  use arg xContext

  xAutoTextEntry = .nil

  tmp =xContext~getServiceManager                                                    -
              ~createInstanceWithContext("com.sun.star.text.AutoTextContainer", xContext)
  xAutoTextContainer = tmp~XAutoTextContainer

  if xAutoTextContainer~hasByName("standard") then
  do
      xAutoTextGroup = xAutoTextContainer~getByName("standard")~XAutoTextGroup

      /*
       * Default AutoText's are locale sensitive!
       */
      aOOoLocale = getOOoLocale(xContext)
      .local~sLang = aOOoLocale~Language
      oAutoTextEntry = .nil

      if .sLang="en" then
      do
          if xAutoTextGroup~hasByName("DT") then
              oAutoTextEntry = xAutoTextGroup~getByName("DT")
      end
      else if .sLang="de" then
      do
          if xAutoTextGroup~hasByName("BT") then
              oAutoTextEntry = xAutoTextGroup~getByName("BT")
      end
      -- else if ...

      xAutoTextEntry = oAutoTextEntry~XAutoTextEntry
  end

  return xAutoTextEntry

/* ********************************************************************************** */
::routine  getOOoLocale
  use arg xContext

  aLocale=.bsf~new("com.sun.star.lang.Locale")
  signal on syntax   -- activate signal handler (jump to label "syntax:")

  clz="com.sun.star.configuration.ConfigurationProvider"
  oConfigurationProvider = xContext~getServiceManager                                -
                                  ~createInstanceWithContext(clz, xContext);
  xConfigurationProvider = oConfigurationProvider~XMultiServiceFactory

  props=uno.createArray(.uno~PropertyValue,1)   -- create Java array of type PropertyValue
  props[1]=uno.createProperty("nodepath","/org.openoffice.Setup/L10N")

  clz="com.sun.star.configuration.ConfigurationAccess"
  xNameAccess=xConfigurationProvider~createInstanceWithArguments(clz,props)~XNameAccess

  sLocale = xNameAccess~getByName("ooLocale")

  if sLocale<>"" then            -- locale information present, if so process it
  do
      -- parse the string into two parts according to the explicit template, assign
language
     parse var sLocale aLocale~Language "-" sCountry
     if sCountry<>"" then        -- if country exists, assign it to the new aLocale
        aLocale~Country=sCountry
  end

syntax:
  return aLocale
```

```
/* ********************************************************************************** */
::routine getRGBColor
  use arg r, g, b
  return b+g*256+r*256**2
```

```
/* little Rexx program that calls createApiInfo.rex directly */
call createApiInfo 'com.sun.star.embed.Storage'
call createApiInfo 'com.sun.star.embed.Storage', 2    -- show member definitions as well
```

```
call "createApiInfo.rex"

desktop          = UNO.createDesktop()
xDesktop         = desktop~XDesktop
xComponentLoader = xDesktop~XComponentLoader
writerComponent  = xcomponentLoader~loadComponentFromURL                                -
                   ("private:factory/swriter", "_blank", 0, .UNO~noProps)
xTextDocument    = writerComponent~xTextDocument
xText            = xTextDocument~getText
xTextCursor      = xText~createTextCursor
xTextRange       = xText~getEnd
xTextRange~setString("Hello, this is ooRexx speaking on:" date("S") time())

call analyzeAndCreateReference writerComponent
call analyzeAndCreateReference writerComponent, 2     -- show definition of members as well

::requires uno.cls
```

```
/*license:

 ------------------------ Apache Version 2.0 license -------------------------
       Copyright (C) 2009 Nicole Scholz, Vienna, Austria

       Licensed under the Apache License, Version 2.0 (the "License");
       you may not use this file except in compliance with the License.
       You may obtain a copy of the License at

            http://www.apache.org/licenses/LICENSE-2.0

       Unless required by applicable law or agreed to in writing, software
       distributed under the License is distributed on an "AS IS" BASIS,
       WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
       See the License for the specific language governing permissions and
       limitations under the License.
 -------------------------------------------------------------------------------
*/

/*****************************************************************************/
/*                              frontend                                     */
/*                           of createApiInfo                                */
/*****************************************************************************/

/* we need a swing-Label quite often, hence import it */
/* the ooRexx proxy class is named "swingLabel" */
call bsf.import "javax.swing.JLabel", "swingLabel"

call bsf.import "java.awt.Cursor", "awtCursor"

/*create the frame for the program*/
frame = .bsf~new("javax.swing.JFrame", "UNO API Viewer by Nicole Scholz")

frame~bsf.addEventListener('window', 'windowClosing', 'call BSF "exit"')
frameContentPane=frame~getContentPane

/*layout is set to Gridlayout*/
frameContentPane~setLayout(.bsf~new("java.awt.GridLayout", "12", "2"))

/*create a menubar*/
```

```
menubar= .bsf~new("javax.swing.JMenuBar")
menu= .bsf~new("javax.swing.JMenu", "File")
menubar~add(menu)
menu2= .bsf~new("javax.swing.JMenu", "Info")
menubar~add(menu2)

frame~setJMenuBar(menubar)

/*menu exit*/
menuitem= .bsf~new("javax.swing.JMenuItem", "Exit")
menuitem~bsf.addEventListener('action', '', '.bsf~bsf.exit')

/*menu information*/
menuinfoitem= .bsf~new("javax.swing.JMenuItem", "Information")
menuinfoitem~bsf.addEventListener('action', '',  'call newWindow')
menu~add(menuitem)
menu2~add(menuinfoitem)

/*create textfields and buttons*/
labelBegriff=.swingLabel~new("Enter a fully qualified UNO IDL name:")
labelLeer   =.swingLabel~new("")

tf_UNO_IDL_Name=.bsf~new("javax.swing.JTextField", "", 30)
tf_UNO_IDL_Name~setText("com.sun.star.embed.Storage")

buttonStartProcessing= .bsf~new("javax.swing.JButton", "Press to start processing")
buttonStartProcessing~bsf.addEventListener("action","",
                      'call processUNO_IDL '
                      'frame,'
                      'tf_UNO_IDL_Name,'
                      'cfLayer,'
                      'cfView,'
                      'cfURL,'
                      'tf_local_URL,'
                      'cffirstlevel,'
                      'cfsecondlevel,'
                      'cfthirdlevel,'
                      'cffont')

frameContentPane~~add("1", labelBegriff)
                ~~add("2", labelLeer)
                ~~add("3", tf_UNO_IDL_Name)
                ~~add("4", buttonStartProcessing)

/*font choice */
cffont = .bsf~new("javax.swing.JComboBox")
cffont~~addItem("Arial")
      ~~addItem("Bitstream Vera Sans Mono")
      ~~addItem("Courier New")
      ~~addItem("Cordia New")
      ~~addItem("DejaVueSans Condensed")
      ~~addItem("DejaVueSans Mono")
      ~~addItem("Franklin Gothic Book")
      ~~addItem("Gill Sans MT")
      ~~addItem("Lucida Sans")
      ~~addItem("Old English Text MT")
      ~~addItem("Times new Roman")
      ~~addItem("Verdana")
cffont~setSelectedItem("Verdana")   -- select "Verdana"
label2=.swingLabel~new("Choose the font to use for rendering:")
frameContentPane~~add("9", .swingLabel~new)~~add("10", .swingLabel~new)

  -- get the current numbering types directly from Open Office
numberingtypedef = uno.getDefinition("com.sun.star.style.NumberingType")
numberingtypes = .array~new
i = 1
do while (numberingtypedef <> "")
parse var numberingtypedef  " " deftypename "|" value "|" numberingtypedef
  numberingtypes[i]= deftypename value
  i= i + 1
end
numberingtypes~remove(i-1)
```

```
  -- sort the numbering types
numberingtypes~sort

  -- array of tool tip text description for numbering types from OOo Api
numberingtypesdesc=.array~of(
      "0 Numbering is put in upper case letters as A, B, C, D, ...." ,
      "1 Numbering is in lower case letters as a, b, c, e,...." ,
      "2 Numbering is in Roman numbers with upper case letters as I, II, III, IV,...." ,
      "3 Numbering is in Roman numbers with lower case letters as i, ii, iii, iv, ...." ,
      "4 Numbering is in Arabic numbers as 1, 2, 3, 4, ...." ,
      "5 Numbering is invisible." ,
      "6 Use a character from a specified font." ,
      "7 Numbering is specified in the page style." ,
      "8 Numbering is displayed as a bitmap graphic." ,
      "9 Numbering is put in upper case letters as A, B, ..., Y, Z, AA, BB, CC, ... AAA, ...." ,
      "10 Numbering is put in lower case letters as a, b, ..., y, z, aa, bb, cc, ... aaa, ...." ,
      "11 A transliteration module will be used to produce numbers in chinese, japanese, etc." ,
      "12 Nativenumbersupplier service will be called to produce numbers in native languages." ,
      "13 Numbering for fullwidth Arabic number" ,
      "14 Bullet for Circle Number" ,
      "15 Numbering for Chinese lower case number" ,
      "16 Numbering for Chinese upper case number" ,
      "17 Numbering for Traditional Chinese upper case number" ,
      "18 Bullet for Chinese Tian Gan" ,
      "19 Bullet for Chinese Di Zi" ,
      "20 Numbering for Japanese traditional number" ,
      "21 Bullet for Japanese AIU fullwidth" ,
      "22 Bullet for Japanese AIU halfwidth" ,
      "23 Bullet for Japanese IROHA fullwidth" ,
      "24 Bullet for Japanese IROHA halfwidth" ,
      "25 Numbering for Korean upper case number" ,
      "26 Numbering for Korean hangul number" ,
      "27 Bullet for Korean Hangul Jamo" ,
      "28 Bullet for Korean Hangul Syllable" ,
      "29 Bullet for Korean Hangul Circled Jamo" ,
      "30 Bullet for Korean Hangul Circled Syllable" ,
      "31 Numbering in Arabic alphabet letters" ,
      "32 Numbering in Thai alphabet letters" ,
      "33 Numbering in Hebrew alphabet letters" ,
      "34 Numbering in Nepali alphabet letters" ,
      "35 Numbering in Khmer alphabet letters" ,
      "36 Numbering in Lao alphabet letters" ,
      "37 Numbering in Tibetan/Dzongkha alphabet letters" ,
      "38 Numbering in Cyrillic alphabet upper case letters" ,
      "39 Numbering in Cyrillic alphabet lower case letters" ,
      "40 Numbering in Cyrillic alphabet upper case letters" ,
      "41 Numbering in Cyrillic alphabet lower case letters" ,
      "42 Numbering in Russian Cyrillic alphabet upper case letters" ,
      "43 Numbering in Russian Cyrillic alphabet lower case letters" ,
      "44 Numbering in Russian Cyrillic alphabet upper case letters" ,
      "45 Numbering in Russian Cyrillic alphabet lower case letters" ,
      "46 Numbering in Persian alphabet letters (aa, be, pe, te, ...)")

  --chapter numbering choice first level
cffirstlevel = .bsf~new("javax.swing.JComboBox")
  --chapter numbering choice second level
cfsecondlevel = .bsf~new("javax.swing.JComboBox")
  --chapter numbering choice third level
cfthirdlevel = .bsf~new("javax.swing.JComboBox")

do numberingtypee over numberingtypes~allItems
  cffirstlevel~addItem(numberingtypee)
  cfsecondlevel~addItem(numberingtypee)
  cfthirdlevel~addItem(numberingtypee)
end

cffirstlevel~setSelectedItem("ROMAN_UPPER 2")   -- select "2  ROMAN_UPPER"
label7=.swingLabel~new("Choose numbering type for first chapter level:")

/* set the tool tip text for the first box */
cffirstlevel~setToolTipText(numberingtypesdesc[cffirstlevel~getSelectedItem~word(2)+1])
cffirstlevel~bsf.addEventListener("action","",
      'cffirstlevel~setToolTipText(numberingtypesdesc[cffirstlevel~getSelectedItem~word(2)+1])'
```

```
cfsecondlevel~setSelectedItem("ARABIC 4")   -- select "4  ARABIC"
label8=.swingLabel~new("Choose numbering type for second chapter level:")

/* set the tool tip text for the second box */
cfsecondlevel~setToolTipText(numberingtypesdesc[cfsecondlevel~getSelectedItem~word(2)+1])
cfsecondlevel~bsf.addEventListener("action","",                                    -
      'cfsecondlevel~setToolTipText(numberingtypesdesc[cfsecondlevel~getSelectedItem~word(2)+1])'
)

cfthirdlevel~setSelectedItem("ARABIC 4")   -- select "4  ARABIC"
label9=.swingLabel~new("Choose numbering type for third chapter level:")

/* set the tool tip text for the third box */
cfthirdlevel~setToolTipText(numberingtypesdesc[cfthirdlevel~getSelectedItem~word(2)+1])
cfthirdlevel~bsf.addEventListener("action","",                                     -
      'cfthirdlevel~setToolTipText(numberingtypesdesc[cfthirdlevel~getSelectedItem~word(2)+1])'

/*view choice */
cfView = .bsf~new("javax.swing.JComboBox")
cfView~~addItem("1 View in writer")                                                -
      ~~addItem("2 Save as odt-file and view in writer")                           -
      ~~addItem("3 Save as odt-file and print document")                           -
      ~~addItem("4 Save as odt-file and create a pdf-file")
label3=.swingLabel~new("Choose view:")
frameContentPane~~add("11",label2)~~add("12",cffont)~~add("13", label7)            -
                ~~add("14",cffirstlevel)~~add("15", label8)~~add("16",cfsecondlevel) -
                ~~add("17", label9)~~add("18",cfthirdlevel)~~add("19", label3)     -
                ~add("20",cfView)

/*layer choice*/
cfLayer = .bsf~new("javax.swing.JComboBox")
cfLayer~~addItem("1 Analyze given UNO IDL only (first layer/level)")               -
       ~~addItem("2 Additionally, analyze the UNO IDL members too (first & second layer/level)")
label5=.swingLabel~new("Choose layer/level to analyze:")
frameContentPane~~add("21", label5)~add("22",cfLayer)

/*url choice*/
cfURL = .bsf~new("javax.swing.JComboBox")
cfURL~~addItem("1 Use the OOo Internet API documentation site")                    -
     ~~addItem("2 Use local API without jump marks/fragment identifiers (Windows)") -
     ~~addItem("3 Use local API with jump marks/fragment identifiers (Unix)")
cfURL~bsf.addEventListener('action', '',
'tf_local_URL~setEditable(cfURL~getSelectedItem~left(1)<>1)')

/* local location to hyperlink to, if cfURL<>1 */
label4=.swingLabel~new("Choose the location to hyperlink to (API reference target):")
frameContentPane~~add("23",label4)~add("24",cfURL)

label5=.swingLabel~new("Enter the directory name of the local SDK installation: ")
tf_local_URL=.bsf~new("javax.swing.JTextField", "", 35)
tf_local_URL~setText("OpenOffice.org_2.4_SDK/")
tf_local_URL~setEditable(.false)    -- do not allow to edit, unless "cfURL<>1"
frameContentPane~~add("25", label5)~add("26", tf_local_URL)

/* paint frame in front and set size of the frame*/
frame~~pack~~show~toFront


do forever
        INTERPRET .bsf~bsf.pollEventText
        if result="SHUTDOWN REXX!" then leave
end

::requires "bsf.cls"       -- get the Java support
::requires "createApiInfo.rex"   -- loads also "rgf_util2.rex"
::requires UNO.CLS
  -- routine for creating the information window
::routine newWindow
  .local~frame1 = .bsf~new("javax.swing.JFrame", "About this Program")

  type="text/html"      -- MIME type of text
  txt="<h3>Program Information</h3>                                                "  -
      "                                                                            "  -
```

```
      "This program queries the given, fully qualified UNO IDL string at runtime and   <br>" -
      "formats the results in an OpenOffice.org writer document, such that the user is <br>" -
      "able to learn about all of its documented capabilities. The UNO IDL definitions <br>" -
      "are hyperlinked either to the OpenOffice.org Internet documentation or to the   <br>" -
      "locally installed OpenOffice.org developer kit.                                  " -
      "                                                                                 " -
      "<p>The user can determine:                                                       " -
      "   <ul>                                                                          " -
      "   <li> the fully qualified name of the UNO IDL definition to analyze,           " -
      "   <li> which font should be used for rendering,                                 " -
      "   <li> which numbering types should be used for the chapter levels,             " -
      "   <li> whether the rendered result should only be viewed or whether it should <br> " -
      "        be saved, printed and/or rendered to PDF,                                " -
      "   <li> whether the members of the analyzed UNO IDL definition should be       <br> " -
      "        analyzed themselves (&quot;2 show first and second layer&quot;)           " -
      "   <li> where the generated hyperlinks point to (Internet or local machine).     " -
      "   </ul>                                                                         " -
      "<p>                                                                              " -
      "                                                                                 " -
      "Just experiment a little bit in order to find the settings of your liking.       " -
      "                                                                                 " -
      "<p>Nicole Scholz, Vienna, Austria 2009                                           "
  jEdit=.bsf~new("javax.swing.JEditorPane", "text/html", txt)
  jEdit~setEditable(.false)

  .frame1~getContentPane~add(jEdit)
  .frame1~getContentPane~setLayout(.bsf~new("java.awt.GridLayout", "1", "1"))
  .frame1~bsf.addEventListener('window', 'windowClosing', 'Call closeInfoWindow')
  .frame1~~pack~~show~~toFront
  return

  -- method for closing one frame
::routine closeInfoWindow
  .frame1~~show~~dispose
  return

  -- routine which invokes the processing, but sets the cursor to WAIT cursor and resets it upon
return
::routine processUNO_IDL
  use arg frame, tf_UNO_IDL_Name, cfLayer, cfView, cfURL, tf_local_URL, cffirstlevel,
cfsecondlevel, cfthirdlevel, cfFont

  cursor=frame~getCursor      -- get current cursor
  frame~setCursor(.awtCursor~wait_cursor) -- set cursor

  call analyzeAndCreateReference tf_UNO_IDL_Name~getText,                           -
              cfLayer~getSelectedItem~word(1),                                      -
              cfView~getSelectedItem~word(1),                                       -
              cfURL~getSelectedItem~word(1),                                        -
              tf_local_URL~getText,                                                 -
              cffirstlevel~getSelectedItem~word(2),                                 -
              cfsecondlevel~getSelectedItem~word(2),                                -
              cfthirdlevel~getSelectedItem~word(2),                                 -
              cffont~getSelectedItem

  frame~setCursor(cursor)     -- re-set cursor
```

```
/*license:
 ------------------------ Apache Version 2.0 license ------------------------
       Copyright (C) 2009 Nicole Scholz, Vienna, Austria

       Licensed under the Apache License, Version 2.0 (the "License");
       you may not use this file except in compliance with the License.
       You may obtain a copy of the License at

            http://www.apache.org/licenses/LICENSE-2.0

       Unless required by applicable law or agreed to in writing, software
       distributed under the License is distributed on an "AS IS" BASIS,
       WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
       See the License for the specific language governing permissions and
       limitations under the License.
 ----------------------------------------------------------------------------*/

/*****************************************************************************/
/*                              createApiInfo                                */
/*                                API-Viewer                                 */
/*****************************************************************************/

.local~bDebug        =.false
.local~linebreak     = "0a"x
.local~rectangleposy = 0
.local~rectangleposx = 100
.local~oldwidth      = 0
.local~yHeight       = 0
.local~yreduction    = 800
.local~nomembers     = .false
.local~tab1          = "09"x
.local~selectedUrl   = 0
.local~layernumber   = 1

  -- import those Java classes that we need multiple times
call bsf.import "com.sun.star.awt.FontWeight", "awtFontWeight"
call bsf.import "com.sun.star.awt.Size",       "awtSize"
call bsf.import "com.sun.star.awt.Point",      "awtPoint"

 -- save Java true and Java false object in .local
.local~java.true  = box("Boolean", .true)      -- Java Boolean object representing .true
.local~java.false = box("Boolean", .false)     -- Java Boolean object representing .false

call bsf.import "com.sun.star.text.ControlCharacter", "ctlChar"
call bsf.import "com.sun.star.text.TextContentAnchorType", "anchorType"
call bsf.import "com.sun.star.style.BreakType", "pageBreakType"
call bsf.import "com.sun.star.awt.FontSlant", "fontSlant"
call bsf.import "com.sun.star.style.ParagraphAdjust", "paragraphAdjust"
call bsf.import "com.sun.star.style.NumberingType", "numTypes"

 -- define a directory to store all kind of useful data for this application, i.e.
 -- (standard) values that we need over and over
.local~scholz=.directory~new
.scholz~font   ="Verdana"        -- default Font

 -- font-weights that we need over and over
.scholz~bold   = box("float", .awtFontWeight~bold)
.scholz~normal = box("float", .awtFontWeight~normal)

 -- pre-defined font sizes boxed as Java Float objects, sizes "6" through "14"
do i=6 to 14
  .scholz[i]=box("float",i)
end

.scholz~footerCharHeight =.scholz[8]   -- font size in footer
.scholz~defaultCharHeight=.scholz[10]  -- default font size
.scholz~backColor = box("int", "A4A4A4"x ~c2d)  -- grey
.scholz~charColor = box("int", 0)               -- black
.scholz~fillColor = box("int", "CECEF6"x ~c2d)  -- malve/violette

 -- array of UNO types, make sure it is sorted
.scholz~uno.types=.array~of("UNO_ATTRIBUTE", "UNO_CONSTANTS", "UNO_ENUM", "UNO_EXCEPTION",-
                            "UNO_INTERFACE", "UNO_METHOD", "UNO_MODULE", "UNO_PROPERTY",  -
                            "UNO_SERVICE", "UNO_SINGLETON", "UNO_STRUCT", "UNO_TYPEDEF")  -
                            ~ sort

 -- get and set the version of Open Office: saving into .scholz-directory
xScriptContext=uno.getScriptContext()  -- if dispatched as a macro by OOo, this will not be .nil
if xScriptContext=.nil then
   call query_OOo_Installation
else                                   -- use the component context from the dispatcher
   call query_OOo_Installation xScriptContext~getComponentContext


if arg()=0 then exit    -- no arguments supplied hence return
```

```
 -- ok invoked with arguments already, hence invoke "analyzeAndCreateReference()"
use strict arg uno_object=.nil, cfLayer=1, cfView=1, cfURL=1, localLink="",                -
               cffirstlevel = 3, cfsecondlevel = 5, cfthirdlevel=5, cfFont="Arial"

call analyzeAndCreateReference uno_object, cfLayer, cfView, cfURL, localLink,             -
               cffirstlevel,cfsecondlevel, cfthirdlevel, cfFont

::requires oooReflectionUtil.cls -- get OOo-reflection utilities (which loads UNO.CLS + BSF.CLS)


/* Routine that queries and renders the UNO type:
   queryIDL..UNO IDL string (may contain '+', if multiple services are implemented)
   cfLayer ..1 (show first /layer), 2 (also show 2nd layer)
   cfView  ..View result: 1 (writer), 2 (save as odt + view in writer), 3 (save as odt + print),
                          4 (save as odt + pdf)
   cfURL   ..UNO HTML docs: 1 (Internet), 2 (local without jump marks), 3 (local with jump marks)
                            local: UDK needs to be installed, marks work currently on Unix only
   locallink..local directory where the OOo UDK is installed (e.g. "OpenOffice.org_2.3_SDK/")
   cfFont  ..fontname to use (e.g. "Verdana")*/
::routine analyzeAndCreateReference public
  use strict arg queryIDL=.nil, cfLayer=1, cfView=1, cfURL=1, localLink="",                -
                 cffirstlevel = 3, cfsecondlevel = 5, cfthirdlevel = 5, cfFont="Arial"

   unoObject=.nil       -- if an UNO object was passed, it is stored with this variable
   title=""             -- title to be used in footer
   tmpAllMembersRelation=.relation~new -- relation to store members , if cfLayer=2
   arrQueryIDL=.nil
   if \arg(1)~isA(.string) then  -- first argument not a string
   do
      unoObject=arg(1)           -- save object
      defUnoObject=unoObject~uno.getDefinition     -- get Definition
        -- if a UNO_SERVICE object, then it may implement multiple UNO services!
      if defUnoObject~match(1,"UNO_SERVICE|") then
      do
           -- get the IDL string (possibly multiples, concatenated by '+'
         parse var defUnoObject before "|" queryIDL "|" after

         arrQueryIDL=queryIDL~makeArray("+")       -- split string at "+" chars

         if arrQueryIDL~items>1 then -- multiple services implemented in UNO service object ?
         do
             tmpStr=arrQueryIDL[1]"+..."
             defUnoObject=before"|"tmpStr"|"after
             title="UNO_SERVICE" tmpStr "(multiple services implemented in UNO service object)"

               -- sort array caselessly by unqualified name
             arr=.array~new
             do s over arrQueryIDL
                arr~append( (s~substr(s~lastpos(.)+1)) s)
             end
             arr=arr~sortWith(.caselessComparator~new)
             do i=1 to arr~items
                parse value arr[i] with . val
                arrQueryIDL[i]=val
             end
         end
         else
         do
              -- make sure we show service in footnote's title hint as well
            title="UNO_SERVICE" pp(queryIDL)
         end
      end
   end

   if arrQueryIDL=.nil then                  -- not yet an array
      arrQueryIDL=queryIDL~makeArray("+")    -- split string at "+" chars

   say "processing "pp(queryIDL)", consisting of" arrQueryIDL~items "part(s) ..."

   .local~selectedUrl = cfURL
   if .selectedUrl = 1 then
       -- link into the OOo's Internet base URL for IDL reference documentation
      .scholz~ooo.baseUrl="http://api.openoffice.org/docs/common/ref/"
   else
   do
        -- link into the OOo's local directory base URL for IDL reference documentation
      a=.java.lang.system~getProperty("os.name")
      if a~left(1)="W" then
      do
         cp=value("ProgramFiles", , "ENVIRONMENT")
         cplink = cp~changestr("\","/")
         .scholz~ooo.baseUrl=cplink"/"locallink||"docs/common/ref/"
      end
```

```
      else
         .scholz~ooo.baseUrl=locallink||"docs/common/ref/"
   end

   if queryIDL="" | abbrev(queryIDL,"com.")=.false then
   do
      .bsf.dialog~messageBox("Please insert a valid UNO IDL string"                    -
                             .endOfLine'e.g.- "com.sun.star"',                         -
                             "Invalid UNO Typedefinition String Entered!","error")
      return .false
   end

   if cfFont<>"" then              -- font given, if so save it with the .scholz-directory
   do
      if cfFont="Times new Roman" then
         cfFont = "Times"

      .scholz~font=cfFont~strip    -- save Font name
   end

   if .bDebug=.true then
   do
      say "view     --->" pp(cfView)
      say "font     --->" pp(cfFont)
      say "queryIDL --->" pp(queryIDL)
   end

   def=uno.getDefinition(arrQueryIDL[1]) -- try to get a definition of this UNO IDL string
   eol2=.endOfLine~copies(2)
   if def="" then        -- nothing returned, maybe RgfReflectUNO's xContext out of scope?
   do
      call UNO.resetRgfReflectUNO
      def=uno.getDefinition(arrQueryIDL[1]) -- try again
      if def="" then                        -- nope, we got troubles!
      do
         .bsf.dialog~messageBox("IDL-Name:" pp(queryIDL) eol2                          -
                                "Please restart application!",                         -
                                "No UNO-Definition returned!", "error")
         return .false
      end
   end
   else
     do
      if .bDebug=.true then
      do
         max=350         -- maximum number of chars from definition to display in the popup
         if length(def)>max then tmpDef=def~left(max)"...<cut>...".endOfLine"..."
                            else tmpDef=def
         .bsf.dialog~messageBox("IDL-Name:" pp(queryIDL) eol2 "IDL-Definition:"         -
                                ppd(tmpDef), "IDL-Definition", "i")
      end
   end

/* ********************************************************************************** */
-- ----------------------> create document, TOC and footer definitions
     -- start the Writer of Open Office
   oDesktop         = UNO.createDesktop()   -- get the UNO Desktop service object
     -- get componentLoader interface
   xComponentLoader = oDesktop~XDesktop~XComponentLoader

   /* open the blank *.odt - file */
   url = "private:factory/swriter"
   /* set document properties: documenttitle and window hidden */
   props = bsf.createArray(.UNO~PropertyValue, 1)
   props[1] = .UNO~PropertyValue~new

     -- set document title
   props[1]~Name = "DocumentTitle"
   props[1]~Value = "IDL-Definitions-"queryIDL
     -- set document hidden that the creation time is shorter (don't show Window)
   props[1]~Name = "Hidden"
   props[1]~Value = .java.true -- box("boolean", .true)

   xWriterComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0, props)
   xWriterDocument  = xWriterComponent~XTextDocument

      -- do not update screen
   if \xWriterDocument~hasControllersLocked then xWriterDocument~lockControllers

   /* get the MulitServiceFactory interface from the current document */
   xDMsf = xWriterDocument~XMultiServiceFactory
   /* create the TextObject and the TextCursor */
   xText = xWriterDocument~getText
```

```
     -- create TOC
   contentTextContent = create_TOC( xDMsf, xText, xWriterDocument, cfLayer, arrQueryIDL,  -
                                    cffirstlevel, cfsecondlevel, cfthirdlevel)

     -- create footer
   if title="" then     -- no title yet, create it
   do
      refObj=createReflectionObject(def)  -- create reflection object from definition
      title= refObj~unoTypeName pp(refObj~fullyQualifiedName)
   end
   call create_footer xDMSf, xWriterDocument, title

   if (cfLayer = 2) then
      sectiontitle = "SECTION A: DEFINITION"
   else
      sectiontitle = ""

   /* disable underlining of hyperlinks in the whole document */
   xFamiliesSupplier = xWriterDocument~XStyleFamiliesSupplier
   xCharStyleFamily = xFamiliesSupplier~getStyleFamilies                                -
                                      ~getByName("CharacterStyles")~XNameContainer
   xInternetLinkStyleProps= xCharStyleFamily~getByName("Internet link")
   xVisitedInternetLinkStyleProps=xCharStyleFamily~getByName("Visited Internet Link")
   xInternetLinkStyleProps~xPropertySet~setPropertyValue("CharUnderline",               -
        box("short", bsf.getConstant("com.sun.star.awt.FontUnderline", "NONE")))
   xVisitedInternetLinkStyleProps~xPropertySet~setPropertyValue("CharUnderline",        -
        box("short", bsf.getConstant("com.sun.star.awt.FontUnderline", "NONE")))

/* ********************************************************************************** */
   if arrQueryIDL~items>1 then   -- multiple service UNO IDLs in service object found ?
   do
      say "First processing aggregate service definitions" "..."
      refObj=createReflectionObject(defUnoObject) --create reflection object from definition
      sectiontitle = "SECTION A: SERVICE OBJECT " refObj~fullyQualifiedName
      lfTab="0a09"x
      hintText="The UNO service object implements multiple services, namely:" "0a"x
      do service over arrQueryIDL   -- iterate over service
         hintText=hintText || lfTab || service
      end

      call createSections xWriterDocument, xDMsf, xText, refObj, hintText, cfLayer,      -
      sectiontitle, arrQueryIDL
      sectiontitle = ""
      call insertDelimiterPage xText, "Individual Service Definitions ...", 24
   end

/* ********************************************************************************** */
   do i=1 to arrQueryIDL~items
      def=uno.getDefinition(arrQueryIDL[i])      -- get UNO IDL string
      say "... processing part #" i":" pp(arrQueryIDL[i]) "..."

      refObj=createReflectionObject(def)  -- create reflection object from definition

      if .bDebug=.true then
      do
         say "--> fullyQualifiedName:" pp(refObj~fullyQualifiedName)
         say "           unoTypeName:" pp(refObj~unoTypeName)
      end

      hintText=""
      if refObj~isA(.uno_definitionWithMembers), refObj~definitions~items=0 then
         hintText="No members defined."
      if arrQueryIDL~items>1 then
      do
        if i = 1 then
          sectiontitle = "SECTION B: CONTAINED SERVICE OBJECTS"
        else
          sectiontitle = ""
      end

      call createSections xWriterDocument, xDMsf, xText, refObj, hintText, cfLayer,      -
                          sectiontitle, arrQueryIDL

         -- if members should be shown and a container type in hand, then save the members
      if cfLayer=2, refObj~isA(.uno_definitionWithMembers) then
         tmpAllMembersRelation~putAll(refObj~definitions)
   end

/* ********************************************************************************** */
   if cfLayer = 2 then      -- show member definitions too
   do
      say
      say "Now processing all the member definitions:"
      say
```

```
      if arrQueryIDL~items>1 then
        sectiontitle = "SECTION C: LAYER 2"
      else
        sectiontitle = "SECTION B: LAYER 2"
      call insertDelimiterPage xText, "Level/Layer 2 Definitions (Member Definitions) ...", 24

      allMembers=tmpAllMembersRelation~allItems~sort
      setProcessed=.set~new
      do member over allMembers
         tmpFQName=member~fullyQualifiedName
         if uno.getTypeName(tmpFQName)<>"", setProcessed~hasIndex(tmpFQName)=.false then
         do
            setProcessed~put(tmpFQName)  -- remember that we have processed this definition

            def=uno.getDefinition(member~fullyQualifiedName)
            refObj=createReflectionObject(def)  -- create reflection object from definition

            say "09"x"--->" pp(editName(refObj~fullyQualifiedName))" ..."

            hintText=""
            if refObj~isA(.uno_definitionWithMembers), refObj~definitions~items=0 then
               hintText="No members defined."
            call createSections xWriterDocument, xDMsf, xText, refObj, hintText, cfLayer, -
                                sectiontitle, arrQueryIDL
            sectiontitle = ""
         end
      end
   end

     -- TOC UPDATEN
   contentTextContent~XDocumentIndex~update()

     -- set WriterDocument visible after creation
   xWriterDocument~XModel~getCurrentController()~XController~getFrame()~XFrame            -
                  ~getContainerWindow()~XWindow~setVisible(.true)
     -- now update screen again
   if xWriterDocument~hasControllersLocked then xWriterDocument~unLockControllers
----------------------------------------------------------
   url=makeURL("IDL-Definitions-"queryIDL".odt")

   if cfView=1 then
   do
        -- don't ask for saving pending changes
      xWriterComponent~XModifiable~setModified(.false)
   end
     -- save the created document as odt-file in the current folder
   else if cfView=2 | cfView=3 | cfView=4 then
   do
        -- save created document as odt-file
      xWriterComponent~XStorable~storeAsURL(url, .UNO~noProps)

      if cfView=3 then        -- print the created document to the default printer
      do
         xPrintable = xWriterComponent~XPrintable
         xPrintable~print(.UNO~noProps) -- print the file to default printer
      end
        -- create a pdf-file from the created odt file
      else if cfView=4 then   -- create a pdf-file from the created odt file
      do
         xStorable = xWriterComponent~XStorable -- get xStorable interface

         -- convert the document to a pdf-file
         props = bsf.createArray(.UNO~propertyValue, 2)
         props[1] = .UNO~PropertyValue~new
         props[1]~Name = "FilterName"
         props[1]~Value = "writer_pdf_Export"
         props[2] = .UNO~PropertyValue~new
         props[2]~Name = "CompressMode"
         props[2]~Value = 2
         storeURL = substr(url, 1, lastpos(".", url)) || "pdf" -- create output file name
         xStorable~storeToUrl(storeURL, props) -- store the file with props to URL
      end
   end

   say "-"~copies(78)

   return

/* Routine that takes an UNO_WithDefinitionsMember reference and returns
   a new relation object where each index represents an UNO-type and associates
   all according members with it. */
::routine makeUNOTypeRel
  use strict arg objRef
```

```
  tmpRel=objRef~definitions          -- get the relation with the members
  newRel=.relation~new               -- create a new relation
  s=tmpRel~supplier
  do while s~available
     newRel[s~item~unoTypeName]=s~item -- associate item with index containing the UNO type name
     s~next                          -- get next index/item pair from the supplier
  end
  return newRel                      -- return new relation

/* Create the subsection "Overview", "Listing by UNO Types", "Graphical Overview" */
::routine createSections
  use strict arg xWriterDocument, xDMsf, xText, refObj, hintText="", cfLayer,            -
                 sectiontitle, arrQueryIDL

   xTextCursor = xText~createTextCursor   -- create a text cursor
   xTextCursor~gotoEnd(.false)
   paraBreak=.CtlChar~paragraph_break     -- gets used multiple times in this routine

   if refObj~isA(.uno_definitionWithMembers) then
    rows=refObj~definitions~allItems~items+1
   else
     rows = 2

     -- create main section heading
   call createMainSectionHeading xDMsf, xText, refObj, cfLayer, sectiontitle, arrQueryIDL
   if hintText<>"" then
   do
      /* create the TextObject and the TextCursor */
      props = xTextCursor~xPropertySet    -- get the properties for this xTextCursor
      props~setPropertyValue("CharFontName", .scholz~font)
      props~setPropertyValue("CharHeight",   .scholz~defaultCharHeight)
      props~setPropertyValue("CharWeight",   .scholz~normal)

      xText~insertControlCharacter(xText~getEnd, paraBreak, .false)
      xText~insertString(xTextCursor, hintText, .false)
      xText~insertControlCharacter(xText~getEnd, paraBreak, .false)
   end

     -- create overview table
   call createTableOverview xDMsf, xText, refObj, cfLayer, arrQueryIDL
   xText~insertControlCharacter(xText~getEnd, paraBreak, .false)

      -- create listing by unoTypes
   call createTableByTypes  xDMsf, xText, refObj, cfLayer, arrQueryIDL
   xText~insertControlCharacter(xText~getEnd, paraBreak, .false)

   if rows > 3 then
     xTextCursor~xPropertySet~setPropertyValue("BreakType",.pageBreakType~"PAGE_BEFORE")
   else
     xText~insertControlCharacter(xText~getEnd, paraBreak, .false)

     -- create overview graphic
   if .nomembers=.true then
     .local~rectangleposy = 2000

   call createGraphicOverview  xWriterDocument, xDMsf, xText, refObj, cfLayer, arrQueryIDL
   xText~insertControlCharacter(xText~getEnd, paraBreak, .false)

/* routine for getting the current Open Office.org installation */
::routine query_OOo_Installation
   use strict arg xContext=.nil

     -- get the version of Open Office
   if xContext =.nil then
      xContext = UNO.connect()         -- connect to server and retrieve the XContext object
   XMcf = xContext~getServiceManager   -- retrieve XMultiComponentFactory

     -- get service object
   service  ="com.sun.star.configuration.ConfigurationProvider"
   sProvider=XMcf~createInstanceWithContext(service, xContext)
   xProvider=sProvider~XMultiServiceFactory  -- get its service manager

     -- create property array to query version
   propArr   =bsf.createArray(.uno~propertyValue, 1)
   prop      =.uno~propertyValue~new
   prop~name ="nodepath"
   prop~value="/org.openoffice.Setup/Product"
   propArr[1]=prop

     -- get a read-only ConfigurationAccess from the sProvider's service manager
   service  ="com.sun.star.configuration.ConfigurationAccess"
   aSettings=xProvider~createInstanceWithArguments(service, propArr)
   xSettings=aSettings~xNameAccess     -- get interface to access by name
```

```
      -- save name and version of product in .local
   .scholz~ooName        =xSettings~getByName("ooName")
   .scholz~ooSetupVersion=xSettings~getByName("ooSetupVersion")

/* Create Table of Content. */
::routine create_TOC
  use strict arg xDMsf, xText, xWriterDocument, cfLayer, arrQueryIDL, cffirstlevel,       -
                 cfsecondlevel, cfthirdlevel

   short3=box("short", 3)     -- Java Short object representing the value "3"
     -- TOC
   contentInd   = xDMsf~createInstance("com.sun.star.text.ContentIndex")
   contentProps = contentInd~XPropertySet

   xFamiliesSupplier = xWriterDocument~XStyleFamiliesSupplier
   xParaStyleFamily  = xFamiliesSupplier~getStyleFamilies                              -
                                        ~getByName("ParagraphStyles")~XNameContainer

   call ParaStyle "ParaStyleHeading", contentProps,xParaStyleFamily, 14
   call ParaStyle "ParaStyleLevel1", contentProps,xParaStyleFamily, 12
   call ParaStyle "ParaStyleLevel2", contentProps,xParaStyleFamily, 10
   call ParaStyle "ParaStyleLevel3", contentProps,xParaStyleFamily, 10

   contentProps~setPropertyValue("CreateFromOutline", .java.true)
   contentProps~setPropertyValue("IsProtected",       .java.true)
   contentProps~setPropertyValue("Level",             short3)
   contentProps~setPropertyValue("Title",             "Table Of Content")

     -- TEXT CONTENT auf TOC setzen und formatieren
   contentTextContent = contentInd~xTextContent()
     -- insert TOC
   xText~insertTextContent(xText, contentTextContent, .false )
   contentIndProps    = contentTextContent~XPropertySet

   /* create two-dimensional array of type PropertyValue */
   propsToc = bsf.createArray(.UNO~PropertyValue, 6, 3)

   /* now set the property values */
     -- show chapter numbering
   propsToc[1,1]=uno.createProperty("TokenType",           "TokenEntryNumber")
   propsToc[1,2]=uno.createProperty("CharacterStyleName",  "")
     -- show hyperlink
   propsToc[2,1]=uno.createProperty("TokenType",           "TokenHyperlinkStart")
   propsToc[2,2]=uno.createProperty("CharacterStyleName",  "")

   propsToc[3,1]=uno.createProperty("TokenType",           "TokenEntryText")
   propsToc[3,2]=uno.createProperty("CharacterStyleName",  "")

   propsToc[4,1]=uno.createProperty("TokenType",           "TokenHyperlinkEnd")
   propsToc[4,2]=uno.createProperty("CharacterStyleName",  "")

   propsToc[5,1]=uno.createProperty("TokenType",           "TokenTabStop")
   propsToc[5,2]=uno.createProperty("TabStopRightAligned", .java.true)
   propsToc[5,3]=uno.createProperty("TabStopFillCharacter",".")
     -- show page number
   propsToc[6,1]=uno.createProperty("TokenType",           "TokenPageNumber")
   propsToc[6,2]=uno.createProperty("CharacterStyleName",  "")

   contentIndProps~setPropertyValue("Level",               short3)
   LevelFormat = contentProps~getPropertyValue("LevelFormat")
   LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(1, propsToc)
   LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(2, propsToc)
   LevelFormat~xIndexAccess~xIndexReplace~replaceByIndex(3, propsToc)

   xChapterNumberingSupplier = xWriterDocument~XChapterNumberingSupplier
   xChapterNumberingRules    = xChapterNumberingSupplier~getChapterNumberingRules()
   aChapterNumberingArray=bsf.createArray(.uno~propertyValue,3)

   numberingType = cffirstlevel
   index = 0
   call setNumberingType aChapterNumberingArray,xChapterNumberingRules,               -
                         "Heading 1", numberingType, index

   numberingType = cfsecondlevel
   index = 1
   call setNumberingType aChapterNumberingArray, xChapterNumberingRules,              -
                         "Heading 2", numberingType, index
   index = 2
   numberingType = cfthirdlevel
   call setNumberingType aChapterNumberingArray, xChapterNumberingRules,              -
                         "Heading 3", numberingType, index
  return contentTextContent

/* routine for setting the numbering type for each heading style */
```

```
::routine setNumberingType
  use arg aChapterNumberingArray, xChapterNumberingRules, heading, numberingType, index
   aChapterNumberingArray[1]=uno.createProperty("HeadingStyleName", heading)
   aChapterNumberingArray[2]=uno.createProperty("NumberingType",box("short",numberingType))
   aChapterNumberingArray[3]=uno.createProperty("Suffix",           ". ")
   xChapterNumberingRules~xIndexAccess~xIndexReplace                                    -
                         ~replaceByIndex(index,aChapterNumberingArray)

/* routine for setting the paragraph style */
::routine ParaStyle
  use arg HeadingStyle, contentProps, xParaStyleFamily, paraCharHeight
   sParaStyleHeading  = contentProps~getPropertyValue(HeadingStyle)
   xHeadingStyleProps = xParaStyleFamily~getByName(sParaStyleHeading)~xPropertySet
   xHeadingStyleProps~setPropertyValue("CharFontName", .scholz~font)
   xHeadingStyleProps~setPropertyValue("CharHeight", box("integer", paraCharHeight))

/* Create the footer. */
::routine create_footer
  use strict arg xDMSf, xWriterDocument, title

     -- create the footer
   xPageStyle   = xDMSf~createInstance("com.sun.star.style.PageStyle")
   xFamiliesSupplier = xWriterDocument~XStyleFamiliesSupplier
   xStyle       = xFamiliesSupplier~getStyleFamilies~getByName("PageStyles")~XNameContainer
   xFooter      = xStyle~getByName("Standard")
   FooterProperty = xFooter~xPropertySet
   FooterProperty~setPropertyValue("FooterIsOn", .java.true)
   footerText   = FooterProperty~getPropertyValue("FooterText")~xText

     -- Creating the total number of pages
   numberingType=box("Short", bsf.getConstant("com.sun.star.style.NumberingType", "ARABIC"))

   pageCount    = xDMSf~createInstance("com.sun.star.text.TextField.PageCount")
   pageCountTC  = pageCount~xTextContent()
   pageCountPS  = pageCount~XPropertySet()
   pageCountPS~setPropertyValue("NumberingType", numberingType)

     -- Creating the actual page number
   pageNumber   = xDMSf~createInstance("com.sun.star.text.TextField.PageNumber")
   pageNumberTC = pageNumber~XTextContent()
   pageNumberPS = pageNumber~XPropertySet()
   pageNumberPS~setPropertyValue("NumberingType", numberingType)
   pageNumberPS~setPropertyValue("SubType", box("Short", "1"))

     -- Create the current date
   datetime     = xDMSf~createInstance("com.sun.star.text.TextField.DateTime")
   datetimeProps= datetime ~XPropertySet()
   datetimeProps~setPropertyValue("IsDate", .java.true)
   datetimeProps~setPropertyValue("IsFixed", .java.true)
   datetimeTC   = datetime~XTextContent()

     -- Create the current time
   datetime1    = xDMSf~createInstance("com.sun.star.text.TextField.DateTime")
   datetimeProps1 = datetime1 ~XPropertySet()
   datetimeProps1~setPropertyValue("IsDate", .java.false)
   datetimeProps1~setPropertyValue("IsFixed", .java.true)
   datetimeTC1  = datetime1~XTextContent()

     -- insert the footer text into the document
   xTextCursorFooter = footerText~createTextCursor
   cur = xTextCursorFooter~xPropertySet
   cur~setPropertyValue("CharHeight", .scholz~footerCharHeight)
   cur~setPropertyValue("CharFontName", .scholz~font)  -- set the font to the choosen font

     -- insert a line before the footer text
   Line = xDMSf~createInstance("com.sun.star.drawing.LineShape")~xShape
   Line~setPosition(.bsf~new("com.sun.star.awt.Point", 1, 1))
   Line~setSize(.bsf~new("com.sun.star.awt.Size", 17000, 0))
   xTextContentShape = Line~xTextContent

     -- insert the page number, the date and the time into the footer
   footerText~insertTextContent(footerText~getEnd, xTextContentShape, .false)
   footerText~getEnd~setString(title .linebreak)
   footerText~getEnd~setString("date: ")
   footerText~insertTextContent(footerText~getEnd, datetimeTC, .false)
   footerText~getEnd~setString(" time: ")
   footerText~insertTextContent(footerText~getEnd, datetimeTC1, .false)
   footerText~getEnd~setString(.tab1 || "version:" .scholz~ooName .scholz~ooSetupVersion)
   footerText~getEnd~setString(.tab1)
   footerText~getEnd~setString("page ")
   footerText~insertTextContent(footerText~getEnd, pageNumberTC, .false)
   footerText~getEnd~setString(" of ")
   footerText~insertTextContent(footerText~getEnd, pageCountTC, .false)
  return
```

```
::routine insertDelimiterPage
  use Arg xText, hint="Level/Layer 2 Definitions (Member Definitions) ...", charHeight=24

   /* create the TextObject and the TextCursor */
   xTextCursor = xText~createTextCursor    -- create a text cursor
   props = xTextCursor~xPropertySet        -- get the properties for this xTextCursor
   xTextCursor~gotoEnd(.false)             -- position at end of text
     -- add paragraph and page break
   xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)
   props~setPropertyValue("BreakType",.pageBreakType~"PAGE_BEFORE")

::routine insertTitleText
  use Arg xText, title,charHeight=14,unoRefObject=.nil,place,heading,cfLayer,serviceobject = .nil

   /* create the TextObject and the TextCursor */
   xTextCursor = xText~createTextCursor    -- create a text cursor
   props       = xTextCursor~xPropertySet -- get the properties for this xTextCursor
   xTextCursor~gotoEnd(.false)             -- position at end of text
   props~setPropertyValue("ParaStyleName",heading)  -- change heading style of a text
   if cfLayer = 2 | serviceobject = 1 then
   do
        -- set different background color for the section heading
      props~setPropertyValue("ParaBackColor", .scholz~fillcolor)
   end
   else
      props~setPropertyValue("ParaBackColor", box("int", "ffffff"x ~c2d))
   props~setPropertyValue("CharFontName", .scholz~font)
   props~setPropertyValue("CharHeight",   .scholz~entry(charHeight))
   props~setPropertyValue("CharWeight",   .scholz~bold)
   if (place = "first") then
   do
      props~setPropertyValue("BreakType",.pageBreakType~"PAGE_BEFORE")
   end

   if cfLayer = 2 | serviceobject = 1 then
   do
      xText~insertString(xTextCursor, title, .false)       -- write bracket
   end
   else
   do
      /* set the headline */
      parse var title "[" IDLname "]" rest
      xText~insertString(xTextCursor, "[", .false)         -- write bracket
     call setUrlWithText xText, idlName, unoRefObject      -- write IDL name with URL
      xText~insertString(xTextCursor, "]" rest, .false)    -- write bracket
   end
      -- add paragraph break
   xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)

/* create main section heading */
::routine createMainSectionHeading
    -- create overview table
  use Arg xDMsf, xText, refObj, cfLayer, sectiontitle, arrQueryIDL

   xTextCursor = xText~createTextCursor()
     -- if there is a second layer choosen then add the queried section to the headlines
   if cfLayer = 2 then
   do
      if sectiontitle <> "" then
      do
        call insertTitleText xText, sectiontitle, , , "first", "Heading 1", cfLayer
        call insertTitleText xText, pp(editName(refObj~fullyQualifiedName)), , refObj,,   -
                             "Heading 2"
      end
      else
        call insertTitleText xText, pp(editName(refObj~fullyQualifiedName)), , refObj,    -
                             "first" , "Heading 2"
   end
   else
   do
      if arrQueryIDL~items>1 then
      do
        if sectiontitle <> "" then
        do
           serviceobject = 1
           call insertTitleText xText, sectiontitle, , , "first", "Heading 1", ,          -
                                serviceobject
           call insertTitleText xText, pp(editName(refObj~fullyQualifiedName)),, refObj,, -
                                "Heading 2"
        end
        else
        do
           call insertTitleText xText, pp(editName(refObj~fullyQualifiedName)), , refObj, -
```

```
                                "first" , "Heading 2"
        end
      end
      else
           call insertTitleText xText, pp(editName(refObj~fullyQualifiedName)),, refObj,  -
                                "first", "Heading 1"
   end

/* create overview table */
::routine createTableOverview
   use Arg xDMsf, xText, refObj, cfLayer, arrQueryIDL

      -- set the paragraph style to Heading 3 or 2
   if cfLayer = 2 | arrQueryIDL~items > 1 then
      heading = "Heading 3"
   else
      heading = "Heading 2"

   call insertTitleText xText, pp(refObj~name) "("refObj~unoTypeName")"  "by Name", ,      -
                        refObj, , heading

   xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)
   xTextCursor = xText~createTextCursor   -- create a text cursor
   xTextCursor~gotoEnd(.false)

      -- has members show them ordered in a table
   if refObj~isA(.uno_definitionWithMembers) then
   do
      refObjTypeName=refObj~unoTypeName
         -- get the relation object with members, get at allItems, sort them
      allMembers=refObj~definitions~allItems~sort

      /* create the table */
      xTextTable = xDMsf~createInstance("com.sun.star.text.TextTable")~XTextTable
      nrRows=allMembers~items+1        -- add one more row for headings
      xTextTable~initialize(nrRows, 2) -- initialize the table

      /* insert TextTable in the Text */
      xText~insertTextContent(xTextCursor, xTextTable, .false)

      /* insert the headlines into the table */
      call setCellText xTextCursor, "A1", "Member Name", xTextTable, 10, "BOLD"
      call setColumnWidth "A1", xTextTable
      call setCellText xTextCursor, "B1", "Type",        xTextTable, 10, "BOLD"
      call setColumnWidth "B1", xTextTable

      /* insert the fullyqualified name into the first column
         and the unoName into the second column of the table */
      row=1
      n = 1
      do member over allMembers
        row+=1
        typeName=member~unoTypeName
        if wordpos(typeName, "UNO_PROPERTY UNO_ATTRIBUTE UNO_METHOD")>0 |                  -
           refObjTypeName="UNO_MODULE" then
           call setCellText xTextCursor, "A"row, n .tab1 member~name, xTextTable,          -
                            9, "NORMAL", member, refObj
        else
           call setCellText xTextCursor, "A"row,                                          -
                            n .tab1 editName(member~fullyQualifiedName), xTextTable, 9,    -
                            "NORMAL", member, refObj
        call setColumnWidth "A"row, xTextTable
        call setCellText xTextCursor, "B"row, typeName, xTextTable, 9
        n+=1
      end
      .local~nomembers = .false
   end
   else
   do
      xText~insertString(xTextCursor, "(No members available!)", .false)
      do 3
        xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)
      end
      .local~nomembers = .true
   end
   return

/* create table ordered by types */
::routine createTableByTypes
  use Arg xDMsf, xText, refObj, cfLayer, arrQueryIDL

   if cfLayer = 2 | arrQueryIDL~items>1 then
      call insertTitleText xText, pp(refObj~name) "("refObj~unoTypeName") by UNO Types",, -
                           refObj, , "Heading 3"
```

```
   else
      call insertTitleText xText, pp(refObj~name) "("refObj~unoTypeName") by UNO Types",, -
                           refObj, , "Heading 2"
   xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)

   xTextCursor = xText~createTextCursor   -- create a text cursor
   xTextCursor~gotoEnd(.false)

     -- has members show them ordered in a table
   if refObj~isA(.uno_definitionWithMembers) then
   do
        -- get a relation where the items are associated with the UNO type names
      memberRel=makeUNOTypeRel(refObj)
      uniqueTypes=getUniqueItemTypes(memberRel)

      /* create the table */
      xTextTable = xDMsf~createInstance("com.sun.star.text.TextTable")~XTextTable

      nrRows=memberRel~items+uniqueTypes~items+1
      if refObj~unoTypeName~equals("UNO_CONSTANTS") |                                  -
         refObj~unoTypeName~equals("UNO_ENUM") then
      do
        xTextTable~initialize(nrRows, 2)       -- initialize the table
      end
      else
      do
          -- add one more row per UNO type (for heading) and one for title cell
        xTextTable~initialize(nrRows, 1)         -- initialize the table
      end

      /* insert TextTable in the Text */
      xText~insertTextContent(xTextCursor, xTextTable, .false)

      call setCellText xTextCursor,"A1", "Name", xTextTable, 10, "BOLD"
      if refObj~unoTypeName~equals("UNO_CONSTANTS")  |                                 -
         refObj~unoTypeName~equals("UNO_ENUM") then
      do
        call setCellText xTextCursor,"B1", "Value", xTextTable, 10, "BOLD"
      end
      row=2
      do type over uniqueTypes      -- iterate over all found types
        if memberRel~hasIndex(type)=.false then iterate

        call setCellText xTextCursor, "A"row, type":", xTextTable, 9, "BOLD"
        call setCellBackground "A"row, xTextTable
        if refObj~unoTypeName~equals("UNO_CONSTANTS")  |                               -
           refObj~unoTypeName~equals("UNO_ENUM") then
        do
           xTextTableCursor=xTextTable~createCursorByCellName("A"row) -- create cursor
           xTextTableCursor~gotoCellByName("B"row, .true)
           xTextTableCursor~mergeRange
        end
        if refObj~unoTypeName~equals("UNO_CONSTANTS") then
           call setColumnWidth "A"row, xTextTable
        row+=1

          -- get all items associated with this type, sort them
        tmpArr=memberRel~allAt(type)~sort
        n = 1
        do member over tmpArr
           if wordpos(type, "UNO_PROPERTY UNO_ATTRIBUTE UNO_METHOD")>0 |               -
              refObjTypeName="UNO_MODULE" then
           do
              if member~unoTypeName~equals("UNO_PROPERTY") then
              do
                if member~modifiers~items>0 then
                do
                  tmpStr = ""
                  tmpStr= "("|| member~modifiers~toString(, "+") ||")"
                  call setCellText xTextCursor, "A"row, n "09"x || member~name,        -
                                   xTextTable, 9, , member, refObj
                  call setCellText xTextCursor, "A"row, " "tmpStr,                     -
                                   xTextTable, 9
                end
                else
                  call setCellText xTextCursor, "A"row, n "09"x || member~name,        -
                                   xTextTable, 9, , member, refObj
              end
              else if member~unoTypeName~equals("UNO_METHOD") then
              do
                call setCellText xTextCursor, "A"row, n "09"x || member~toJavaSyntax,  -
                                 xTextTable, 9, , member, refObj, ,"UNO_METHOD"
              end
```

```
            else
              call setCellText xTextCursor, "A"row, n "09"x || member~name,              -
                               xTextTable, 9, , member, refObj
         end
         else
         do
            call setCellText xTextCursor, "A"row, n "09"x ||                            -
                             editName(member~fullyQualifiedName), xTextTable, 9, ,      -
                             member, refObj
            if refObj~unoTypeName~equals("UNO_CONSTANTS")  |                            -
               refObj~unoTypeName~equals("UNO_ENUM") then
            do
               call setColumnWidth "A"row, xTextTable
               if refObj~unoTypeName~equals("UNO_ENUM")  then
               do
                 if member~value = refObj~defaultValue then
                   call setCellText xTextCursor, "B"row, "(default) "member~value,      -
                                    xTextTable, 9, , member, refObj, "right"
                 else
                   call setCellText xTextCursor, "B"row, member~value, xTextTable, 9, , -
                                    member, refObj, "right"
               end
               else
                 call setCellText xTextCursor, "B"row, member~value, xTextTable, 9, ,   -
                                  member, refObj , "right"
            end
            if refObj~unoTypeName~equals("UNO_SERVICE") |                               -
               refObj~unoTypeName~equals("UNO_INTERFACE") then
               do
                  parse var member membertype "[" servicename "|" membername "|"        -
                  wholemembername "|" optional "]"
                  if optional~equals("OPTIONAL") then
                    call setCellText xTextCursor, "A"row, " (" || optional||")",        -
                                     xTextTable, 9
               end
         end

         row+=1
         n+=1
       end
     end
   end
   else
   do
      xText~insertString(xTextCursor, "(No members available!)", .false)
   end
  return

/* create graphic overview */
::routine createGraphicOverview
  use Arg xWriterDocument,xDMsf, xText, refObj, cfLayer, arrQueryIDL

   if cfLayer = 2 | arrQueryIDL~items>1 then
      heading = "Heading 3"
   else
      heading = "Heading 2"

   call insertTitleText xText, pp(refObj~name) "("refObj~unoTypeName") Graphical", , ,  -
                        refObj, heading

   xTextCursor = xText~createTextCursor   -- create a text cursor
   xTextCursor~gotoEnd(.false)
   props = xTextCursor~xPropertySet       -- get the properties for this xTextCursor

   titleSize = 9
   memberSize= titleSize-1
   name      = refObj~fullyQualifiedName

     -- insert IDL name
   call insertShape xDMsf, xText,xTextCursor, refObj~unoTypeName, name, titleSize,      -
                    refObj,  "center"
   xText~insertControlCharacter(xText~getEnd, .CtlChar~paragraph_break, .false)

     -- has members, show them in sorted order
   if refObj~isA(.uno_definitionWithMembers) then
   do
        -- get a relation where the items are associated with the UNO type names
      memberRel=makeUNOTypeRel(refObj)
      uniqueTypes=getUniqueItemTypes(memberRel)
      refObjTypeName=refObj~unoTypeName

        -- create a .endOfLine-delimited string of member names
      do type over uniqueTypes
```

```
        if memberRel~hasIndex(type)=.false then iterate

        tmpStr=""
        do member over memberRel~allAt(type)~sort -- get all members and sort them
           if wordpos(type, "UNO_PROPERTY UNO_ATTRIBUTE UNO_METHOD")>0 |                -
              refObjTypeName="UNO_MODULE" then
              str=member~name
           else
              str=editName(member~fullyQualifiedName)

           if tmpStr="" then
              tmpStr=str
           else
              tmpStr=tmpStr || .endOfLine || str
        end
        call insertShape xDMsf, xText,xTextCursor, type,tmpStr,memberSize,refObj,"left"
     end
  end
  else
  do
     xText~insertString(xTextCursor, "(No members available!)", .false)
  end
  .local~rectangleposy = 0
  .local~rectangleposx = 100
  .local~oldwidth      = 0
  .local~yreduction    = 800
  .local~yHeight       = 0
  .local~selectedUrl   = 0
  .local~layernumber   = 2
  .local~chapternumber = 2
  .local~chaptersubnumber = 0
return

/* Create and insert shapes into passed xText. */
insertShape: procedure
  use arg xDMsf, xText,xTextCursor, title="title n/a", string="string n/a",          -
          charSize=8, refObj, alignment

    xText~insertControlCharacter(xTextCursor, .CtlChar~paragraph_break, .false)

      -- determine maximum width of shape (in chars)
    maxWidth = length(title)
    tmpStr   = string
    do while tmpStr<>""
       parse var tmpStr val (.endOfLine) tmpStr
       maxWidth = max(maxWidth, length(val))
    end
    rectWidth = maxWidth*200 + 500   -- format() used to round result to integer

      -- +1 (one .endOfLine less than items) +1 (header) +1 (empty line)
    rectHeight = (string~countStr(.endOfLine)+1+2)*350+600
    maxrectHeight = (string~countStr(.endOfLine)+1+2)
      -- create and insert a shape object
    shape = xDMsf~createInstance("com.sun.star.drawing.RectangleShape")~xShape
    shape~setSize(.awtSize~new(rectWidth,rectHeight))
      -- set the anchor at_paragraph so the rectangle can be moved to another position
    shape~xPropertySet~setPropertyValue("AnchorType", .anchorType~"AT_PARAGRAPH")
    shape~xPropertySet~setPropertyValue("TextVerticalAdjust",                       -
          bsf.getConstant("com.sun.star.drawing.TextVerticalAdjust", "TOP"))
      -- calculate the position of the rectangles
      -- the first rectangle is created in the middle of the page
    if alignment~equals("center") then
    do
     nodex=((17000/2)-(rectWidth/2))
     nodey= .rectangleposy
     .local~rectangleposy = .rectangleposy + rectHeight
     call shapeTextInsertion shape, nodex, nodey, xText, string, refObj, xDMsf
    end
    else do
     nodex = .rectangleposx
     nodey = .rectangleposy
     summeWidth = rectWidth + .oldwidth

     widthsum = rectwidth + .oldwidth

     if widthsum < nodex then
       widthsum = nodex

     if   (widthsum <=17000) & (.rectangleposy + rectHeight + 200  <= 17000) &      -
         ( rectHeight < 17000 )then
     do
         -- calculate the highest rectangle in one row for the next y position
       if rectHeight > .yHeight then
       do
```

```
        .local~yHeight = rectHeight
      end
      .local~rectangleposy = .rectangleposy - .yreduction
      .local~rectangleposx = .rectangleposx + rectWidth + 300
      .local~oldwidth = rectWidth + .oldwidth + 300
      call shapeTextInsertion shape, nodex, nodey, xText, string, refObj, xDMsf
    end
    else do
    if rectHeight < 17000 then
    do
      .local~yreduction = 0
      -- if the page is full then insert a pagebreak
      if .rectangleposy + .yHeight + 200 + rectHeight > 17000 then
      do
        Props = xTextCursor~xPropertySet
        Props~setPropertyValue("BreakType",.pageBreakType~"PAGE_BEFORE")
        .local~rectangleposy = 0
        .local~yHeight       = 0
      end
      .local~rectangleposy =  .rectangleposy + .yHeight + 200 - .yreduction
      .local~rectangleposx = 100
      .local~oldwidth      = rectWidth + 300
      .local~yreduction    = 800
      nodex = .rectangleposx
      nodey = .rectangleposy
      .local~rectangleposx = .rectangleposx + rectWidth + 300
      .local~rectangleposy =  .rectangleposy  - .yreduction
      if rectHeight > .yHeight then
      do
        .local~yHeight = rectHeight
      end
      call shapeTextInsertion shape, nodex, nodey, xText, string, refObj, xDMsf
    end
    else
    do
      if rectHeight > 18500 & .rectangleposy <=2050 & .rectangleposx < 9000  then
      do
         Call arrangeRectangleHigh string, rectWidth, nodex, nodey, xText, shape,       -
                                   maxrectHeight, xTextCursor, refObj, xDMsf
        .local~rectangleposx = rectWidth + .oldwidth + 300
        .local~oldwidth      = rectWidth + 300
        .local~rectangleposy = .rectangleposy -.yreduction
      end
    else
    do
      do
        Props = xTextCursor~xPropertySet
        Props~setPropertyValue("BreakType",.pageBreakType~"PAGE_BEFORE")
        .local~rectangleposy = 0
        .local~yHeight       = 0
        .local~yreduction    = 0
        .local~rectangleposy =  .rectangleposy + .yHeight + 200 - .yreduction
        .local~rectangleposx = 100
        .local~oldwidth      = 0
        .local~yreduction    = 800
        nodex = .rectangleposx
        nodey = .rectangleposy
        if rectHeight > 18500 then
        do
          Call arrangeRectangleHigh string, rectWidth, nodex, nodey, xText, shape,      -
                                    maxrectHeight, xTextCursor, refObj, xDMsf
        end
        else
        do
          .local~rectangleposx = .rectangleposx + rectWidth + 300
          .local~rectangleposy =  .rectangleposy  - .yreduction
          .local~oldwidth      = rectWidth + 300
          call shapeTextInsertion shape, nodex, nodey, xText, string, refObj, xDMsf
        end
      end
      if rectHeight > .yHeight then
      do
          .local~yHeight = rectHeight
      end
     end
    end
   end
  end
  return

/* method for dividing too high rectangles into 2 rectangles */
arrangeRectangleHigh:
  use arg string, rectWidth, nodex, nodey, xText, shape, maxrectHeight,              -
```

```
          xTextCursor, refObj, xDMsf
      stringi    = string
      string     = ""
      string1    = ""
      rectHeight = 18500
      maxWidth   = 0
      do 50
          -- parse the string and create a new string
        parse var stringi partstring "0a"x rest
        string = string "0a"x partstring
          -- calculate the new width of the first rectangle
        if maxWidth < length(partstring)then
          maxWidth=length(partstring)
        stringi = rest
      end
      maxWidth  = max(maxWidth, length(val))
      rectWidth = maxWidth*200 + 500
      shape~setSize(.awtSize~new(rectWidth,rectHeight))
      call shapeTextInsertion shape, nodex, nodey, xText, string, refObj, xDMsf
      .local~rectangleposx = .oldwidth + rectWidth + 600
      .local~rectangleposy = .rectangleposy - .yreduction
      .local~oldwidth      = .oldwidth + rectWidth + 300
      .local~yHeight       = 18500

      xText~insertControlCharacter(xTextCursor, .CtlChar~paragraph_break, .false)
      maxWidth = 20
      if maxRectHeight <= 101 then
        k = maxRectHeight-50
      else
        k = 50
      do k
          -- parse the string and create a new string
        parse var stringi partstring "0a"x rest
        string1 = string1 "0a"x partstring
          -- calculate the new width for the second rectangle
        if maxWidth < length(partstring)then
          maxWidth=length(partstring)
        stringi = rest
      end
      maxWidth  = max(maxWidth, length(val))
      rectWidth = maxWidth*200 + 500
      if string1~countStr("0a"x) >= 50 then
        rectHeight = 18500
      else
      do
        i = maxrectHeight - 50
        rectHeight=(string1~countStr("0a"x)+3)*350+600
      end
        -- create a second rectangle
      shape1 = xDMsf~createInstance("com.sun.star.drawing.RectangleShape")~xShape
      shape1~setSize(.awtSize~new(rectWidth,rectHeight))
        -- set the anchor at_paragraph so the rectangle can be moved to another position
      shape1~xPropertySet~setPropertyValue("AnchorType", .anchorType~"AT_PARAGRAPH")
      shape1~xPropertySet~setPropertyValue("TextVerticalAdjust",                    -
            bsf.getConstant("com.sun.star.drawing.TextVerticalAdjust", "TOP"))

      if .oldwidth + rectWidth >= 17000 then
      do
        Props = xTextCursor~xPropertySet
        Props~setPropertyValue("BreakType",.pageBreakType~"PAGE_BEFORE")
        .local~rectangleposy = 0
        .local~yHeight       = 0
        .local~yreduction    = 0
        .local~rectangleposx = 100
        .local~oldwidth      = 0
        .local~yreduction    = 800
      end
      nodex = .rectangleposx
      nodey = .rectangleposy


      call shapeTextInsertion shape1, nodex, nodey, xText, string1, refObj, xDMsf
      .local~rectangleposx = .oldwidth + rectWidth + 500
      .local~oldwidth      = .oldwidth + rectWidth + 200
      .local~rectangleposy = .rectangleposy - .yreduction
      .local~yHeight       = 18500

      if maxrectHeight > 101 then
      do
        xText~insertControlCharacter(xTextCursor, .CtlChar~paragraph_break, .false)
        string2  = ""
        maxWidth = 0
        i        = maxrectHeight - 100
```

```
        do i
            -- parse the string and create a new string
          parse var stringi partstring "0a"x rest
          string2 = string2 "0a"x partstring
            -- calculate the new width for the second rectangle
          if maxWidth < length(partstring)then
            maxWidth = length(partstring)
          stringi    = rest
        end
        maxWidth  = max(maxWidth, length(val))
        rectWidth = maxWidth*200 + 500
        if string2~countStr("0a"x) > 50 then
          rectHeight = 18500
        else
          rectHeight = (i+1+2)*350
          -- create a third rectangle
        shape2 = xDMsf~createInstance("com.sun.star.drawing.RectangleShape")~xShape
        shape2~setSize(.awtSize~new(rectWidth,rectHeight))
          -- set the anchor at_paragraph so the rectangle can be moved to another position
        shape2~xPropertySet~setPropertyValue("AnchorType", .anchorType~"AT_PARAGRAPH")
        shape2~xPropertySet~setPropertyValue("TextVerticalAdjust",                              -
              bsf.getConstant("com.sun.star.drawing.TextVerticalAdjust", "TOP"))

        if .oldwidth + rectWidth >= 17000
        then
        do
          Props = xTextCursor~xPropertySet
          Props~setPropertyValue("BreakType",.pageBreakType~"PAGE_BEFORE")
          .local~rectangleposy = 0
          .local~yHeight       = 0
          .local~yreduction    = 0
          .local~rectangleposx = 100
          .local~oldwidth      = 0
          .local~yreduction    = 800
        end
        nodex = .rectangleposx
        nodey = .rectangleposy
        call shapeTextInsertion shape2, nodex, nodey, xText, string2, refObj, xDMsf
        .local~rectangleposx = .oldwidth + rectWidth + 200
        .local~oldwidth      = .oldwidth + rectWidth + 300
        .local~rectangleposy = .rectangleposy - .yreduction
        .local~yHeight       = 18500
      end
  return

  /* method for inserting the string into the rectangle */
  shapeTextInsertion:
    use arg shape, nodex, nodey, xText, string, refObj, xDMsf

      shape~setPosition(.awtPoint~new(nodex, nodey))
      shape~xPropertySet~setPropertyValue("FillColor", .scholz~fillColor)
      xTextContentShape = shape~xTextContent        -- get xTextContent rendering
      xText~insertTextContent(xText~getEnd, xTextContentShape, .false)  -- insert shape

        -- get text object, add text to shape
     xShapeText = Shape~xText
     xShapeTextCursor = xShapeText~createTextCursor
     xShapeTextCursor~gotoEnd(.false)
     props = xShapeTextCursor~xPropertySet          -- get properties
        -- define common properties
     props~setPropertyValue("CharFontName", .scholz~font)
     props~setPropertyValue("CharHeight",   .scholz~entry(charSize))

        -- define property for title text
     props~setPropertyValue("CharWeight",   .scholz~bold)
     xShapeText~insertString(xShapeTextCursor, .linebreak , .false)
     xShapeText~insertString(xShapeTextCursor, title, .false) -- write title
     xShapeText~insertControlCharacter(xShapeText~getEnd, .CtlChar~paragraph_break, .false)
     xShapeText~insertControlCharacter(xShapeText~getEnd, .CtlChar~paragraph_break, .false)

        -- define property for rest of text
     xShapeTextCursor~gotoEnd(.false)
     props~setPropertyValue("CharWeight",   .scholz~normal)   -- reset font to normal

     do 50
      parse var string string "0a"x rest
      call setUrlWithText xText, string, refObj, , , xDMsf, xShapeText,                 -
                          string, xShapeTextCursor
      string = rest
    end
  return

::routine editName
  use strict arg longName, kind="ENCODE"
```

```
   kind=kind~space(0)~left(1)~translate -- get first char
   if pos(kind,"EUP")=0 then            -- default to "E"ncode
      kind="E"

   pos=lastpos(".",longName)            -- a fully qualified name in hand?
   if pos=0 then                        -- no, just return received name
      return longName

   shortName=substr(longName,pos+1)     -- extract unqualified name

   if kind="E" then                     -- encode
   do
      if pos>1 then                     -- path given
         return shortName "("left(longName,pos-1)")"
      return longName                   -- return name unchanged
   end
   else if kind="U" then                -- unqualified name
   do
      return shortName
   end
   else                                 -- path
   do
      if pos>1 then                     -- chars before last dot, i.e. a path ?
         return left(longName,pos-1)
      return ""
   end
  return "n/a"                         -- will never execute

::routine makeUrl
  return  ConvertToURL(directory() || "\" || arg(1))

/* Routine that inserts a text at the end of an XText object. If 'unoRefObject'
   (an UNO_Definition) is given, then define a URL for the text; if in addition
   'xref' is true, then add the string "[xref]" with the appropriate link. */
::routine setUrlWithText
  use strict arg xText, strText, unoRefObject=.nil, xref=.true, parentRefObject=.nil,     -
                 xDMsf=.nil, xShapeText=.nil, string=.nil, xShapeTextCursor=.nil

   xTextCursor=xText~createTextCursor   -- get a text cursor for the xText object
   xTextCursor~goToEnd(.false)          -- move cursor to the end of the xText object


   if unoRefObject=.nil then
   do
      xText~insertString(xTextCursor, strText, .false) -- insert Text
   end
   else
   do
       -- get base URL append fully qualified name of UNO object,
       -- replace dots with forward slashes
      fqn         = unoRefObject~fullyQualifiedName   -- get fully qualified name
      linkTo      = fqn                               -- part of URL
      unoTypeName = unoRefObject~unoTypeName          -- get UNO typename
      if parentRefObject<>.nil then
         parentTypeName = parentRefObject~unoTypeName -- get UNO typename of parent object
      else
         parentTypeName = .nil

      bIsContained = .false   -- part of a container?
      datatype = ""           -- indicate no datatype

      if parentRefObject<>.nil then
      do
          -- a member object (e.g. UNO_SERVICE, UNO_EXCEPTION, etc.?)
          -- no fully qualified name, a member, e.g. a propert
        if unoRefObject~hasMethod("PARENT") then
        do
             -- get parent's (container's) fully qualified name
           linkTo=unoRefObject~parent~fullyQualifiedName
             -- e.g.: http://api.openoffice.org/docs/common/ref/com/sun/module-ix.html
           chunk="#"||unoRefObject~name
           bIsContained=.true
        end
        else if unoRefObject~hasMethod("DEFINEDBY") then
        do
           linkTo=unoRefObject~definedBy     -- get parent's/container's name
           chunk="#"||unoRefObject~name
           bIsContained=.true
        end

        if wordPos(unoTypeName, "UNO_PROPERTY UNO_ATTRIBUTE")>0 then
        do
             -- get Java name of the datatype "javaDatatype:unoDatatype:referen
```

```
        parse value unoRefObject~datatype with datatype ":" .
      end
    end

    linkTo=.scholz~ooo.baseUrl  || linkTo~changestr(".","/")

    if xShapeText <> .nil then
    do
        -- create a text field at the document factory
      xTextProps=xDMsf~createInstance("com.sun.star.text.TextField.URL")~XPropertySet
      xTextProps~setPropertyValue("Representation",string)
      xTextProps~setPropertyValue("TargetFrame", "_blank")
      urlvariable = "URL"
    end
    else
    do
      xTextProps  = xTextCursor~xPropertySet -- get the properties for the text
      urlvariable = "HyperLinkURL"
    end

    if unoTypeName = ("UNO_SERVICE") then
      linkTypeName = "#Services"
    if unoTypeName = ("UNO_INTERFACE") then
      linkTypeName = "#Interfaces"
    if unoTypeName = ("UNO_CONSTANTS") then
      linkTypeName = "#ConstantGroups"
    if unoTypeName = ("UNO_EXCEPTION") then
      linkTypeName = "#Exceptions"
    if unoTypeName = ("UNO_STRUCT") then
      linkTypeName = "#Structs"
    if unoTypeName = ("UNO_ENUM") then
      linkTypeName = "#Enums"
    if unoTypeName = ("UNO_TYPEDEF") then
      linkTypeName = "#Typedefs"
    if unoTypeName = ("UNO_MODULE") then
      linkTypeName = ""

    /* define the link property */
    if bIsContained then          -- part of a collection!
    do
      if parentTypeName = "UNO_MODULE" then
      do
         if .selectedUrl = 2 then
            xTextProps~setPropertyValue(urlvariable,linkTo||"/module-ix.html")
         else
            xTextProps~setPropertyValue(urlvariable,linkTo||"/module-ix.html"||linkTypeName)
      end
      else
      do
         if .selectedUrl = 1 then
            xTextProps~setPropertyValue(urlvariable,linkTo||chunk)
         if .selectedUrl = 2 then
            xTextProps~setPropertyValue(urlvariable,linkTo||".html")
         else
            xTextProps~setPropertyValue(urlvariable,linkTo||".html"||chunk)
      end
      xref=.false                 -- there are no xref's for members
    end
    else
    do
      if unoTypeName = "UNO_MODULE" then
      do
         xTextProps~setPropertyValue(urlvariable,linkTo||"/module-ix.html")
         xref=.false              -- there are no xref's for modules
      end
      else if unoTypeName = "UNO_STRUCT" then
      do
         xTextProps~setPropertyValue(urlvariable, linkTo||".html")
      end
      else
      do
         if .selectedUrl = 1 then
            xTextProps~setPropertyValue(urlvariable, linkTo)
         else
            xTextProps~setPropertyValue(urlvariable, linkTo||".html")
      end
    end
    if xShapeText = .nil then
    do
      xText~insertString(xTextCursor, strText, .false)      -- insert Text
      if datatype<>"" then
        xText~insertString(xTextCursor, " <"datatype">", .false) -- insert datatype info
```

```
        /* insert the xref-URL too */
        if xref=.true then
        do
          xTextProps~setPropertyValue("HyperLinkURL", "")
          xText~insertString(xTextCursor, " ", .false)        -- insert a blank

          refURL=linkTo || "-xref.html"
          xTextProps~setPropertyValue("HyperLinkURL", refUrl)
          xText~insertString(xTextCursor, "[xref]", .false)
        end
      end
      else
      do
        xFieldTextContent = xTextProps~XTextContent
          -- insert the field at the SHAPE text
        paraBreak = .CtlChar~paragraph_break    -- gets used multiple times in this routine
        xShapeText~insertTextContent(xShapeTextCursor, xFieldTextContent, .false)
        xShapeText~insertControlCharacter(xShapeTextCursor, .CtlChar~paragraph_break, .false)
      end
      if xShapeText = .nil then
      do
          -- make sure we remove the URL from now on
        xTextProps~setPropertyValue("HyperLinkURL", "") --set a property value for the cell
      end
   end
  return

/* routine for setting the cell text of the table */
::routine setCellText
    -- define default values for 'charsize' and 'weight' in case they are omitted
  use arg xTextCursor, cell, text, xTextTable, charsize="10", weight="NORMAL",            -
          unoRefObject=.nil, parentRefObject=.nil, alignB, method
   xCellText   = xTextTable~getCellByName(cell)~XText
   cursorProps = xCellText~createTextCursor~XPropertySet
   cursorProps~setPropertyValue("CharColor",    .scholz~charColor)
   cursorProps~setPropertyValue("CharFontName",.scholz~font)
   cursorProps~setPropertyValue("CharHeight",  .scholz~entry(charSize)) -- set font size
   cursorProps~setPropertyValue("CharWeight",  .scholz~entry(weight))
   if  text~pos("(default)")<> 0 then
      cursorProps~setPropertyValue("CharPosture",  .fontSlant~"ITALIC")
   if alignB="right"  then
      cursorProps~setPropertyValue("ParaAdjust", .paragraphAdjust~"RIGHT")
   if method~equals("UNO_METHOD")then
   do
      parse var text returnType type methodName "(" +0 rest
      call setUrlWithText xCellText, returnType, unoRefObject, ,parentRefObject
      call setUrlWithText xCellText, .tab1 type" ", unoRefObject, ,parentRefObject
      cursorProps~setPropertyValue("CharWeight",                                        -
                box("float", bsf.getConstant("com.sun.star.awt.FontWeight", "BOLD")))
      call setUrlWithText xCellText, methodName, unoRefObject, ,parentRefObject
      cursorProps~setPropertyValue("CharWeight",  .scholz~entry(weight))
      call setUrlWithText xCellText, rest, unoRefObject, ,parentRefObject
   end
   else
      call setUrlWithText xCellText, text, unoRefObject, ,parentRefObject
  return

/* routine to set the background color of a table cell */
::routine setCellBackground
  use arg cell, xTextTable
   xTextTableCursor = xTextTable~createCursorByCellName(cell)
   xTextTableCursor~XPropertySet~setPropertyValue("BackColor", .scholz~backColor)
  return

/* routine to change the column width of a table */
::routine setColumnWidth
  use arg cell, xTextTable
   xTextTableCursor = xTextTable~createCursorByCellName(cell)
   xTableProps   = xTextTable~xPropertySet
   colRelSum     = xTableProps~getPropertyValue("TableColumnRelativeSum") --usually: 10000
   colSeparators = xTableProps~getPropertyValue("TableColumnSeparators")  --total sum
   colSeparators[1]~position=trunc(colRelSum*0.70)  -- new position of the column separator
     -- set the property with the calculated position of the seperator
   xTableProps~setPropertyValue("TableColumnSeparators", colSeparators)
  return

/* method to count the UNO-Types the relation contains */
::routine getDatatypeCount
  use arg coll
   count=0
     -- "hasIndex" returns .true=1 or .false=0, hence just add those values up
   do type over .scholz~uno.types
      count=count+coll~hasIndex(type)  -- just add those values up
```

```
    end
  return count

/* return a sorted array containing all unique types found in index */
::routine getUniqueItemTypes
  use arg relation
   set=.set~new
   s=relation~supplier
   do while s~available
      set~put(s~index)
      s~next
   end
  return set~allIndexes~sortWith(.CaselessComparator~new)
```