EN

SEI-BUD Version 5

Reference Manual

MRF

Version 3.00 (18 May 2007)

Table of contents

MRF

Version 3.00 (18 May 2007)

1. INTRODUCTION

1.1. Document history

Version	Status	Date	Authors	Description
1.0	Final	30/07/2003	PDM	Initial version, in French
-	-	15/11/2003	DG T	Translation into English (DG T / COM)
1.1	Final	07/04/2004	PDM	Update for v4.2
2.0	Final	11/10/2004	SAG/ALL	Addition of Amendments system.
2.1	Final	03/11/2004	SAG/ALL	Integration of EUR-OP reception comments
2.2	Final	27/01/2005	GMO	Exhaustive description of the integration process (section “Amendments integration process”)
2.3	Final	03/06/2005	SAG/ALL	Description of Trados TWB integration in seiamd_trad template
2.4	Final	09/03/2006	GMO	Addition of the “Vote Results” report for the European Parliament
2.5	Final	06/06/2006	SAGED	Addition for the new “layout features” of vl0 documents
2.6	Final	15/06/2006	SAGED	New screenshots for the translator functions of the amd Word-based editing tool
2.7	Final	29/06/2006	SAGED	Automatic update of CODICT data coming from CODICT system using the CODICTreader daemon. Addition of reuses in the reuse library for vl0 publications
2.8	Draft	18/09/2006	SAGED	Various updates (WP3.2 - CtrTool [19/7/2006] + corrections + CAT/POL report using metadata of ABB VOL4 document)
2.9	Final	04/10/2006	SAGED	Modification of the client tools in order to make them execute in a Java 1.5 runtime environment for the European Parliament environment
2.10	Final	31/10/2006	SAGED	Introduction of a specific deadline for the translation of the relay languages
2.11	Final	10/11/2006	SAGED	In the AMD CAT/POL generation, new sums end with '.' to avoid the sums like "3.0.12" ==> "3.0.1" and new LinkAndReuse to allow substract like '4' - '4.0.9'
2.12	Final	10/01/2007	SAGED	Added information relative to the new reuse for summary tables of revenue
3.00	Final	18/05/2007	SAGED	Addition of TR/TF

1.2. Purpose of this document

This document is the reference manual for version 5 of the SEI-BUD system.

The first part of the document provides a conceptual and functional description of the system. The second part of the reference manual presents the general system architecture and breaks the system down into several functional sub-systems, going on to document each sub-system. The last part of the reference manual is concerned with exchanges, or the way the SEI-BUD system interfaces with other systems.

This manual is the result of a collaborative and concurrent effort of several people; it has been prepared as a "Volume 0" document using the SEI-BUD system. The layout of "Volume 0" documents is copied from the LegisWrite template used for EU document authoring.

1.3. Background

1.3.1. The SEI-BUD system

SEI-BUD is an acronym for*Système Editorial Informatisé pour les documents BUDgétaires*or 'Computer-aided publishing system for budget documents'. It is an inter-institutional system designed to assist in the publication of the budget of the European Institutions in the 22 official languages, budgetary cycle after

budgetary cycle. It has been operating since 1996, and has been evolving since then to better meet its users' needs and as technological migration requires.

Version 5 of the SEI-BUD system was developed and used in April 2007 to produce the Preliminary Draft of the 2008 Activity Based Budget (ABB). The many functions of this version of SEI-BUD include:

1) coverage of the ABB (Activity Based Budget – a Commission initiative);

2) foundations capable of handling other publications (for example, Volume 0);

3) technological updates to the system (application server, Java, EJB);

4) system open to XML (more tools available than with SGML);

5) system open to XSL (instead of the proprietary style sheet language SiT (SGML Integrated Toolkit));

6) New interface POETRY for TR (Translation request) and TF (Translation Follow-up).

7) Less maintenance and production support is needed due to the simplified system.

1.3.2. SEI-BUD and the handling of EPs (Budget Estimates) and of Volume 0

Between November 2002 and March 2003, developments to the system were introduced to make sure it could handle the production of the Commission's EPs and Volume 0 of the Commission. Between March 2003 and May 2003, the SEI-BUD system was used to produce:

1) the Budget Estimate for the 2004 revenue and expenditure of the Commission

2) the 2004 Volume 0 of the Commission

3) the 2004 Preliminary Draft Budget

The ability to handle the production of the Budget Estimates and Volume 0 brought with it an important new function:**REUSE**. Briefly, this function allows structured information to be reused effectively within and between budget documents. This important new function ties in with the key objectives behind the use of the SEI-BUD system, namely:

1) **to improve the quality**of publications (with repeated information being produced by REUSE, in all languages, publications no longer contain inconsistencies)

2) **to reduce production costs**(by reducing the workload of the author, translator and corrector)

3) **to reduce lead times**(by reducing the workload of the author, translator and corrector)

1.3.3. SEI-BUD and the handling of Amendments

The Amendment subsystem fulfils the following functions:

1) entry of the amendment data into the repository;

2) registry of the various budgetary procedure steps: tabling, vote, etc;

3) consultation of repository information;

4) production of various reports;

5) integration of adopted amendments in a target Budget publication .

The Amendment subsystem has been mostly rewritten starting from end 2003 to mid 2004 in order to achieve a technological unification with the version 4 of the SEI-BUD system. However the old Web user interface (known as the WebInt module) has been left unchanged for a large part.

2. CONCEPTUAL AND FUNCTIONAL DESCRIPTION OF THE SEI-BUD SYSTEM

2.1. The budget of the European Institutions

To achieve the tasks it has to accomplish, the European Union has a budget for 2003 of €99.7 billion.

The Community financial system relies on the medium-term programming of expenditure (the financial perspective), which is the result of an agreement between the European Parliament, the European Council and the Commission. The current agreement dating from 6 May 1999 defines ceilings for the period 2000-2006 for the various categories of Community expenditure: agriculture, cohesion policy, internal EU policies, external policy and preparation for enlargement, and administrative expenditure.

This programme produces an annual budget at the end of a procedure involving the Commission on the one hand, and the Council and Parliament, with decision-making powers, on the other. The budget determines the amount of expenditure allocated to each policy area in which the European Union is active.

2.2. The budgetary process

In practice, in 1977, a practical timetable was agreed between the three institutions, and the various stages of the process have proceeded as follows since that time:

— The**preliminary draft budget (PDB)**is prepared by the Commission and sent to the budgetary authority by 15 June at the latest.

After an internal exploratory debate, which provides an opportunity for defining the major political and budgetary priorities for the coming financial year, the Commission prepares its budget estimate, by gathering together all the requests from spending agencies and proceeding to internal arbitration. It also takes account of the conclusions of a 'trialogue' between the three institutions on budgetary priorities. In addition, it receives budget estimates from the other institutions, and puts them all together in a preliminary draft budget (PDB), which constitutes the overall forecast of revenue and expenditure for the coming financial year. This document is adopted by the College at the start of May, and sent to the budgetary authority in all the Community languages by 15 June at the latest.

The Preliminary Draft Budget (PDB) may be amended by the Commission in an**amending letter**, to take account of new elements that only became apparent after the budget was prepared.

— The preparation of the**draft budget (DB)**by the Council

The Council proceeds with its first reading, and on the basis of the Preliminary Draft Budget (PDB) and following consultation with a delegation from the European Parliament, it adopts a draft budget before 31 July that it sends to the Parliament during the first fortnight of September. Alongside this reading, an ad hoc consultation process is also taking place on the subject of compulsory

expenditure to be entered in the budget; first, though, a 'trialogue' is held between the institutions towards the end of June.

— The Parliament first reading (PEL1)

Using the draft prepared by the Council, the Parliament commences its first reading during October; amendments relating to non-compulsory expenditure (NCE) require the majority support of MEPs. Proposals for modifications to compulsory expenditure (CE) require a majority of the votes cast, when agreement on the amount of this expenditure has not been found during the ad hoc consultation.

— The Council second reading (COL2)

The Council carries out this second reading during the third week in November, following consultation with a delegation from the European Parliament. The draft budget may be modified on the basis of amendments (for non-compulsory expenditure (NCE)) or proposals for modifications (for compulsory expenditure (CE)) accepted by the Parliament.

The outcome of the Council's deliberations in the second reading on compulsory expenditure normally leads to the setting of final figures: indeed the Council really has the last word on this category of expenditure, unless the Parliament rejects the whole budget. The amended and modified draft budget is once again submitted to the Parliament around 22 November.

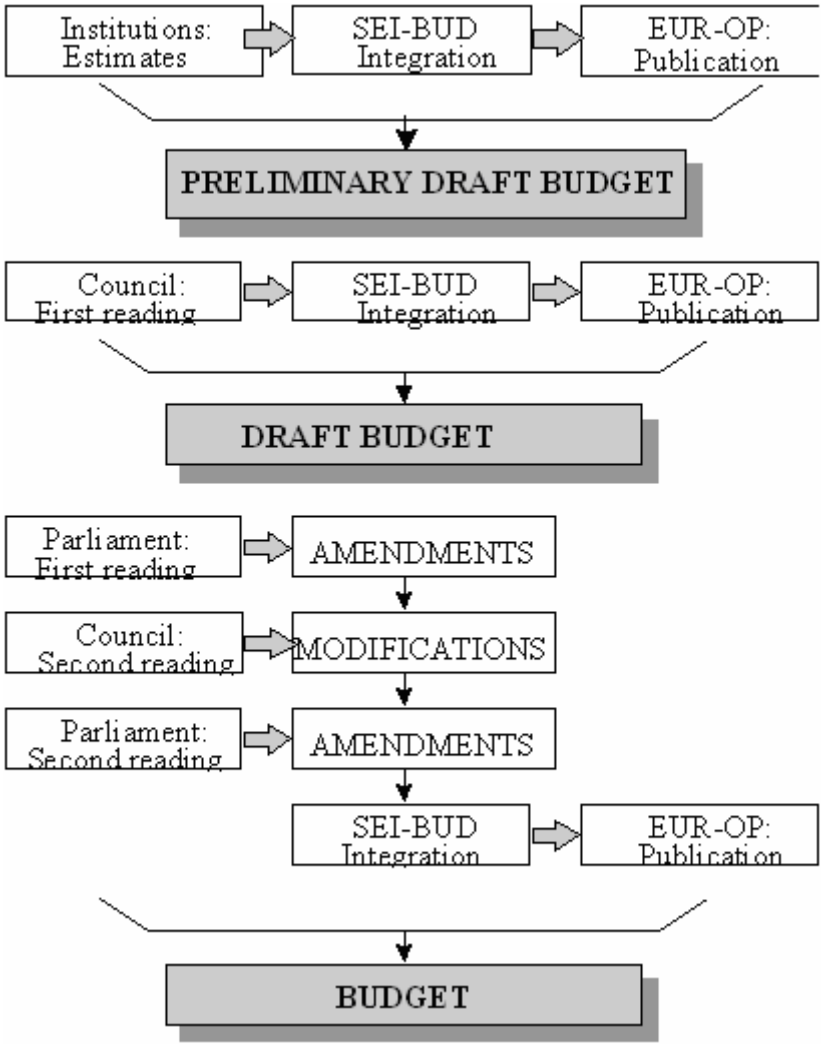
— The Parliament second reading (PEL2) and the adoption of the budget (B)

Now that the Council has given its last word on compulsory expenditure, the Parliament devotes most of the plenary session in December to examining non-compulsory expenditure, on which it may accept or reject the Council's proposals. The Parliament, with a majority of MEPs and three fifths of the votes cast, then adopts the budget. The budget is declared adopted by the President of the Parliament and becomes enforceable.

In the event of unavoidable, exceptional or unexpected circumstances, the Commission may have to propose modifications to the approved budget during the financial year, in the form of preliminary draft amending budgets. Amending budgets are also used to enter the balance left over from the previous year into the budget for the current financial year.

These amending budgets (ABs) are subject to the same rules of procedure as the general budget.

The diagram shown below summarises the different stages in the budgetary process and the different publications with which they are associated:



2.3. The budget structure

The budget structure includes sections on revenue and expenditure:

2.3.1. Revenue

The first part of the budget of the European Union is a general statement of Revenue. In addition to the EU's own resources, this statement sets out different categories of revenue such as taxes, revenue from the administrative operation of the institutions, interest on late payments and fines and, in some cases, the excess from previous financial years.

Own resources include:

— agricultural contributions (agricultural duties on the import of agricultural products resulting from the common organisation of the market, and sugar and isoglucose levies provided for within the framework of the common organisation of the market in sugar);

— customs duties from the application of the Common Customs Tariff to the customs value of goods imported by third countries;

— the VAT resource, from the application of a uniform percentage rate to the VAT assessment base of each Member State, harmonised according to Community rules;

— an additional resource from Gross National Product (GNP), from the application of a rate to the sum of the Gross National Product (GNP) of all the Member States, determined each year under the budgetary procedure in the light of all other revenue in the budget. It is a resource used to balance the budget, in order that total revenue will cover the total payment appropriations needed to finance the expenditure entered in the budget of a particular financial year.

2.3.2. Expenditure

The statement of Expenditure in the General Budget is split into six sections: the European Parliament (section I), the Council (section II), the Commission (section III), the Court of Justice (section IV), the Court of Auditors (section V), the Economic and Social Committee and the Committee of the Regions (section VI). The appropriations entered under sections I, II, IV, V and VI are administrative appropriations only. The appropriations entered under section III (Commission) are of two types:

— administrative appropriations, covering the Commission's expenditure on staffing, premises and equipment, publications, computer systems, the administration of delegations, expenditure on specific tasks such as subsidies and some interinstitutional expenses, such as the pensions of officials and temporary agents in all the institutions.

— operating appropriations.

2.3.3. The specific case of the Commission: Activity Based Budgeting (ABB)

The establishment of Activity Based Budgeting (ABB) is the first component of the broader Activity Based Management (ABM), which was adopted by the Commission and on which the Commission's services have been working since 1999. Its principal aim is to ensure that the allocation of resources is a political process within the framework of which resources of all types are allocated in a way that is compatible with the political priorities and objectives already defined. Thus, the setting of priorities, planning, preparation of the budget, monitoring and reporting are processes within a common conceptual framework where the activity is the common denominator.

As a result, the structure of the Commission's budget is undergoing a significant transformation and becoming politically more significant. The 2004 budget will be the first officially to follow the ABB structure. The traditional separation of administrative resources and operating resources is therefore replaced by**a structure that presents the Commission's resources according to policy area and activity.**A structure along these lines has been prepared and adopted by the Commission's services and will constitute the basis for the budget planning and preparation processes and their respective reporting processes.

In addition, the political management of the process of allocating resources begins at the stage of the Annual Policy Strategy decision, where the orientations are set for each of the Commission's policy areas and then integrated into the preparation of the preliminary draft budget.

2.4. Players in the budgetary cycle

The aim of the budgetary cycle is the 'paper' publication of the budget in the 22 official languages. Besides the Institutions, the Publications Office and the printer are also important players in the budgetary cycle.

The players in the budgetary cycle are as follows:

— the Commission (volumes 1 and 4)

— the Council (volume 3)

— the Parliament (volume 2)

— the Court of Justice (volume 5)

— the Court of Auditors (volume 6)

— the Economic and Social Committee (volume 7)

— the Committee of the Regions (volume 8)

— the European Ombudsman (volume 9)

— the Publications Office

— the printer

As will be explained, the SEI-BUD system is designed to solve the problems inherent in the participation and collaboration of such a large number of players in achieving a common objective: the publication of the budget in the 22 official languages.

2.5. Production constraints

The following constraints are associated with the production of the budget documents:

— The **volume**of data to be processed: the budget is a publication of around 2 000 pages, published in the 11 official languages (i.e. publication of around 22 000 pages), three times a year (Preliminary Draft Budget, Draft Budget and Final Budget, i.e. publication of around 66 000 pages per year). (Following enlargement, the annual publication will be of around 120 000 pages per year). Below, we will look at how far the SEI-BUD system has provided a solution to this constraint.

— Production **deadlines**: the budget is a very sensitive matter and, despite the volume of information to be handled, it is not difficult to see how important it is that deadlines are respected in view of its political role, the players involved, and the need for compliance with the budget preparation procedure. Below, we will look at how far the SEI-BUD system has provided a solution to this constraint.

— The **technical constraints**linked to the IT standards of the Institutions: the production of the budget of the Institutions is the result of an interinstitutional effort involving authors (from all the Institutions), translators (from all the Institutions), correctors (from the Publications Office) and a printer. Because of this, the idea of any attempt at computerisation imposing a particular publishing tool on the process could never have been contemplated. One of the constraints is therefore that all users must be able to use their own office computer tools to make changes to the budget publications. For example, during the 1997 budget year (PDB 1998, DB 1998, B 1998), the SEI-BUD system had to cope with the following:

  — during the production of the 1998 PDB:

    — the Commission author used **Word 6.0**to make changes to the budget documents

    — the Commission translator used **WordPerfect 5.2**to translate the budget documents

    — the Publications Office corrector used **Interleaf**to correct the budget documents

    — the printer used the standard **Formex V2rev** to print the paper copy of the budget documents

  — during the production of the 1998 budget:

    — the authors and translators at the European Parliament (EP) used WordPerfect 6.1 as their editing tool

— the following year, for the 1999 budget procedure:
  — the authors and translators all used **Word 97**to edit the notes (except for the ESC, which used Word 95)
  — the correctors at the Publications Office used **Adept Editor**instead of Interleaf
  — the printer used the standard **Formex V3**instead of Formex V2rev to print the paper copy of the budget documents
— etc.

This example illustrates the constraint associated with the use of multiple tools and formats (Word 6.0, Word 95, Word 97, WordPerfect 5.2, WordPerfect 6.1, Interleaf, Adept Editor) for the publication of the budget documents. Below, we will look at how far the SEI-BUD system has provided a solution to this constraint.

— the constraint of**publication quality**: working from the principle that 'it's true if it's in print', the quality of budget publications is of great importance. Below, we will look at how far the SEI-BUD system has provided a solution to this constraint.
— the constraint of**quality associated with the multilingual nature**of the publication:
  — the workload associated with any modification of the budget is multiplied by 11 on account of the linguistic dimension (by 20 in 2004);
  — equivalence between the different language versions (synoptism) must be guaranteed.
— the constraint associated with the geographical dispersal of the various players involved in producing the budget documents:
  — the Commission is located in Brussels, in a number of different buildings;
  — the Parliament is located partly in Brussels and partly in Luxembourg;
  — the printer is located in France;
  — etc.
— etc.

2.6. The objectives associated with the implementation of the SEI-BUD system

In view of the constraints presented above, the objectives of the SEI-BUD systems are:

— to improve the**quality**of budget publications
— to reduce lead times
— to reduce**costs**

2.7. SEI-BUD: An XML (SGML) strategy

The SGML (Standard Generalised Mark-up Language) standard is the technology on which the SEI-BUD system is based. The strategy of adopting this technology is the basis of the success of the publishing system in an interinstitutional context, improving the quality of budget publications, reducing production times and reducing production costs. Since 2002, this implementation strategy based on SGML technologies has evolved into a strategy using XML technologies.

The table below shows how far this strategy has enabled the implementation of functions that satisfy the constraints described above, resulting in the fulfilment of the objectives set:

Function	Description/comments	Impact on		
		Better quality	Lower costs	Shorter lead times
XML (SGML) for the validation (conformance) of budgetary instances	The use of XML/SGML to save budgetary data ensures the 'conformance' of budgetary data with respect to a chosen structure (grammar). This 'quality control' (validation with respect to the DTD each time a fragment of a budget document is updated) must have a positive impact on costs and lead times throughout the production process. For example, an error corrected at the author stage is not introduced into the 10 other language versions, and does not therefore have to be corrected in the 11 different language versions.	+++	+	+
XML (SGML) to control the synoptism of budgetary instances	Using the XML strategy within the framework of SEI-BUD means that the synoptism of each language edition can be checked (for example, when a translator update is made) with respect to the author reference. In addition to the obvious impact on the quality of budget publications, this 'quality control' must have a positive impact on costs and lead times throughout the production process.	+++	+	+
XML (SGML) to ensure the continuing validity of the information	New versions of budget documents (PDB, DB, B) are always produced by modifying the previous version of the budget document. For example: — the 2003 PDB is produced by modifying the 2002 B; — the 2003 DB is produced by modifying the 2003 PDB; — etc. So the same document basis is used, modified and adapted from one production (PDB, DB, B) to the next, from year to year. It is therefore essential for this document basis to be protected from the obsolescence associated with market-led technological migration (for example: WordPerfect 5.2, Word 6.0, Word 95, Word 97, Word XP). In addition, reusing the document basis from one production to the next has a positive impact on quality, and reduces costs and lead times (for example, anything that has been translated and corrected and is not changed by an author does not have to be translated or corrected again).	++	+++	+++
XML (SGML) and simultaneous editing	The XML (SGML) strategy has meant that systems can be implemented to enable simultaneous access to different parts of the same budget document. The XML repository implemented within SEI-BUD permits access, locking and versioning at an XML document node. This is an essential function for authorising simultaneous access for several authors to the same budget document (access to different fragments). The function is also important in terms of the reduction (compression) of production times. It not only enables several authors to work simultaneously on the same budget document, but it also allows translators to work on fragments released by the authors without having to wait until the full document is in translation, and allows correctors to work on fragments released by the translators, without having to wait until the full document is in correction, etc. N.B.: You may think that this function could have been provided by physically cutting up the budget document into several sections (e.g. several Word documents). Without going into too much detail, the limitations of this solution would quickly become apparent, particularly if concurrent access were combined with 'access on demand' (see below).		+	+++
XML (SGML) and 'access on demand'	As explained above, the XML strategy employed in SEI-BUD provides the capability of accessing fragments of a budget document. Virtually, this access (the XML repository used allows this) can 'descend' to the smallest element of a budget document (a single paragraph, for example). In reality, the levels of access correspond to the budget nomenclature entries (revenue, expenditure, title, chapter, article, item, annex). In short, a user can access a fragment of a budget document 'on demand'. The choice of level depends on the type of user. For example, authors generally work at the level of the budget TITLE, while translators work at CHAPTER level. This function is important for reducing lead times (the smaller the level of segmentation, the more concurrent working will be possible and the more lead times will be reduced).		+	+++
XML (SGML) as standard (pivot) exchange format	The adoption of XML as the pivot format for storage means that all proprietary formats can be handled without loss of information following conversion. These are: — Word 6.0, Word 95, Word 97 and soon Word XP — WordPerfect 5.2, WordPerfect 6.1 — Interleaf — etc.	+++	+++	

	(see also 'continuing validity' and the preservation of the document basis faced with technological evolution.)			
XML (SGML) for automating carryovers	The XML strategy as implemented for SEI-BUD (use of a repository allowing access at the level of an XML document node) means that all author modifications can automatically be carried over to other language versions. For example, if an author deletes a paragraph, this paragraph will automatically be removed from all the language versions, and no translation work will be required. In addition to the obvious benefits of this function in terms of production times and costs, automating the transfer of author modifications improves publication quality significantly, since it guarantees equivalent content in all languages (especially budget figures or figures in external tables).	+++	+++	+++
XML (SGML) and productivity tools	The printing of differences, which can be 'reworked' depending on the person for whom it is intended, is a very important productivity tool in SEI-BUD. The printing of differences is used: — by authors, so they can check the changes made to a budget document; — by translators, so they can easily see and locate changes that actually need translating; — by correctors, so they can easily see and locate changes by translators that may need correcting. The benefits in terms of quality, production times and costs associated with the implementation of these tools are easy to comprehend.	+++	+++	+++

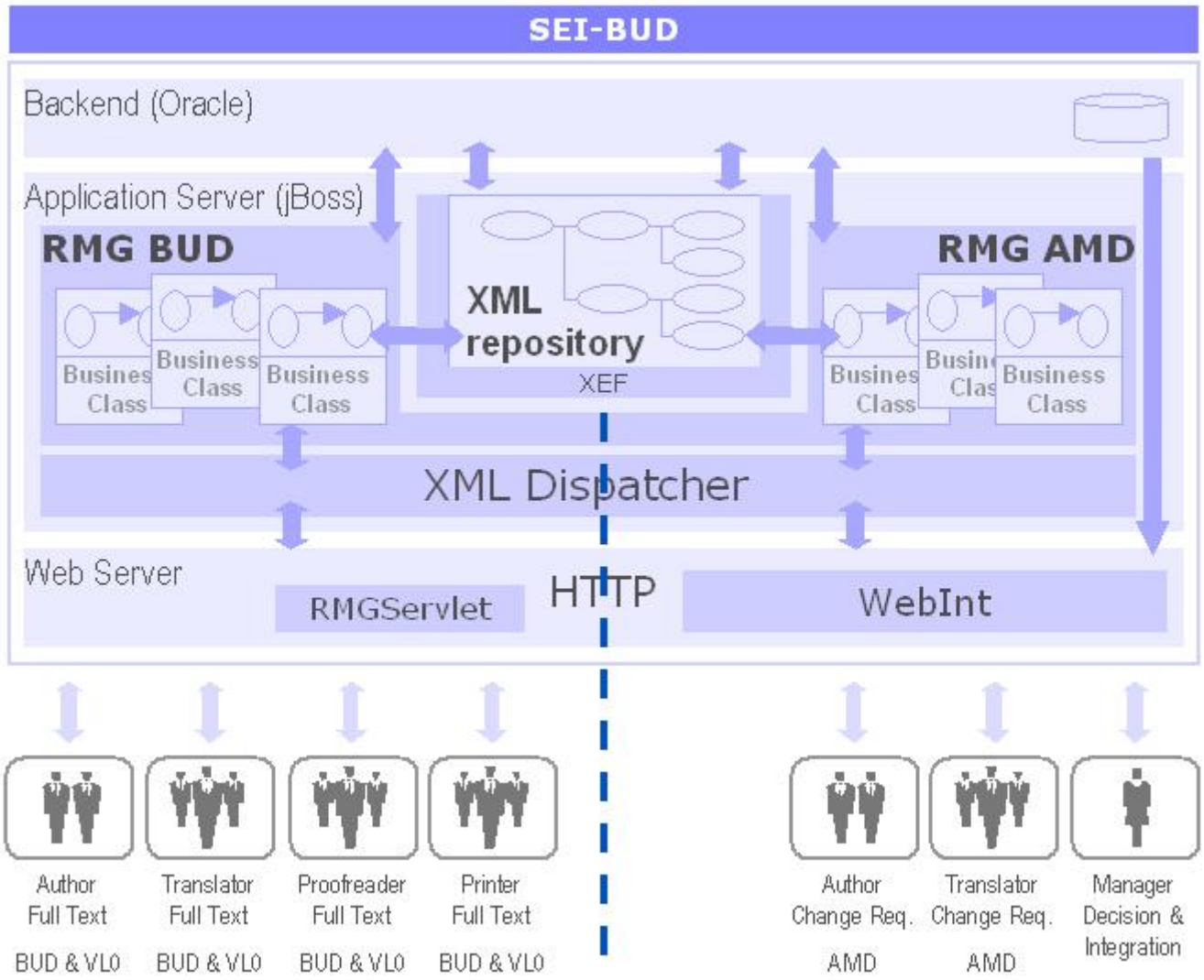
2.8. SEI-BUD amendments

For a conceptual and functional description of the Amendments subsystem, the reader is referred to the first section of the following manual: "Gestion des amendements budgétaires: Manuel de l'auteur d'amendements".

3. ARCHITECTURE OF THE SEI-BUD SYSTEM

3.1. General architecture

The general architecture of the unified SEI-BUD system can be functionally represented as follows:



— All classes of users (authors, translators, etc) access the system via a Web Server using the HTTP protocol; some users access the Web Server using a dedicated client: the Control Tool and some other users preparing and manipulating amendments access the Web Server using a Web Browser (Internet Explorer);

— The Control Tool uses an XMLDispatcher client to issue requests to the Web Server (in fact to a small servlet named RMGServlet) using the HTTP protocol with requests and responses serialized in XML.

— The WebInt module is a servlet running inside the Web Browser and responsible for managing the user interface for amendments; the WebInt servlet makes JDBC queries to the oracle database.

— Both the WebInt and the RMGServlet issue request to the EJB based application server, the so-called Repository ManaGer (RMG) using an XMLDispatcher; these requests are described in section 5.2 (Definition of business objects)

— The Repository ManaGer (RMG) contains business objects or business classes under the form of Enterprise Java Beans (EJB); the unified Repository ManaGer contains beans for managing budget documents and beans for managing amendment to budget documents.

— Some business objects manage documents ("AbbDocument", "AmdDocument",...) and use services of the XML Editorial Framework (XeF) which is mainly a "multi-lingual" layer on top of the XML repository;

— Some other business objects use a JDBC connection to store/retrieve their persistent state into the Oracle database; this connection is obtained from the JBoss EJB container;

— The XML repository manages XML documents (whether budget documents or amendment documents); it does not maintain the relationship between different linguistic versions (as mentioned above, this is the XeF responsibility). The XML repository uses a JDBC connection to the Oracle database to store/retrieve XML documents; this JDBC is passed by business objects who obtain it from the JBoss EJB container.

— All JDBC database accesses (whether issued by business objects or by the XML Repository on behalf on business objects) are performed inside transactions controlled by JBoss.

3.2. General architecture layout (subsystems)

The architecture of the SEI-BUD system is designed in several logical tiers, with each tier depending only on the tier situated immediately below it.

The use of three-tier architecture is now traditional. The main objective of this separation into three tiers is to isolate the business logic from the code behind the user interface (presentation logic) and prevent this code from having direct access to the stored data. A system implemented using this architecture is easier to maintain because the user interface can be modified without changing the logic.

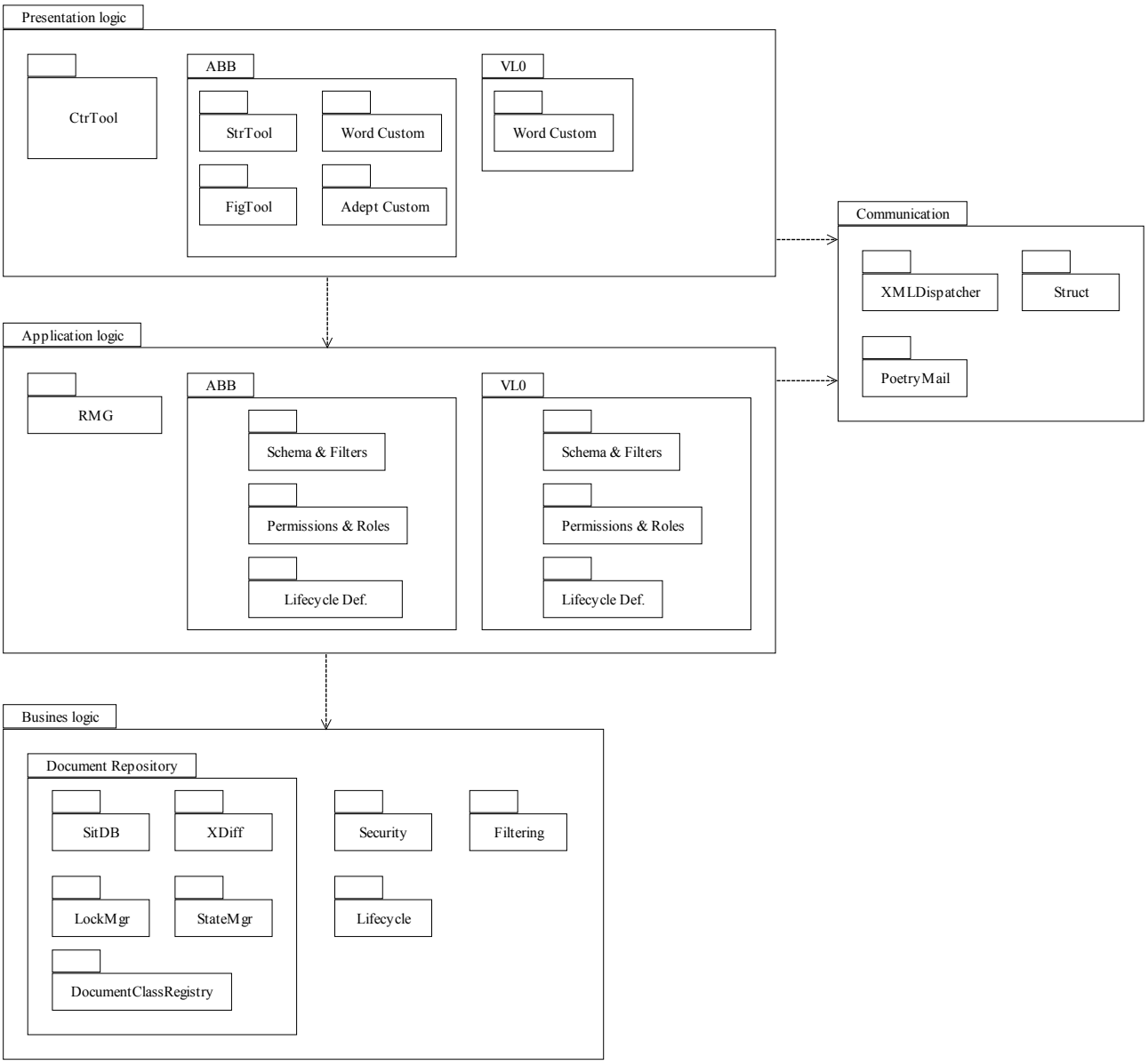
A three-tier architecture can be insufficient when there are important requirements in terms of modularity and maintainability. The data model of the business logic tier is not generally exposed directly by the presentation tier. The presentation tier constructs a view of this model by selecting and reorganising its data. To increase modularity and maintainability, there is some benefit in extracting the code giving access to the business objects (controllers) from the presentation tier. Together, the controller objects form an intermediate tier between the business logic tier and the presentation logic tier. This additional tier implements the application logic.

Setting up an application tier that is separate from the presentation tier also reduces network traffic when the presentation tier is deployed on a physically separate (fat client) node. When the presentation tier accesses objects directly in the business logic tier, exchanges over the network are multiplied, which can lead to significant waiting times in the processing of user requests. By deploying the application tier on the same physical node as the business logic tier (server machine), exchanges over the network are reduced. A user request may even require only one exchange.

The introduction of an application logic tier also makes more reusable business objects possible. By introducing the distinction between business logic and application logic straight away, the application's own processing can be extracted from the business tier at any time.

The diagram below gives a general view of the tiered architecture of the SEI-BUD system.

N.B.: This diagram is a conceptual representation of the SEI-BUD architecture. Not all the packages shown are necessarily present in the system's hierarchy of Java classes.



The**business logic tier**implements the basic functions of the publishing system: an XML repository giving access to documents by fragment, in consultation and locking modes. The repository also makes it possible to access the different versions of a document and highlight changes between the versions. The repository also maintains status information associated with the document fragments.

The**application logic tier**

The**presentation logic tier**includes a fat client, CtrTool (the control tool) and several text editors. These include configured text editors (Word, Adept Editor) and specially developed tools: a figures tool (FigTool) and structure tool (StrTool).

The tier that handles the storage of data is not represented. This consists of an Oracle database system and its driver.

3.3. Brief description of the different subsystems

3.3.1. The Repository Manager (RMG)

The Repository Manager subsystem implements the business objects of the SEI-BUD system. These business objects are defined and accessible in terms of EJB (Enterprise JavaBeans).

The various components of this sub-system can be split into two groups, depending on whether or not they are implemented as business objects accessible by clients.

Business objects

— The server object ("**Server**") is the SEI-BUD server independent of any publication.

— The publication object ("**AbbDocument**") offers different methods of access to two classes of publication supported by the SEI-BUD system: ABB and Volume 0 publication.

— The processing report object ("**ProcessingReport**") gives methods of access to processing reports for requests, whether or not these are currently being processed.

— The server activity journal object ("**FollowUp**") gives the different methods of access to the server activity journal.

Other components of the RMG subsystem

The other components of the RMG subsystem are 'technical' components that mainly inherit certain 'generic' components from the Xef framework; the following components are grouped in the package "`com.softwareag.belgium.seibud.rmg.bo`":

— The**life cycle managers**implemented by the classes "`AbbLifeCycle`" and "`Vl0LifeCycle`" based on the class "`LifeCycle`" of the Xef framework;

— The**role managers**implemented by the classes "`AbbRoleClass`" and "`Vl0RoleClass`" based on the class "`RoleClass`" of the Xef framework;

— The**permissions manager**implemented by the class "`PermissionClass`" based on the class "`BasicPermission`" of the Xef framework;

— The**Poetry translation requests manager**implemented by the class "`PoetryIdentifier`";

Other component utilities are grouped in the package "`com.softwareag.belgium.seibud.rmg.tools`"; these are:

— The**connection context manager**is implemented by the class "`RMGCtx`", which is based on the class "`XMLDispatcherContextImpl`";

— The**request logger**is implemented by the class "`RqstLogger`";

— The**persistent logcreator**is implemented by the class "`LogDb`";

— The**Poetry requests switcher**is implemented by the class "`PoetryMailDispatcher`".

3.3.2. The Xef Framework (XML Editorial Framework)

3.3.2.1. Principle

The Xef framework groups together a number of functions relating to the management of large multilingual documents in a collection of expandable and configurable modules.

The different functions of the Xef editorial tier are generally used by characterising by inheritance the operation of certain classes.

The Xef framework has a configuration file communicated by the RMG tier. In the SEI-BUD project, the name of this file is "document-repository-config.xml".

3.3.2.2. The modules of Xef

The different modules of this framework are grouped into packages:

— The package "`com.softwareag.belgium.xef.impl.delta`" groups together the**facilities for the comparison**of documents or fragments of documents;

— The package "`com.softwareag.belgium.xef.impl.filters`" is the**filtering framework**;

— The package "`com.softwareag.belgium.xef.impl.lifecycle`" implements the**lifecycle management**;

— The package "`com.softwareag.belgium.xef.impl.security`" implements**security management**(users, roles, permissions, groups, etc.);

— The package "com.softwareag.belgium.xef.impl.xpath" implements an**XPath resolution**;

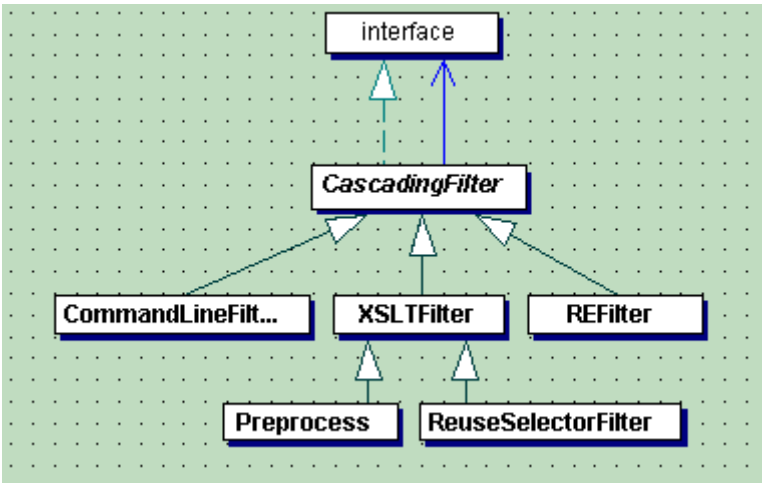
— The package "`com.softwareag.belgium.xef.impl.repository`" manages different basic concepts of the framework:**document**,**repository**,**format**,**lockmanager**,**versioned XPath**; it contains the following subpackages:

  — "`toc`" covers the functions associated with the table of contents or**nomenclature**;

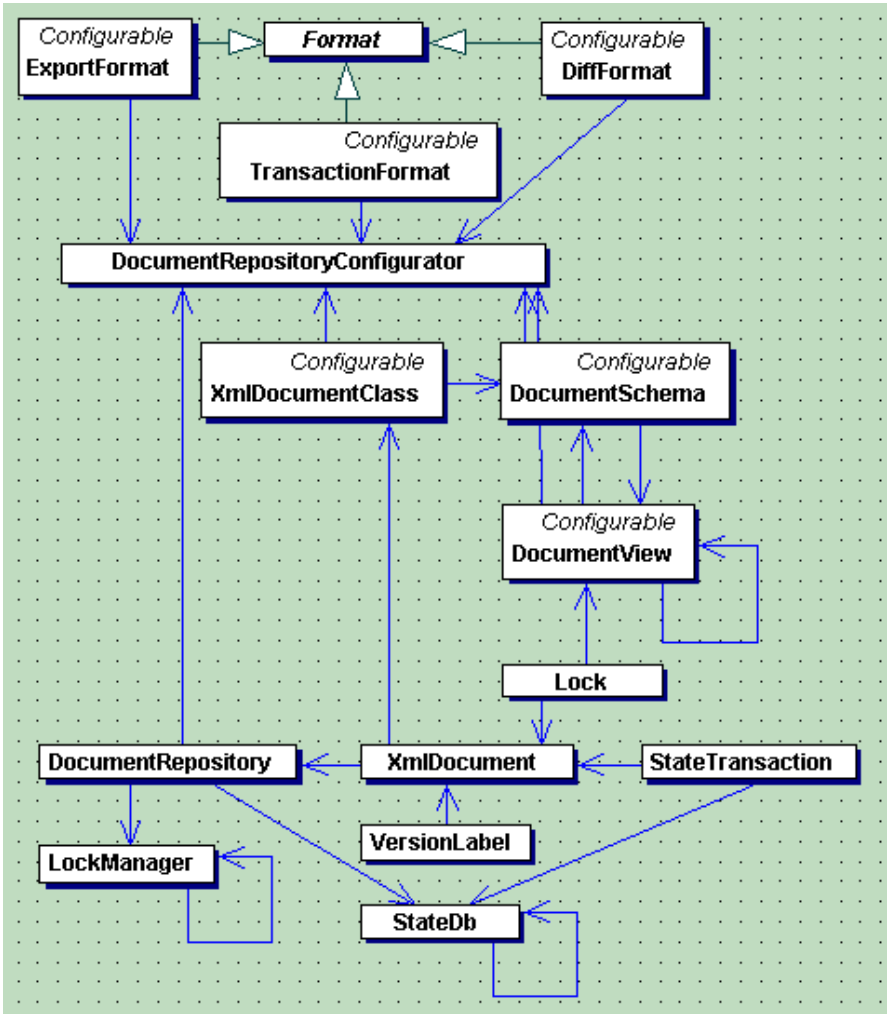
  — "`state`" implements the**fragment state**function;

  — "`merge`" manages the**merging of branches**.

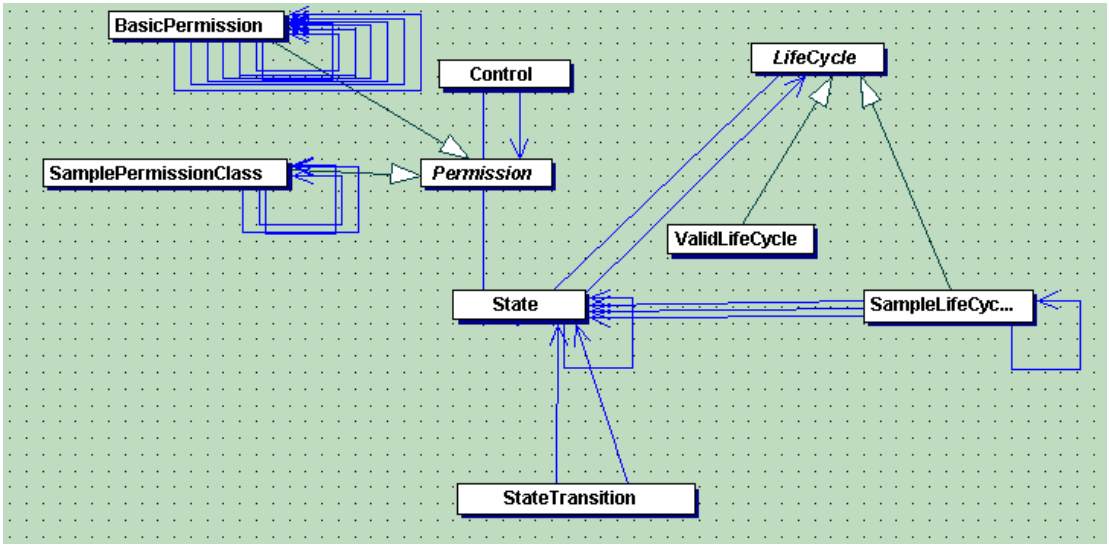
The simplified class diagram shown below gives an idea of the components involved in the management of filters:



The simplified class diagram below shows the relationships between some of the key classes in the Xef framework (the classes "`Xmldocument`", "`Format`", "`DocumentRepository`", and "`DocumentView`"):



And the last simplified class diagram shows the relationships between "`LifeCycle`", "`Permission`" and "`State`":



3.3.2.3. The configuration file for the basic Xef functions

This configuration file is modelled by a DTD ("document-repository-config.dtd").

```
<!ELEMENT document-repository (url-base, filepath-base, exploit-dir, schemas, formats,
document-classes)><!ELEMENT url-base (#PCDATA)>
<!ELEMENT filepath-base (#PCDATA)>
<!ELEMENT exploit-dir (#PCDATA)>
```

The element "`url-base`" contains a URL prefix to locate the other resources needed by Xef. For example: "".

The element "`filepath-base`" indicates a directory containing certain files such as DTDs and schemas. For example: "*/u01/users/seibud/catalog*".

The element "`exploit-dir`" indicates the operating directory (see "Operating Manual").

The element "`document-repository`" contains information used by a number of schemas, formats and document classes.

```
<!ELEMENT schemas (schema*)>
<!ELEMENT schema (name, descr, path, toc-entry-locator, delta?, synoptism, discard-
presentation, discard-presentation-delta, views)>
<!ATTLIST schema type (DTD | XMLSCHEMA) #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT descr (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT toc-entry-locator (#PCDATA)>
<!ELEMENT delta (#PCDATA)>
<!ELEMENT synoptism (#PCDATA)>
<!ELEMENT discard-presentation (#PCDATA)>
<!ELEMENT discard-presentation-delta (#PCDATA)>
```

A schema has a name ("`name`"), a description ("`descr`"), a filename ("`path`") that relates to the parameter "`filepath-base`", and a class ("`toc-entry-locator`"), which is invoked by Xef to determine the content of the table of contents. The element "`delta`" provides information for the class responsible for differentials, while the element "`synoptism`" provides information for an XSLT style sheet that filters document fragments to check for linguistic synoptism. The elements "discard-presentation" and "discard-presentation-delta" refer to style sheets that filter presentation information.

```
<!ELEMENT views (view*)>
<!ELEMENT view (name, descr, selector, view-refs)>
<!ATTLIST view language-dependent (y | n) #REQUIRED toc-edition-enabled (y | n) #REQUIRED>
<!ELEMENT selector (#PCDATA)>
<!ELEMENT view-refs (view-ref*)>
<!ELEMENT view-ref (#PCDATA)>
```

A schema can refer to several views; each view is qualified by a name ("`name`") and a description ("`descr`"); the element "`selector`" refers to an XSLT style sheet that selects the document constituents displayed in the view; the element "`view-refs`" allows the indication, where appropriate, of references to other views, thus creating a sort of composite view. The attribute "`language-dependent`" indicates whether or not the view includes constituents that are language dependent. The attribute "`toc-edition-enabled`" indicates whether the view allows changes to be made to the table of contents.

```
<!ELEMENT formats(export-format*, diff-format*, transaction-format*)>
<!ELEMENT export-format (name, descr, filename-extension, supported-schema, export-formatter,
import-formatter, resolve-links?)>
<!ATTLIST export-format read-only (y | n) « n »>
<!ELEMENT filename-extension (#PCDATA)>
<!ELEMENT diff-format (name, descr, filename-extension, supported-schema, diff-formatter)>
<!ELEMENT transaction-format (name, descr, filename-extension, supported-schema, export-
formatter, transaction-interpreter)>
<!ELEMENT supported-schema (schema-ref, supported-views)>
<!ELEMENT schema-ref (#PCDATA)>
<!ELEMENT supported-views (supported-view*)>
<!ATTLIST supported-views global-view-supported (y | n) #REQUIRED>
```

```
<!ELEMENT supported-view (#PCDATA)>
<!ELEMENT export-formatter (#PCDATA)>
<!ELEMENT diff-formatter (#PCDATA)>
<!ELEMENT transaction-interpreter (#PCDATA)>
```

The formats for which information is held in the element "`document-repository`" can be of various types: export formats, differential formats or transaction formats. Each format has a name ("`name`") and a description ("`descr`") as well as a filename extension that will be used to name the exported file ("`filename-extension`"); information is also provided on the different schemas supported by the format ("`supported-schema`") and the views supported in the schema ("`supported-views`"). The elements "export-formatter", "`diff-formatter`" and "`transaction-interpreter`" contain information for filter classes fulfilling the corresponding function.

```
<!ELEMENT document-classes (document-class*)>
<!ELEMENT document-class (name, descr, schema-ref, format-refs)>
<!ELEMENT format-refs (format-ref*)>
<!ELEMENT format-ref (#PCDATA)>
```

The document classes (' document-classes ,) for which information is held in the element « `document-repository` » take a name (' name ,), a description (' descr ,) and references to a schema (« `schema-ref` ») and to formats (« `format-refs` »).

The following is a (partial) example of a configuration file:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<document-repository>
<url-base>http://seibud/catalog</url-base>
<filepath-base>/u01/users/seibud/catalog</filepath-base>
<exploit-dir>exploit</exploit-dir>
<schemas>
<schema type="DTD">
<name>abb</name>
<descr>schema for abb publicationscr>
<path>abb.dtd</path>
<toc-entry-locator>com.softwareag.belgium.seibud.abb.AbbTocEntryLocator </toc-entry-locator>
<delta>com.softwareag.belgium.seibud.abb.AbbDeltalta>
<synoptism>abb2syn.xslt</synoptism>
<discard-presentation>abb2nopres.xsltscard-presentation>
<discard-presentation-delta>com.softwareag.belgium.seibud.abb.AbbNoPresDelta
scard-presentation-delta>
<views>
<view language-dependent="y" toc-edition-enabled="n">
<name>data</name>
<descr>Chiffresscr>
<selector>data.xslt</selector>
<delta>com.softwareag.belgium.seibud.abb.AbbFixedStructureDelta lta>
<view-refs>
</view-refs>
</view>
<view language-dependent="y" toc-edition-enabled="n">
<name>text+note+heading</name>
<descr>Commentaires (trad)scr>
<selector>text+note+heading.xslt</selector>
<delta>com.softwareag.belgium.seibud.abb.AbbFixedStructureDeltalta><view-refs>
<view-ref>text</view-ref>
<view-ref>data</view-ref>
<view-ref>heading</view-ref></view-refs>
</view>
</views>
</schema>
<formats>
<export-format>
<name>rtf-text-abb</name>
<descr>Word97 (.rtf)scr>
<filename-extension>rtf</filename-extension>
```

```
<supported-schema>
<schema-ref>abb</schema-ref>
<supported-views global-view-supported="n">
<supported-view>text+note+heading</supported-view>
</supported-views>
</supported-schema>
<export-formatter>com.softwareag.belgium.seibud.abb.Abb2rtf </export-formatter>
<import-formatter>com.softwareag.belgium.seibud.abb.Rtf2abb </import-formatter>
</export-format>
<diff-format>
<name>htm-diff-text-abb</name>
<descr>Internet Explorer (.htm)scr>
<filename-extension>htm</filename-extension>
<supported-schema>
<schema-ref>abb</schema-ref>
<supported-views global-view-supported="n">
<supported-view>data</supported-view>
<supported-view>text</supported-view>
<supported-view>heading</supported-view>
<supported-view>text+note+heading</supported-view>
<supported-view>nmc</supported-view>
</supported-views>
</supported-schema>
<diff-formatter>com.softwareag.belgium.seibud.abb.AbbDiffFormatter ff-formatter>
ff-format>
<transaction-format>
<name>xml-nmc-abb</name><descr>StrTool2 (.xml)scr>
<filename-extension>xml</filename-extension>
<supported-schema>
<schema-ref>abb</schema-ref>
<supported-views global-view-supported="n">
<supported-view>nmc</supported-view>
</supported-views>
</supported-schema>
<export-formatter>com.softwareag.belgium.seibud.abb.Abb2nmc </export-formatter>
<transaction-interpreter>com.softwareag.belgium.seibud.abb.Meta2xupd </transaction-
interpreter>
</transaction-format>
</formats>
<document-classes>
<document-class>
<name>abb</name>
<descr>ABB Documentscr>
<schema-ref>abb</schema-ref>
<format-refs>
<format-ref>htm-text-abb</format-ref>
<format-ref>adp-text-abb</format-ref>
<format-ref>rtf-text-abb</format-ref>
<format-ref>rtf-read-only-text-abb</format-ref>
<format-ref>xml-text-abb</format-ref>
<format-ref>xml-heading-abb</format-ref>
<format-ref>xml-data-abb</format-ref>
<format-ref>xml-pres-global-abb</format-ref>
<format-ref>htm-diff-text-abb</format-ref>
<format-ref>xml-nmc-abb</format-ref>
<format-ref>xml-trn-global-abb</format-ref>
<format-ref>xml-trn-nmc2-abb</format-ref>
</format-refs>
cument-class>
cument-classes>
cument-repository>
```

3.3.3. The XML Repository

The XML Repository subsystem is a system that gives DOM access to persistent and versioned documents. As well as DOM access, documents can be updated using an Xupd transactional interface. The XML Repository subsystem is based on a relational backend (Oracle for SEI-BUD) both for

persistent storage and for the management of locks. This subsystem does not manage transactions on the relational backend; this is left to the system that uses the Repository (the application server JBoss for SEI-BUD).

The different modules of the XML Repository are as follows:

— The**object allocator**("`NoFreeAllocator`")

— The**page allocator**("`NoFreeAllocatorPage`")

— The**cache manager**("`PageCache`" interface)

— The**lock manager**(for example "`OracleLockManager`")

— The**backend connection manager**("`SITDBBackendConnection`" interface)

— The**DOM manager**("com.softwareag.belgium.sitdb.dom" package)

— The**DOM verifier**("`XMLVerifier`")

— The**XML parser**enabling imports ("`PVDOMParser`")

— The**update transaction manager**("`Xupd`")

3.3.4. Client tools

The client tools of the SEI-BUD system are as follows:

— The**control tool**is a Java application that allows communication with the Repository via an HTTP connection. All users have a control tool on their computer and can communicate with the SEI-BUD server at any time. All communications with the SEI-BUD server must pass through it.

— The**structure tool**allows the editorial structure (section, part, title, etc.) and aliases (PDB, DB and B) to be displayed and modified.

— The**figures tool**allows the budget figures for instances managed by the system (ABB publication) to be displayed and modified.

— The**Word-based ABB editing tool**is used to display and modify text in an ABB budget document fragment. It is associated with two types of editorial object: notes and headings

— The**Word-based Volume 0 editing tool**is used to display and modify the 'text' AND structure of a document or document fragment for publication of the type 'Volume 0'.

— The**Adept-based correction tool**allows correctors to make changes to the content of the budget (PDB, DB or B) in all languages.

— The**XMetal**correction tool also allows correctors to make changes to the content of the budget (PDB, DB or B) in all languages.

3.3.5. The Web Interface (Webint) server

*Webint*is the name given to the set of Java servlets that handle all the SEI-AMD user interactions. Jakarta-Tomcat 3.x of the Apache Foundation is the servlet container that runs these servlets.

Webint is mainly used by the European Parlement and by the Council to enter amendments to the (preliminary) draft budget. As a consequence Webint is a web application front-end offering the following functions:

— Creation and edition of**amendments**and**letters of amendments;**

- modify an existing budget line composed of figures and remarks;
- create a new budget line;
- delete a budget line;
- split a budget line into two lines;
- merge two budget lines;

— Translation management
- create special translation files that show exactly which sentences should be translated;
- follow the translation process;
- make some sanity checks on the translated files;

— Creation and management of several**reports;**
- "cahier texte";
- "cahier chiffres";
- vote list;
- CSV export;
- line by line report;
- CAT/POL report (several layouts available);

— Management of**phases**,**groups**and**users**with specific roles and duties
- a phase is attached to a budget publication, to an European institution and possibly to a specific entity in the institution;
- a group defines the tasks that users may execute and commonly map to an entity in an institution;
- there are always separated groups for the authors and the translators

— **Integration**of the amendments into a budget publication

Webint is only the front end. It visualizes the data from the management database without intermediary. All the operations that modify the management data or that are related to the texts of the amendments and to the reports are handled over to the Repository Manager.

3.3.6. The Background Processing Server BPSRV

With BPSRV, the background processing server, business objects can schedule requests to be called on business objects asynchronously after the current requested has terminated.

Tasks are scheduled and stored in the database by using the BpsrvTask business object. The different flavors of the static method BpsrvTask.schedule() will create a new BpsrvTask object in the database which contains all information about the task to execute. This task can be call to any XOo°f method in any business object of RMG. The task is executed in the specified queue and with a specific priority.

Bpsrv uses a self-managed pool of connections to repeatedly read newly inserted tasks from the database and to execute these tasks. These connections are taken at the startup of Bpsrv . The task will be put into a queue and will eventually be executed by entering the handler chain like any other HTTP request sent to the server. Many tasks can thus seamlessly share only a few connections to do this. The pool size is configurable.

3.3.6.1. *Configuration*

Bpsrv is configured via the config file run/conf/bpsrv-config.xml. In this file the different queues can be defined with their specific capacity. Further, basic parameters, like the polling time, the database table, or the connection pool size can be defined, as well as the choice, if failed or passed (or all) tasks should be automatically removed from the database (recommended).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rmg>
 <bpsrv>
 <config
table="${seibud.bpsrv.tablename}"
       delete-failed-tasks="true"
       delete-passed-tasks="true"
       pool-size="5"
       pollDelay="5000"
performance-logging-enabled="true"
 />
 <!-- queue names must be identical to the enumeration BpsrvTask.Queue -->
 <root name="ROOT" capacity="25">
       <queue name="DEFAULT"                  capacity="5"/>
       <queue name="TRIGGERS"                 capacity="2"/>
<queue name="POETRY"                    capacity="5"/>
       <queue name="EURAMIS"             capacity="2"/>
<queue name="REPORTING"          capacity="2"/>
       <queue name="CHECKOUT"            capacity="3"/>
       <queue name="DIFFREPORT_TITLE"    capacity="2"/>
       <queue name="DIFFREPORT_VOL4"     capacity="2"/>
       <queue name="OBJECT_DELETION"     capacity="2"/>
 </root>
 </bpsrv>
</rmg>
```

3.3.6.2. Scheduling

To schedule a task, the static method BpsrvTask.schedule() is used. It will create the appropriate BpsrvTask objects.

The taks can be assigned different priorities. When the tasks are read from the database to be executed, they are read ordered by their priority.

Priority :

LOWEST,

REALLY_LOW,

LOW,

LOWER_THAN_MEDIUM,

MEDIUM,

HIGHER_THAN_MEDIUM,

HIGH,

REALLY_HIGH,

HIGHEST.

It is generally possible to create tasks of different types:

TaskMode :

COMMON_MONITORED_TASK - just one task, monitored by a ProcessingReport

MULTITASK_MASTER_TASK - the first in a sequence of monitored tasks

MULTITASK_CLIENT_TASK - a client task in a sequence of monitored tasks

STEALTH_TASK - a task not monitored by ProcessingReport

Multitasking tasks are a sequence of many tasks to be performed in order to complete a work unit. In order to initiate a multitasking sequence, one must always start with a MULTITASK_MASTER_TASK which will return a processing report id. This id can be used to add further MULTITASK_CLIENT_TASKs to the master task. This can be done even in the client tasks or the first master task in a dynamic way, i.e. without knowing how many client tasks will actually be added when the master task is initiated. Only when the last client transaction finishes, no more tasks can be added.

For all task types but the STEALTH_TASK, a ProcessingReport object is created which allows to follow up the status of the asynchronous tasks. Since the ProcessingReport object will remain in the database, we strongly recommend to let Bpsrv automatically clean up the Bpsrv table.

The application name is an identifier string that shortly describes the operation that is performed. It can be used to retrieve the the processingReportId (of the master BpsrvTaskDAO instance) when adding further client tasks and the processingReportId is unknown to the client task.

An internal scheduler triggers BPSRV to regularly check for newly arrived tasks in the database to be executed. It may also be manually triggered to check for new tasks, which is done whenever a transaction successfully commits.

3.3.6.3. *State Graph*

A task executed by BPSRV changes its state during the course of execution. In the beginning, when it is created and first stored in the database, it is in the state *idle* and waits to be retrieved by BPSRV.

Once BPSRV scans the database for tasks to execute, it loads the task as object into the memory and puts it into a queue. The task is then *scheduled*.

As soon as the scheduler picks the tasks and executes it, it changes its state to *running*.

After the task is finished, it finds itself either in the state *passed* or *failed*, depending on an exception that might have occurred. Tasks that failed, remain in the database in this state until they are manually either removed or changed and rescheduled to be executed again.

Tasks that have passed successfully are automatically set into the state *deletable* by calling the method finalizze() and soon thereafter deleted from the database.

The state graph is shown in the following illustration:

Illustration 5: BpsrvTask State Graph

3.3.6.4. *General Task Properties*

When a task is scheduled with the static method BpsrvTask.schedule(), a new persisted instance of a BpsrvTask object is created. It contains as general properties

- **taskId** – An ID for this BpsrvTask business object (generated automatically)
- **priority** – The priority used to run the task.
- **queue** – The queue to use, defined in the config file.
- **creationTime** – The time the request was created.
- **userId** – The id of the user that initiated the request.
- **errorMessage** – A container for a possible error message which might be captured during execution.

3.3.6.5. *Task Specific Properties*

Apart from the general properties that are stored for a BpsrvTask, there are also the following application specific properties in the BpsrvTask object:

- **applicationName** – An explanatory string describing the current task. For instance "Create a new version for document abc". It is used to find multiple tasks that have been spawned for the same application.
- **verb** – An identifier for the method type. It may be:
  - a class method (value = 1),
  - a method to create a new instance (value = 2), (never used in this context)
  - an instance method (value = 3),

1) **className** – The name of the business class to call. This is always set to "BpsrvTask" since the BpsrvTask implements all methods that are callable asynchronously.

2) **methodName** – The name of the method to call. This is one of the methods mentioned in the description of the BpsrvTask business object on page 27.

3) **instanceId** – The id of the business object to call. This defaults to the taskId (see above) since this BpsrvTask object is specifically created for the execution of just this scheduled method.

4) **requestData** – The parameter to hand over to the method. This string is an unmarshalled XML structure containing all data for the method to do its job.

5) **sessionData** – Information about the current session as base64-encoded string of a serialized SessionData object.

3.3.6.6. *Management Tasks*

The previously described properties characterize the actual task to be executed. Bpsrv offers the possibility to execute one task before the main task, and two tasks after the main task depending on the outcome of the main task: If the main task passed, it may call an okMethod, if the main task failed, it may call a notOkMethod.

This is used in Seibud to create the change the states of the ProcessingReport object to ‘running’ in the beforeTask, and to set the state of the ProcessingReport object to ‘ready-ok’ when the main task (or the complete multitask sequence) passed; or it will set the state of the ProcessingReport object to ‘ready-ko’ if the main task failed.

If one task of a multitask sequence fails, the ProcessinReport object will immediately go into the state ‘running-ko’ and thus already indicate to the user the erroneous state.

In case of error, the exception stack trace will be retrieved from the Bpsrv motor and will be added as AttachedResult to the ProcessingReport.

3.3.7. The Scheduler Demon

A generic scheduler demon is running in Seibud, similar to a Unix cron job. It allows to schedule periodical tasks to be executed either in periodic intervals or at periodic points in time. The tasks can be Xooof methods in the server. Currently the implementation is limited to static class methods which can be called (no instance methods).

The scheduler is configured with the config file run/conf/scheduler-config.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rmg>
 <!-- pollDelay in milli seconds indicates sleep time of scheduler (minimum event distance) --
>
       <scheduler pollDelay="10000">
       <event       class="TranslationServicePoetryUserDemand"
              method="handleNonBlockingTranslations"
              userId="poetry"
              queue="POETRY"
              priority="HIGH">
              <periodic hours="0" minutes="1"/>
       </event>
<event       class="EuramisAlignRequest"
              method="scanWoodResultsDirectory"
              userId="poetry"
              queue="EURAMIS"
              priority="HIGH">
              <periodic hours="0" minutes="1"/>
</event>
       </scheduler>
</rmg>
```

3.3.8. The Trigger Handler

A trigger handler is installed in the chain of handlers, which will trigger a certain event (i.e. call to a Xooof method) when a trigger event occurrs (i.e. a specific Xooof method has been called).

This allows to react on specific user actions and do further processing if e.g. certain conditions are met. Multiple methods can be called upon one trigger. The triggered methods are executed completely asynchronously via BPSRV in the queue TRIGGERS.

The trigger handler is configured with the config file run/conf/triggers.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rmg>
       <triggers>
        <!-- the triggers listed in the event will trigger the execution of the event
class.method()
                     Currently only static (class) methods are allowed as event targets and no
rqst struct is supported
```

```
                Example:
            <event class="TheClassToCall" method="theMethodToCall">
                  <trigger class="TheTriggerClass1" method="theTriggerMethod1"/>
                  <trigger class="TheTriggerClass1" method="theTriggerMethod2"/>
                  <trigger class="TheTriggerClass1" method="theTriggerMethod3"/>
                  <trigger class="TheTriggerClass2" method="theTriggerMethoda"/>
                  <trigger class="TheTriggerClass2" method="theTriggerMethodb"/>
            </event>
        -->
        </triggers>
</rmg>
```

4. THE TRANSACTIONAL XML REPOSITORY SUBSYSTEM

The XML Repository subsystem is a system that gives DOM access to persistent and versioned documents. As well as DOM access, documents can be updated using an Xupd transactional interface. The XML Repository subsystem is based on a relational backend (Oracle for SEI-BUD) both for persistent storage and for the management of locks. This subsystem does not manage transactions on the relational backend; this is left to the system that uses the Repository (the application server JBoss for SEI-BUD).

4.1. Persistent DOM

The XML Repository used in the SEI-BUD project is a component developed and used by SAGB for the projects it is involved with. This XML Repository is based on the implementation in Java of a versioned persistent DOM (Document Object Model). This repository has the following principle characteristics:

— It offers a 'DOM Level 1 Core' Java programming interface (it successfully passes the W3C Doxygen test suite).

— It is only available in the form of a Java library and is intended to be integrated in a wider system: its target use is actually integration in an EJB (Enterprise JavaBeans) application server.

— It supports storage and access for multiple documents in one or more databases that can be located on different machines.

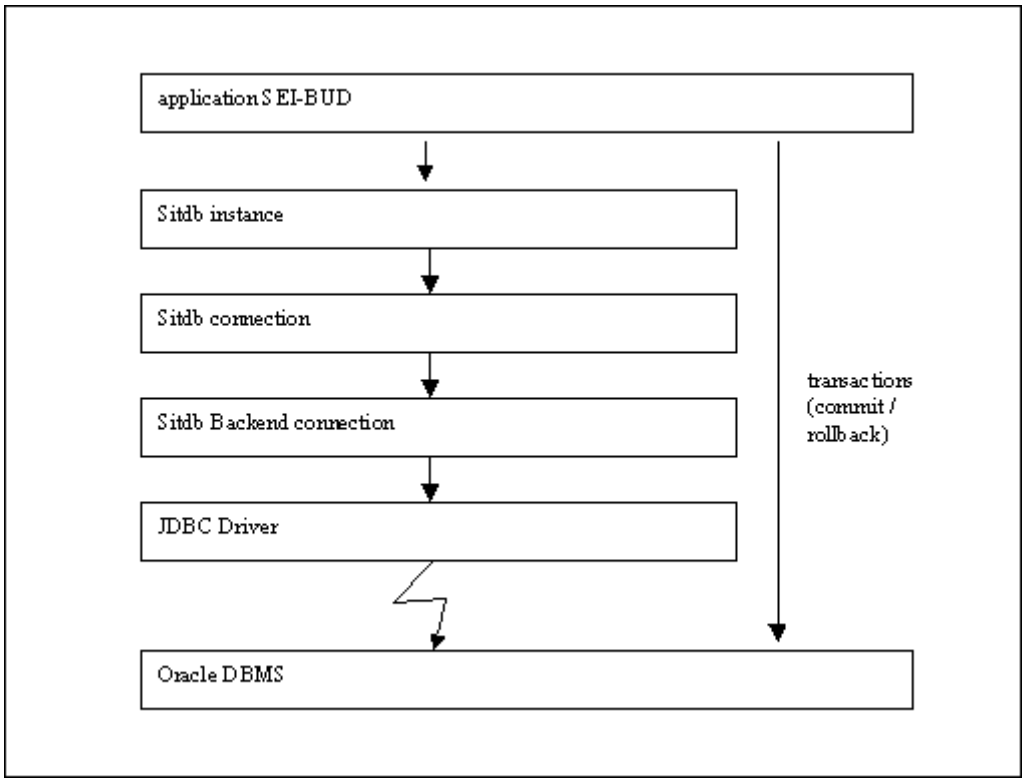
— In a single database, each document is identified by a unique name.

— It is a versioned repository, i.e. every change to a DOM node increments its version number. An identifier can be allocated to a particular version of a document. It is also possible to manipulate branches and a facility exists for branch mergers ("merge").

— It requires the support of a backend to maintain and manage locks. An Oracle backend is used in the SEI-BUD system.

4.1.1. General architecture of the persistent DOM

The architecture of the persistent DOM is broadly illustrated by the diagram below (ideally it should be shown using a class diagram, but that type of diagram is outside the scope of this document):



As mentioned above, the repository requires the support of a backend database permitting the storage of the pages containing DOM objects. These pages are considered by the backend to be 'blobs' (Binary Large Objects). The repository supports a memory backend for which there is no persistence. This memory backend is used essentially to simplify operations when system components are developed. For the SEI-BUD project, the repository uses an**Oracle**backend, which uses Oracle "**LONG RAW**" fields.

Because the repository is written entirely in Java, a**JDBC driver**(Java Database Connectivity) is used to access the database management system. The use of this JDBC interface therefore enables a certain amount of independence in the code with respect to the backend.

The repository does not manage**transactions**: it is normally used as a library in an EJB application server; management of the database connection and transactions is taken up to the level of the EJB component(s) using the repository or even to the level of the EJB container.

The component **SitdbBackendConnection** is responsible for implementing an interface for database access on the backend; this includes not only access to pages (creation, modification, deletion of pages) but also the management of locks.

The component **SitdbConnection** uses the services of the component SitdbBackendConnection. It is independent of any backend. It manages instances (see below) by allowing the creation, deletion and location of an instance. This component also manages a page buffer ('cache') system. Each SitdbConnection actually has two types of page cache: read-only and read/write. A page cache is in fact a construction that allows pages to be saved using a link between a page identifier and its content. This means access to pages can be optimised without having to interrogate the backend. (Please note that the performance of the repository within SEI-BUD could be significantly improved by the development of a page cache capable of integrating the pages from different simultaneous requests into a single entity; this, however, would require access to transaction start and end events).

The component **SitdbInstance** provides services for accessing an XML document. The main methods for this component are:

— the obtaining of a DOM document;

— the selection of a version or branch of navigation in the DOM document;

— the ability to increment the version of the current instance;

— the creation of a new branch;

— the closure of an instance;

— the possibility of obtaining correspondence between two nodes in different versions (where these nodes actually exist in each of the versions). This facility is used intensively in the update tier Xupd.

It is also this component that determines the way page content is managed, i.e. what basic components are used to allocate and manage DOM objects.

4.1.2. Organisation of DOM objects in pages

This section contains a fairly low-level description to give an idea of how DOM objects are organised.

Each SitdbInstance component refers to two page allocators: one allocator for the XML text (whether this is #pcdata or attribute values) and one allocator for the structure (DOM nodes). Two allocators are used for better 'localisation' when navigating a tree.

DOM objects are allocated sequentially, by filling the pages. SEI-BUD implementation uses pages of 256 objects. Note that gaps are not normally created in a page because all the versions are systematically saved and are never deleted. Objects (e.g. DOM nodes) never contain references to other objects: links are created using 'handles'.

A handle consists of three pieces of information:

— the position of the object in a vector of allocated objects in a page ('slot' number)

— a page identifier that is unique within the document

— a unique document identifier (page set) in the database associated with the current backend.

The**basic postulate** on which the entire organisation and concurrent use of the repository is based is that once allocated and validated ('committed') within the database, a **DOM object will always be accessible and valid through its handle**. Thus, if a read request starts to load pages in its cache in order to list the sub-elements of a particular element, it could happen that a write request then modifies certain pages in the same document, and in particular, the number of sub-elements in question; once the modification request has been validated (committed) in the backend, the read request can continue and any modified pages can be reloaded without any problem.

A document is therefore a collection of pages and is portable from one backend system to another, as long as there is no conflict in the page set identifiers (an import function prevents this problem from occurring by modifying each handle in a document using a new page set identifier).

A page can be in a number of states:

— resident in the memory (in a cache), with all objects available (deserialised);

— resident in the memory (still in a cache), with all objects serialised;

— resident only in the database management system used (either on disk or in a cache area).

4.1.3. DOM interface

The XML Repository offers a 'DOM Level 1 Core' interface. Support for this standard interface enables the use of other components, such as an XPath or XSLT processor. The update component ("Xupd") built on the XML Repository in fact uses an XPath processor (in the SEI-BUD project, it is the XPath processor that is included in the XSLT Xalan processor).

4.1.4. XML parser

The XML repository also uses the DOMParser interface as defined in the Xerces (Apache) implementation, to give it a component capable of importing XML documents.

4.1.5. XML validation

The repository offers a validation function based on Sun Multi-Schema XML Validator technology (). However, the repository does not automatically perform any validation with respect to a schema or DTD; validation must be initiated by a user component (such as the update tier or after using the import function).

4.1.6. Management of locks

The repository settles document access conflicts using locks; to do this, it uses the facilities provided by the corresponding backend database: in the case of SEI-BUD, this is Oracle.

Locks are managed within the SitdbConnection component. When an instance is created, deleted or called up, the component SitdbbackendConnection is used to obtain a lock on this instance. This lock can be read-only or write. Similarly, it is possible to wait and see whether or not a lock is successfully obtained. Obtaining a lock returns an object that is preserved at the level of the instance and will be used to release the lock when that instance is closed. It should be noted with regard to this that the lock is not really released until the end of the current transaction (commit/rollback).

Locks are managed by a dedicated component linked to the backend being used. In the case of Oracle, lock management uses the package "sys.dbms_lock".

4.1.7. Management of versions and branches

The version management uses an instance version number. This number is incremented every time the instance is modified. It should be borne in mind that only the last version of an instance branch can be modified.

Each node of a document has a list of version ranges. A version range has a name and two version numbers: a first version number and a last version number. The last version number is in fact the first version number the node of which no longer belongs to the instance corresponding to that version.

From a DOM point of view, the nodes of each version are all linked: all the daughter nodes of all the versions are linked as sisters; the navigation determines whether a particular node is valid in the current navigation version.

Branches are divergent modifications of the same node. One or more new branches can be created at all levels of a DOM tree. A branch has a name and a range of version numbers. A merge function is

available; this detects and reports potential conflicts between the current navigation branch and a named branch.

4.1.8. Description of the tables used

The following tables are created by the SQL script "c_oracle_longraw.sql" (see the Installation Guide (GIL)):

4.1.8.1. The SEIBUD_PAGESETS table

This table gives the information relating to a document in a particular language version. When a document is opened, the "`pagesetid`" value is found for a particular document name ("`pagesetname`"). The field "`lastpageid`" is used to create and assign new page identifier values.

Name	Type	Description
pagesetid	number	unique document identifier (primary key)
lastpageid	number	last page identifier used for the document
pagesetname	varchar	document name (primary key)

4.1.8.2. The SEIBUD_PAGES table

This table provides access to the different pages that make up a document using two values as a key: the document identifier ("`pagesetid`") and a page identifier.

Name	Type	Description
pagesetid	number	unique document identifier (part of key)
pageid	number	unique page identifier within the document (part of key)
pagelen	number	length of page in bytes
pagedata	long raw	page data

4.2. Xupd interface

The XML repository used for SEI-BUD offers not only an update interface that conforms to the DOM API but also a more high-level interface specifying transactions and inspired by Xupdate ().

An example of an Xupd transaction file generator is the difference calculation tool, which receives two XML documents and produces an XML document representing the transactions necessary to obtain the second document from the first.

4.2.1. Definition of the Xupd format

N.B.: this DTD is a pseudo-DTD in that some operations may include tags corresponding to another DTD (insertion operations).

```
<!ELEMENT modifications
(insert-before|insert-after|append|update|remove|update-atts
|insert-atts|test)*>
<!ATTLIST modifications
version NMTOKEN #REQUIRED
xmlns CDATA #FIXED ""
default-select-version CDATA "0"
default-xpath-prefix CDATA #IMPLIED
default-xpath-resolver CDATA #IMPLIED>
```

An Xupd document consists of a series (which may be empty) of transactions or operations. These different operations will be described in detail below.

The attribute**version**relates to the version of the Xupd format (currently version 1.0).

The attribute **xmlns** determines an XML namespace.

The attribute **default-select-version** indicates the default selection version for the instance to which the transactions are being applied. A value of zero (the default value) specifies the current version. A negative value specifies a previous version to the current one; thus the value "-1" refers to the document version before the application of the first transaction. A positive value specifies an absolute version number.

The attribute **default-xpath-prefix** specifies a prefix that should complete all the selections specified in the transactions. It is generally used to identify the XPath of the root element of the fragment to which the transactions relate.

The attribute **default-xpath-resolver** allows an alternative XPath processor to be specified. The default XPath processor used is Xalan. An alternative giving better performance, particularly when selections relate to the current version, avoids re-indexation by Xalan by using a very rudimentary processor ("simpleXpathResolver"). This processor is capable of interpreting the XPaths generated by the difference calculation module, broadly-speaking the absolute XPaths.

4.2.1.1. insert-before

```
<!ELEMENT insert-before (other-markup|value-of)*>
<!ATTLIST insert-before
select CDATA #REQUIRED
select-version CDATA #IMPLIED
reIndexSource CDATA "1">
```

This operation allows a node to be inserted just before the specified node. The specified node cannot be an attribute node. If the selection (see attribute**select**) is targeting several nodes, the insertion is made before each of the nodes.

The element **other-markup** is a 'dummy' element that represents any other tag in a namespace other than the Xupd namespace.

The attribute **select** must contain an XPath expression that may be prefixed by a part of XPath relating to the fragment to which the transactions apply, and must be given a value as a 'node-set' that does not contain any attribute nodes.

The attribute **select-version** has the same semantics as the attribute **default-select-version** of the element **modifications** and takes precedence over this value by default.

If the attribute **reIndexSource** has the value "0", this shows that it is not necessary to re-index the target document before applying this transaction (by default, re-indexation takes place every time). This attribute is only relevant when the version used for selections is the current version; if it is the previous version, it does not change again and therefore only has to be indexed once.

4.2.1.2. insert-after

```
<!ELEMENT insert-after (other-markup|value-of)*>
<!ATTLIST insert-after
select CDATA #REQUIRED
select-version CDATA #IMPLIED
reIndexSource CDATA "1">
```

This operation is similar to the operation **insert-before**.

4.2.1.3. append

```
<!ELEMENT append (other-markup|value-of)*>
<!ATTLIST append
select CDATA #REQUIRED
select-version CDATA #IMPLIED
reIndexSource CDATA "1">
```

This operation is also like**insert-before**and allows nodes to be added like additional children.

4.2.1.4. update

```
<!ELEMENT update (other-markup|value-of)*>
<!ATTLIST update
select CDATA #REQUIRED
select-version CDATA #IMPLIED
reIndexSource CDATA "1">
```

This operation is like **insert-before**.

4.2.1.5. remove

```
<!ELEMENT remove EMPTY>
<!ATTLIST remove
select CDATA #REQUIRED
select-version CDATA #IMPLIED
ignore-empty (yes|no) no
reIndexSource CDATA "1">
```

This operation is used to remove a node. Unlike the other operations, the selection can be resolved as attribute nodes.

The attribute **ignore-empty** indicates that the error should not be produced if the selection is empty, that is, if the XPath expression does not define a node.

4.2.1.6. insert-atts

```
<!ELEMENT insert-atts (other-markup|attribute+)>
<!ATTLIST insert-atts
select CDATA #REQUIRED
select-version CDATA #IMPLIED
reIndexSource CDATA "1">
```

This operation allows one or more attributes to be added. The selection targets element nodes.

There are two ways of specifying attributes: either by using the sub-element "attribute", or by using an empty 'dummy' element the attributes of which will be used (see examples of use).

4.2.1.7. update-atts

```
<!ELEMENT update-atts (other-markup|attribute+)>
<!ATTLIST update-atts
select CDATA #REQUIRED
select-version CDATA #IMPLIED
reIndexSource CDATA "1">
```

This operation is like the operation **insert-atts**. An attribute that does not exist is inserted and an attribute that already exists will have its value modified.

4.2.1.8. attribute

```
<!ELEMENT attribute EMPTY>
<!ATTLIST attribute
name NMTOKEN #REQUIRED
namespace CDATA #IMPLIED
```

```
value CDATA #REQUIRED>
```

This construction is used to specify an attribute and its value.

The attribute **name** specifies the name of the attribute to be inserted or modified.

The attribute **namespace** allows its XML namespace to be indicated if necessary.

The attribute **value** specifies the value of the attribute.

4.2.1.9. value-of

```
<!ELEMENT value-of EMPTY>
<!ATTLIST value-of
select CDATA #REQUIRED
select-version CDATA #IMPLIED
ignore-empty (yes|no) no
reIndexSource CDATA "1">
```

This construction enables reference to be made to one or more nodes of the modified document. It is very useful in insertion operations because it allows part of the document to be copied to the point of insertion.

4.2.1.10. test, if, else

```
<!ELEMENT test (if, else?)>
<!ELEMENT if (insert-before|insert-after|append|update|remove|
update-atts|insert-atts|)>
<!ELEMENT else (insert-before|insert-after|append|update|remove|
update-atts|insert-atts|)>
<!ATTLIST test
select CDATA #REQUIRED
select-version CDATA #IMPLIED
reIndexSource CDATA "1">
```

This construction allows conditional updates to be performed. This is useful when the application generating the transactions does not have access to the full document.

The operations specified by the construction "if" are executed when the selection specified by the element "test" is not empty. If this selection is empty, it is the operations specified by the construction "else" that are executed.

4.2.2. Examples of the use of Xupd

Only the first example mentions the root element "modifications" for reasons of conciseness.

The target of all the examples is the following XML document:

```
<addresses>
<address id="1">
<name>
<first>John</first>
<last>Smith</last>
</name>
<city>Houston</city>
<state>Texas</state>
<country>United States</country>
<phone type="home">333-300-0300</phone>
<phone type="work">333-500-9080</phone>
<note>This is a new user</note>
</address>
</addresses>
```

4.2.2.1. Insert Element Before

Add a middle name element before the last name element

```
<xupd:modifications version= »1.0 »
xmlns:xupd= » »>
<xupd:insert-before select="/addresses/address[@id = 1]/name/last" >
<middle>Lennox</middle>
</xupd:insert-before>
</xupd:modifications>
```

4.2.2.2. Insert Element After

Add a cell phone element after the home phone element

```
<xupd:insert-after select="/addresses/address[@id = 1]/phone[@type='home']" >
<phone type="cell">490-494-4904</phone>
</xupd:insert-after>
```

4.2.2.3. Append Element

Append a zip code element to the address record.

```
<xupd:append select="/addresses/address[@id = 1]" >
<zip>90200</zip>
</xupd:append>
```

4.2.2.4. Insert attribute

Add an extension attribute to the work phone element.

```
<xupd:insert-atts select="/addresses/address[@id = 1]/phone" >
<xupd:attribute name="extension" value="223"/>
</xupd:insert-atts>
```

4.2.2.5. Insert Text Content

Append the "of America" clause to the country element. Note that leading white space characters are significant in this case.

```
<xupd:append select="/addresses/address[@id = 1]/country" > of America</xupd:append>
```

4.2.2.6. Insert XML Block

Add a new address record to the top level addresses element.

```
<xupd:append select="/addresses" >
<address id="2">
<name>
<first>Susan</first>
<last>Long</last>
</name>
<city>Tucson</city>
<state>Arizona</state>
<country>United States</country>
<phone type="home">430-304-3040</phone>
</address>
</xupd:append>
```

4.2.2.7. Update Element

Change the first name of address with id = 1 to be Johnathan.

```
<xupd:update select="/addresses/address[@id = 1]/name/first/text()">Johnathan</xupd:update>
Update Attribute
Change the type of the phone number 333-300-0300 to be a cell.
```

```
<xupd:update-atts select="/addresses/address[@id = 1]/phone[.='333-300-0300'] "
><xupd:attribute name="type" value="cell"/>
</xupd:update-atts>
or
<xupd:update-atts select="/addresses/address[@id = 1]/phone[.='333-300-0300'] " ><dummy
type="cell"/>
</xupd:update-atts>
```

4.2.2.8. Delete Element

Remove all phone elements.

```
<xupd:remove select="/addresses/address[@id = 1]/phone"/>
```

4.2.2.9. Delete Attribute

Delete all type attributes on phone elements.

```
<xupd:remove select="/addresses/address[@id = 1]/phone/@type" />
```

4.2.2.10. Delete Text Content of an Element

Clear the content of the country element.

```
<xupd:remove select="/addresses/address[@id = 1]/country/text()" />
```

4.2.3. Operation of the Xupd module

The module Xupd could be considered a 'front-end' tier for applying transactions to the XML repository. This module uses an XML parser, an XML verifier, one or more XPath processors and the persistent DOM API provided by the repository.

Its basic function is a transaction application function that accepts the following parameters:

— a backend connection (SITDBBackendConnection);

— the name of an instance to which the transactions are to be applied;

— a branch name;

— a transaction file;

— an XPath prefix relating to the document fragment on which the transactions are to be applied (the transactions can actually have been calculated from two corresponding fragments, without knowing their context);

— a URL referring to a schema or a DTD that will be used to validate the modifications once all the transactions have been applied.

Where the transactions have been generated by the difference calculation module, the transactions are expressed in relation to the version of the document before modification; two instances are then opened: a read-only instance that will be used to resolve the selections, and a write instance.

Transactions are interpreted 'on the fly' by a SAX parser.

Depending on the operating mode, the selection of the target for the transactions will be handled either by a rudimentary but effective processor included in the Xupd module, or by the Xalan XPath processor. It should be noted that the use of the Xalan XPath processor theoretically requires the document to be re-indexed completely at each transaction. Xalan creates its own indexation structures but does not have a mechanism for incremental indexation (for this, it would have to use a modification notification system).

Each XPath selection can produce a collection of nodes. If the nodes refer to the version of the document before modification, a specific facility of the repository is used that allows you to obtain a reference to the corresponding node in the current version; it is to this corresponding target node that the modification will be made.

Once all the transactions have been applied, the document fragment to which they have been applied is validated with respect to either a DTD or a schema. This validation is based on the SUN Multi-Schema XML Validator library ().

4.3. The difference calculation tool (xdiff)

4.3.1. Introduction

Before going into detail on the operation of the difference calculation tool, it is important to remember a few key aspects of a system like SEI-BUD.

One of the strengths of the SEI-BUD system is that it offers a 'check-out/check-in' interface similar to those found in version control systems such as CVS. The user makes changes to a reserved document and validates these changes in the repository. The system calculates the differences before and after modification and expresses them in the form of transactions, partly to minimise the volume of changes in the database and partly to produce transactions that can also be used to modify the other languages.

Being able to produce transactions for modifications that are valid for all the languages is a key feature of the SEI-BUD system.

It should be noted that when new material is added by an author in a given language, this is inserted in all the other languages. This method of working provides translators with a strong incentive to use translation memory based tools. The difference calculation tool does not try to generate update operations on the basis of 'fuzzy' algorithms (percentages of similar words, etc.). This type of algorithm is left to dedicated TM-based tools (such as Trados Translator's Workbench).

4.3.2. Use

The tool **xdiff** is a batch tool that works on XML files. It can work in several different modes; these modes are controlled by options. The **xdiff** activation syntax is as follows:

```
xdiff old_file new_file xupd_file
[-t] [-m marked_file] [-w word_level_marked_file]
[-p parameter_file]
```

(The parameters in square brackets are optional).

— **old_file** is the XML document before modification

— **new_file** is the XML document after modification

— **xupd_file** is the transaction file to be created

— **-t**specifies that transactions must be calculated at the 'mixed content' level (see below)

— **-m marked_file** allows a marked file to be created

— **-w word_level_marked_file** allows a file to be created in which the differences are marked at word level. This option is incompatible with the generation of a transaction file.

— **-p parameter_file** specifies an XML file containing additional operating parameters (see below)

4.3.2.1. Transaction granularity and 'mixed-content'

By default, xdiff automatically detects the 'mixed-content' elements, i.e. elements that can contain both character data and sub-elements. Elements appearing in mixed content are not generally linguistically synoptical; that is why transactions are calculated using a granularity that does not pick up mixed content elements, or to put it another way, why the XPaths generated never refer to these elements. Where only a single language is being used, xdiff can calculate differences using a finer granularity by specifying the option "-t".

4.3.2.2. Documents marked to show differences

Xdiff can generate a marked file in which the transactions are tagged. Two short examples are shown here:

```
<doc><P><diff diff="ins">Pierre Jeanff><diff diff="del">Pierre Paul Jeanff></P>c>
<doc><P diff="del"><diff diff="del">p1ff></P><P>p2</P><P>p3</P>c>
```

In practice, the difference is marked by adding the attribute "diff" to the modified elements; this attribute can have the values "ins", "del" or "copy". In addition, the portions of text affected are encompassed by a "diff" tag with the same attribute "diff". This marked format is useful for presenting differences or generating a "track-change" file.

Xdiff can also generate documents marked to word level or word group level. This marked format is useful for generating a file of the type "track-change". Here is a short example: <doc><P>Pierre <diff diff="ins">Paul ff>Jean</P>c>

4.3.2.3. Parameters file

Xdiff was designed to make the calculation of differences as simple as possible, avoiding the multiplication of parameters. Nevertheless, there is an operating mode that still requires the parameters for the type of document compared to be specified: this is the operating mode associated with a fixed structure.

In collaborative publishing projects such as SEI-BUD, it is necessary to prevent certain parts of the document from being modified – the table of contents or the linguistically synoptic structure of the document, for example.

This is why the names of elements belonging in the structure of the document can be specified easily. The difference calculation programme starts by aligning the two documents using each root as a starting point and linking the elements on each side that are named in the list.

Example of a parameters file:

```
<xdiff_params>
<match name="abb"/>
<match name="nmc-section"/>
<match name="nmc-sectpart"/>
<match name="nmc-expenditure"/>
<match name="nmc-revenue"/>
<match name="nmc-title"/>
</xdiff_params>
```

4.3.3. Operation

The difference calculation tool is written in C++ and optimised for speed and to use a minimal amount of memory. Comparing two versions of a document of several megabytes such as volume 4 of SEI-BUD and producing a transaction file of several megabytes takes just a few seconds.

The programme's operation can be broken down into the following stages:

— parsing of the two documents, determination of the 'mixed content' elements and creation of a data structure that suits the algorithm used;

— alignment of the two documents;

— generation of transactions on the basis of the aligned documents;

— creation of one or more results documents.

The core of the operation is obviously the alignment. This is based on a principle known about for many years and documented, particularly in the 'Literate Programming' column in 'Communications of the ACM (CACM)', June 1989 issue (32, 6, 740-755) by D.C. Lindsay.

The principle is based on the detection of unique strings and the opening of windows or the enlargement of corresponding areas around these unique alignment points. This principle, applied originally to sequences of lines, has been modified so that it can be applied to a tree structure, particularly by using the notions of absolute XPath and relative XPath.

With regard to the generation of transactions, it should be pointed out that partly because the output format of the transactions is relatively poor (no 'surround' or 'join', etc.), and partly because the transactions are expressed sequentially from a pivot, which is the largest portion aligned, it is necessary to 'undo' part of a completed alignment.

4.3.4. Transaction format

See the section "Xupd Interface"/"Definition of the Xupd format".

5. THE REPOSITORY MANAGER AND COMMUNICATIONS SUB-SYSTEM

5.1. The principle governing communication between the server and the clients

Within the context of the SEI-BUD system, communication between clients and server – an EJB server, as it happens – is based on a number of 'best practices' and 'design patterns':

— multi-tier architecture: a client never accesses a database directly but uses services provided by a third party that implements the business logic

— the information exchanged between clients and servers is represented in the form of XML messages; these messages can be analysed for syntax, modelled by a schema or DTD and validated using standard tools, independently of any programming language

— by using the EJB framework, the specification of client/server interfaces is expressed in a form that meets an industrial standard

— the use of a 'facade session' pattern allows, in particular, more flexibility in the development of business components

— the use of the HTTP protocol (although this is not a constraint) gives greater ease of administration within the framework of the networks of the European institutions.

In the SEI-BUD project, the application of most of these principles has been an automatic consequence of using the XOo f framework, a tool of Software AG. The complete documentation for the XOo f framework lies outside the scope of this manual: the specific use of the framework will be described in relation to the client/server communication diagram.

5.1.1. Basic client/server communication diagram

The way communication works between clients and servers in the SEI-BUD system can be broadly summed up as follows:

A client request can be broken down into the following stages:

— The client creates an XMLDispatcher object; in this case, since the protocol being used is HTTP, the client creates an 'XMLDispatcherHTTPFormClient', which is a specific form of the normal client for the HTTP protocol. Then the client uses the method "dispatchInstanceMethod" or "dispatchClassMethod", depending on whether it is a class or instance method. In both cases, the parameters given are a class name corresponding to the type of business object, a method name, and a parameter, which is an XOo f structure (see further on). In the case of an instance method, an additional parameter is needed; this is the reference to the instance of the business object.

— The XMLDispatcher translates the parameters of the request into XML (it serialises the parameter of the request and posts this request to the web server using the HTTP protocol).

— The HTTP request is redirected by the web server to a dedicated servlet. This servlet (RMGServlet) unpacks the HTTP message and sends it to the EJB server using the RMI protocol.

— In the EJB server, the request stops at the XMLDispatcherServer, which is implemented as a 'session bean' using the design pattern 'facade session'. Among other advantages, using this pattern means that coupling between clients and business beans can be reduced. It also means that business beans can be implemented like local beans, giving better performance.

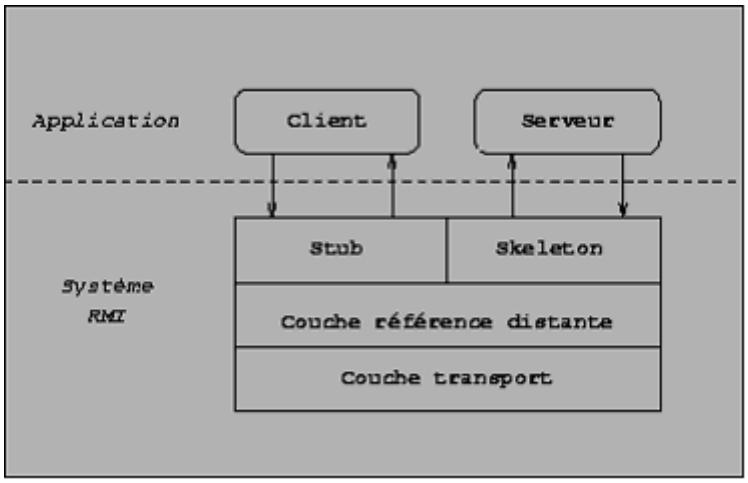
— Depending on the case, the request is then either transformed or passed directly to an entity bean; what is done depends on whether it is a class method or an instance method.

The steps are carried out in reverse to give the return parameter.

5.1.1.1. Background on the RMI protocol

The RMI system contains three tiers: the stub/skeleton tier, the remote reference tier and the transport tier. Each one is independent of the others and uses a specific protocol; for example, the transport protocol used can be either TCP or UDP.

Two techniques are used to transmit objects: serialisation and dynamic loading of the stub, which enables the client to load the stub dynamically when it only has the interface.



Key:

Serveur = Server

Système RMI = RMI system

Couche référence distante = Remote reference tier

Couche transport = Transport tier

The link between the three tiers is shown in this figure.

The stub (used by the client) is an implementation of the remote interfaces and its purpose is to invoke the remote object (by calling the remote reference tier), pack the arguments of the method in a stream, inform the remote reference tier, wait for the return value, unpack the return value or the exception if there is one, and inform the remote reference tier that the invocation is finished.

The skeleton (server side) contains a method that invokes the methods of the remote object. Its purpose is to unpack the arguments of the call, invoke the method concerned, and pack the return value or exception, if there is one.

The remote reference tier manages the type of RMI (point-to-point, multi-cast invocation, etc.).

The transport tier deals with the actual communication. It handles network connections, listens to calls, manages the table of recorded objects and locates the server during a call.

5.1.2. Diagram of classes of an entity bean with the XOo f framework

The diagram below shows the different relationships that exist between the classes generated by the XOo f framework, the XMLDispatcher and the EJB infrastructure. Note that the XOo f framework allows integration either by 'subclassing' or by delegation; the latter method is used for the SEI-BUD system.

In this diagram, the five types of classes are shown in different colours (the character string "MyBO" corresponds to a particular bean name):

— The **EntityBean** interface, which is part of the J2EE framework

— the classes that are part of the XOo f framework:

— the class **BaseBeanDelegateImpl**, which implements the EntityBean interface

— the interface XMLDispatcherState

— the classes or interfaces generated by the framework XOo f:

— **MyBOHome** is the local "Home" interface and extends the interface EJBLocalHome, which is part of the J2EE framework

— **MyBO** is an interface that extends the EJBLocalObject interface

— **MyBOBean**is a class that inherits from the class BaseBeanDelegateImpl, which is part of the XOo f framework. It is this class that handles the delegation of requests to the class MtBoImpl programmed by the developer.

— the classes to be programmed by the user of the XOo f framework:

— **MyBoImpl** is the class to be developed by the bean programmer; this class inherits from/extends the class BaseBO

— **BaseBO** is a class that implements the EntityBean interface; this class provides additional flexibility on the one hand by introducing a degree of indirection vis-à-vis the EntityBean class and on the other by offering the possibility of grouping behaviour common to several beans within it. In the SEI-BUD system, this class is used in particular for tracing and for managing the context of the entity bean.

the classes generated by the EJB container when a bean is initialised

5.1.3. Sequence diagram for a typical synchronous request

To illustrate the interactions between the different components involved in client/server communication, here is a macroscopic sequence diagram for a synchronous request such as a table of contents request.

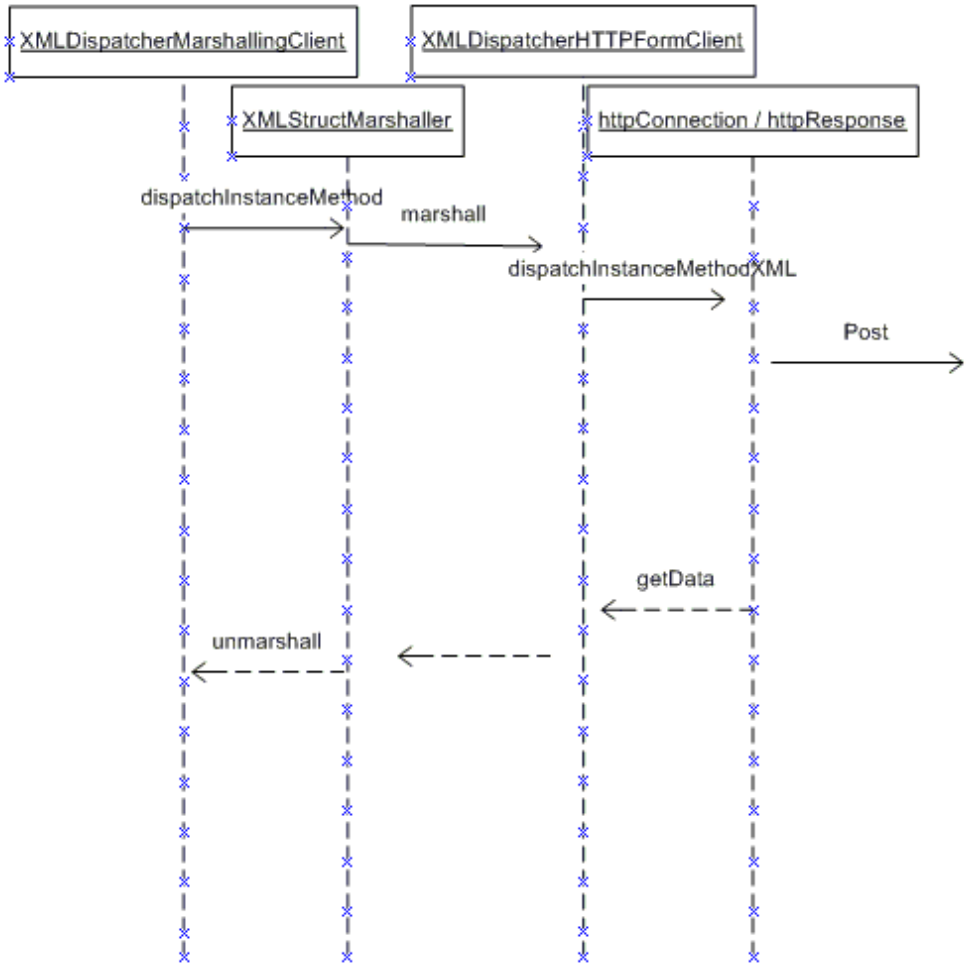
This sequence diagram can be broken down into three parts:

— a client part showing what happens in the XOo f tier with the XMLDispatcher;

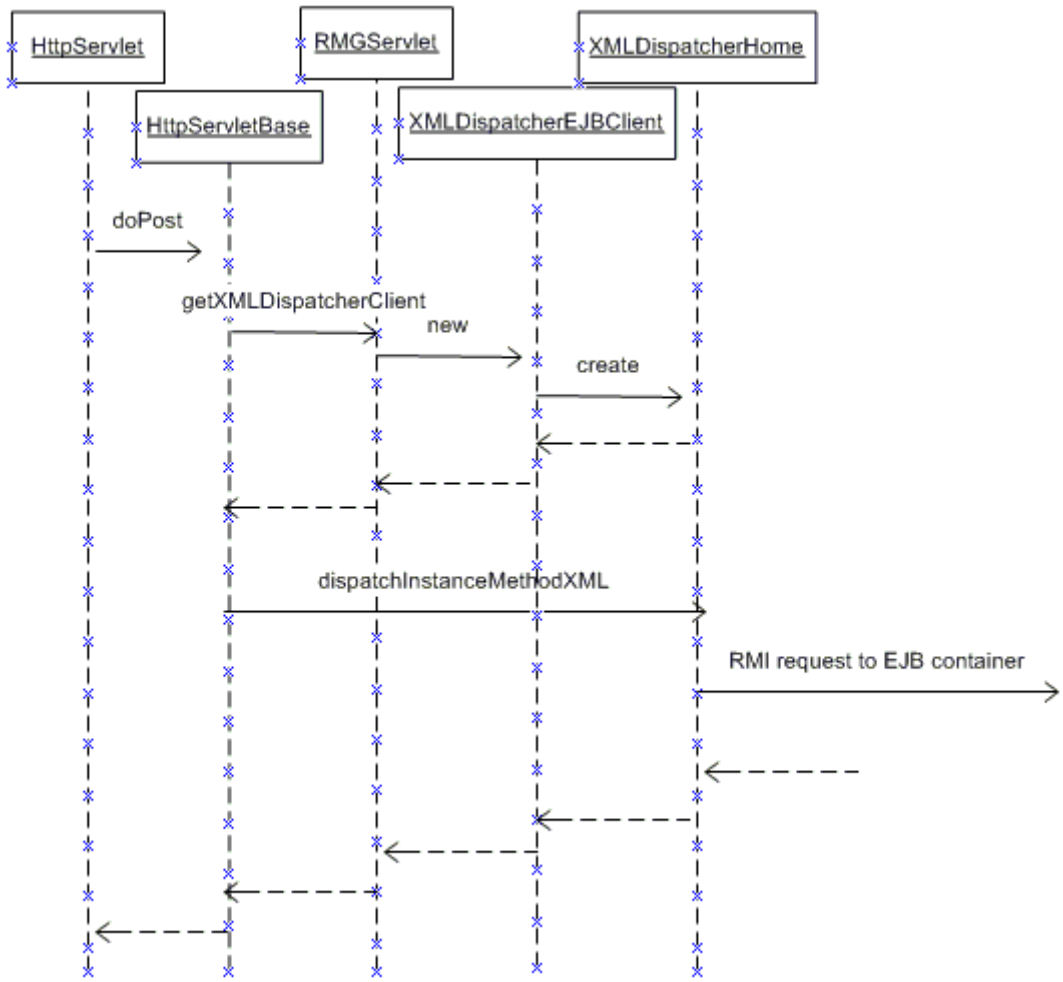
— a servlet part located in the web server (which happens to be Tomcat);

— an EJB part located in the EJB server (JBoss in this case).

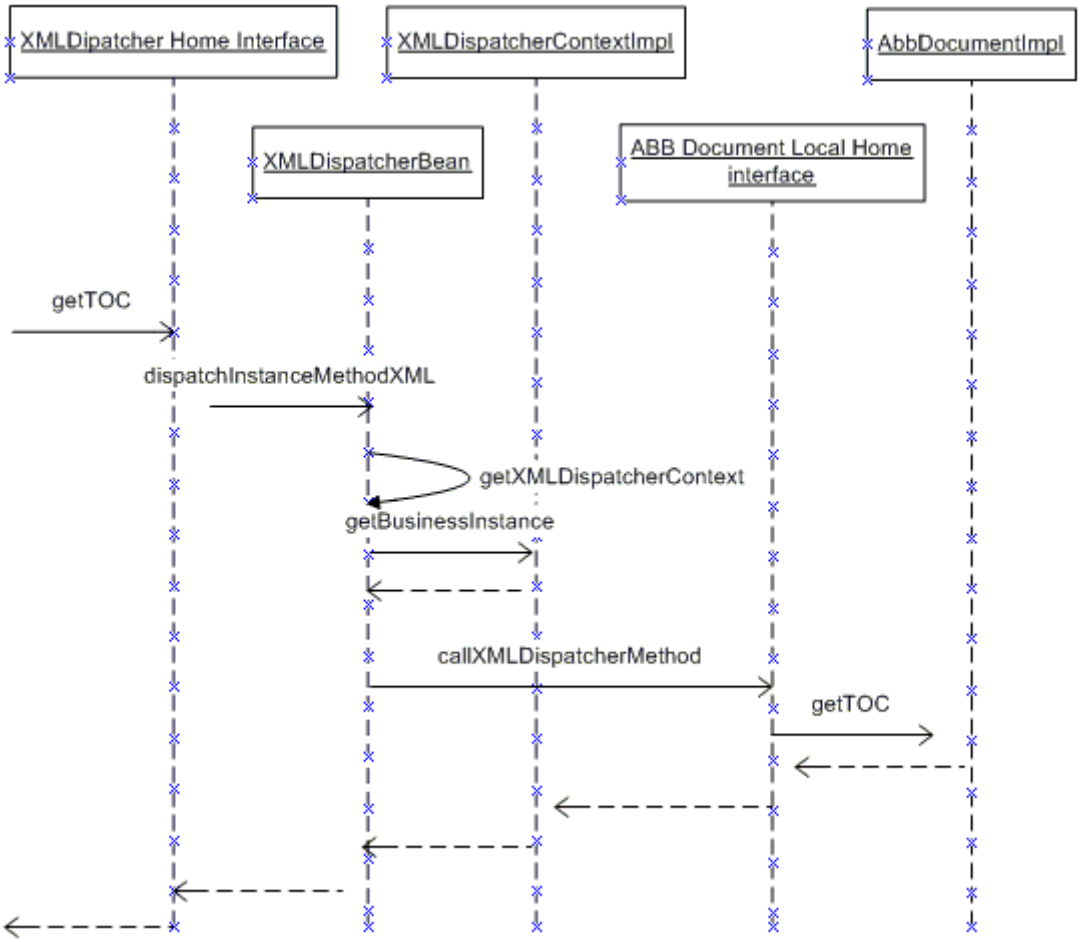
The sequence diagram below illustrates the operation of the XMLDispatcher at client level: an XMLStructMarshaller is responsible for serialising the XOo f data structure; an HTTP adapter passes the request to the protocol HTTP.



The sequence diagram below shows the interactions at web server level. The point of reception of the HTTP message is the class "HttpServlet"; next is "HttpServletBase", which is part of the XOo f framework. The request is then processed by "RMGServlet" itself, which instantiates an "XMLDispatcherEJBClient"; this creates an "XMLDispatcherHome", which will be used by "RMGServlet" to execute the method "dispatchInstanceMethodXML"; as a result of this, an RMI request is sent to the EJB server.



The last sequence diagram illustrates what happens at the EJB server. The request is received on the Home (remote) interface of the XMLDispatcher, which is a "session bean". This request is processed by the class "XMLDispatcherBean"; this obtains an "XMLDispatcherContext", which is used to determine which instance of the "entity bean" should be used. The request is then delegated to the local interface of this bean (in this case, the method "getTOC" is called from "AbbDocumentImpl").

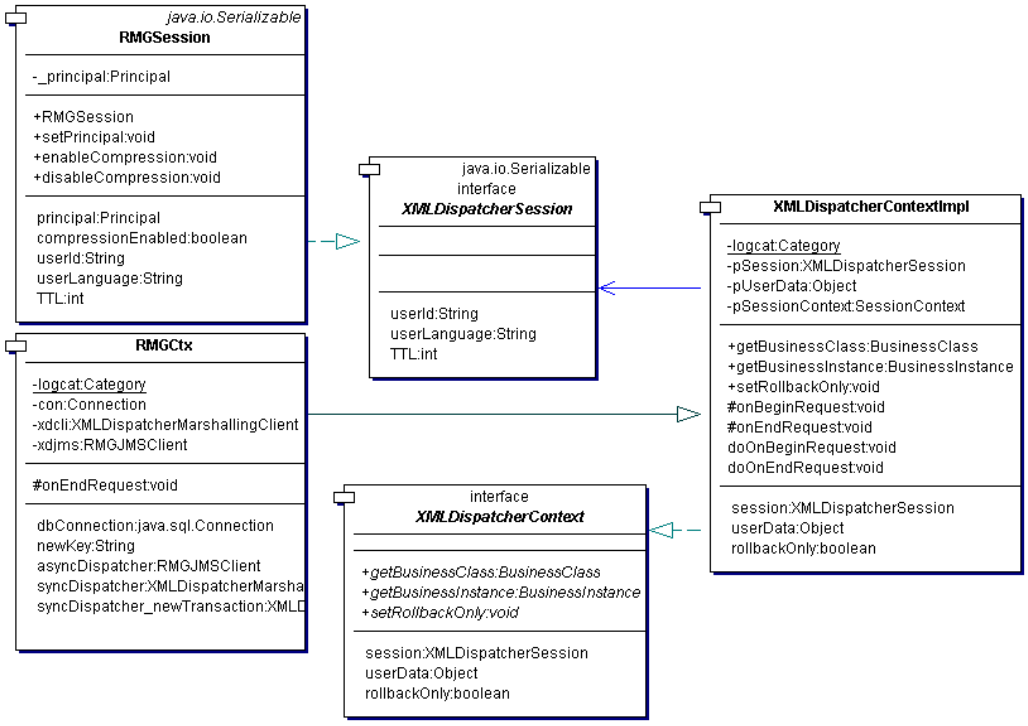


5.1.4. The session context

The session context is established when the user logs in; at the application server, the login is processed in the "Login" method of the "Server" object; this method receives the "LoginRqst" login request parameter, which provides a username and password.

The "Login" method performs user authentication (see the section on security) and receives the object "Principal". The "Principal" object is used as a parameter for the creation of a new session context (RMGSession). This session context is then stored in the RMG context ("RMGCxt"). As shown in the class diagram below, the class "RMGCxt" inherits from the class "XMLDispatcherContextImpl", which is itself an implementation of the interface "XMLDispatcherContext".

Every request made in the current session (each EJB object method) can interrogate and recover this session context in order to validate any execution permissions.



5.1.5. Asynchronous requests

In the SEI-BUD system, certain requests are likely to take more than a few seconds. If so, they are handled asynchronously. The client station sends its request synchronously and receives notification of acceptance (request being processed). What it actually receives is an object (ProcessingReport) for the request being processed, and it can use this object to interrogate the server on the progress of its request.

5.1.5.1. Synchronous part

In this case, the request is divided into two parts. The first part of the request is synchronous and the mechanism used is identical to the synchronous request mechanism; however, the server code part only posts the request using a "message bean" and creates a ProcessingReport object; this first part of the request then ends by returning a reply indicating that the request is being processed.

To give an example, the method for reserving a fragment in author mode ("checkout_authoring_async") receives as input parameters an XMLDispatcherContext and the XOo f structure determining the parameters of the request; it returns a string of characters containing the identifier of the "ProcessingReport" object created. The different stages executed after this method are as follows:

— an "asynchronous dispatcher" is obtained using the "XMLDispatcherContext"; this is done by creating an "RMGJMSClient", which inherits from the "XMLDispatcherJMSClient", and by interrogating the context JNDI ("Java and Naming Directory Interface") regarding the inputs "RMG_QUEUE_CONNECTION" and "RMG_RQST_QUEUE".

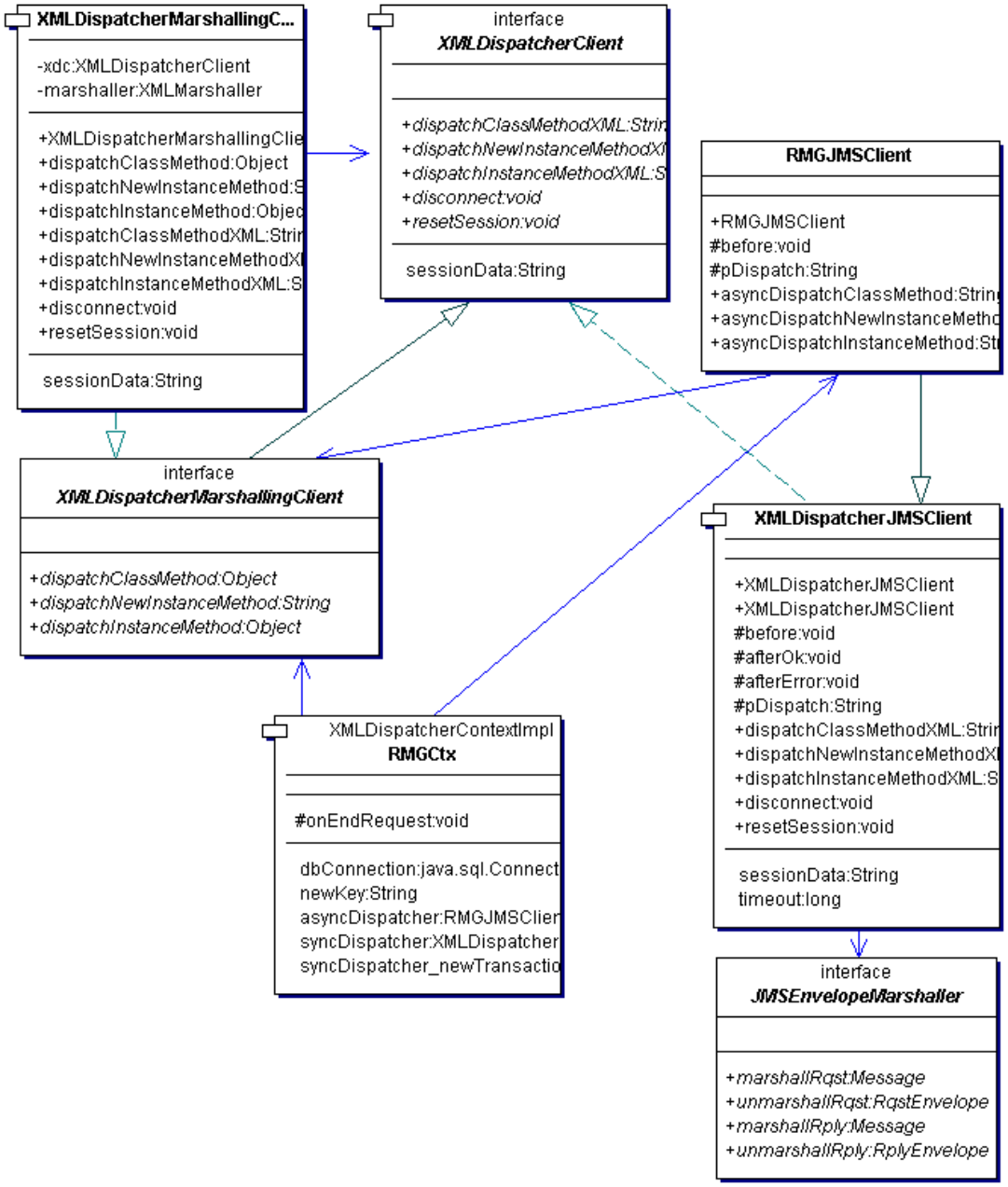
— "RMGJMSClient" is used to execute its method "asyncDispatchInstanceMethod", which carries out the following operations:

— creation of a new processing report ("ProcessingReport");

— use of an "XMLDispatcherMarshallingClient" to activate ("dispatchNewInstanceMethod") the method "new" in order to create a new instance of the "ProcessingReport" object

— invocation of the method of the basic class "XMLDispatcherJMSClient" in order to post the request to the right message bean.

The simplified class diagram below shows the interactions between the different parts of the XMLDispatcher and the specific SEI-BUD code.

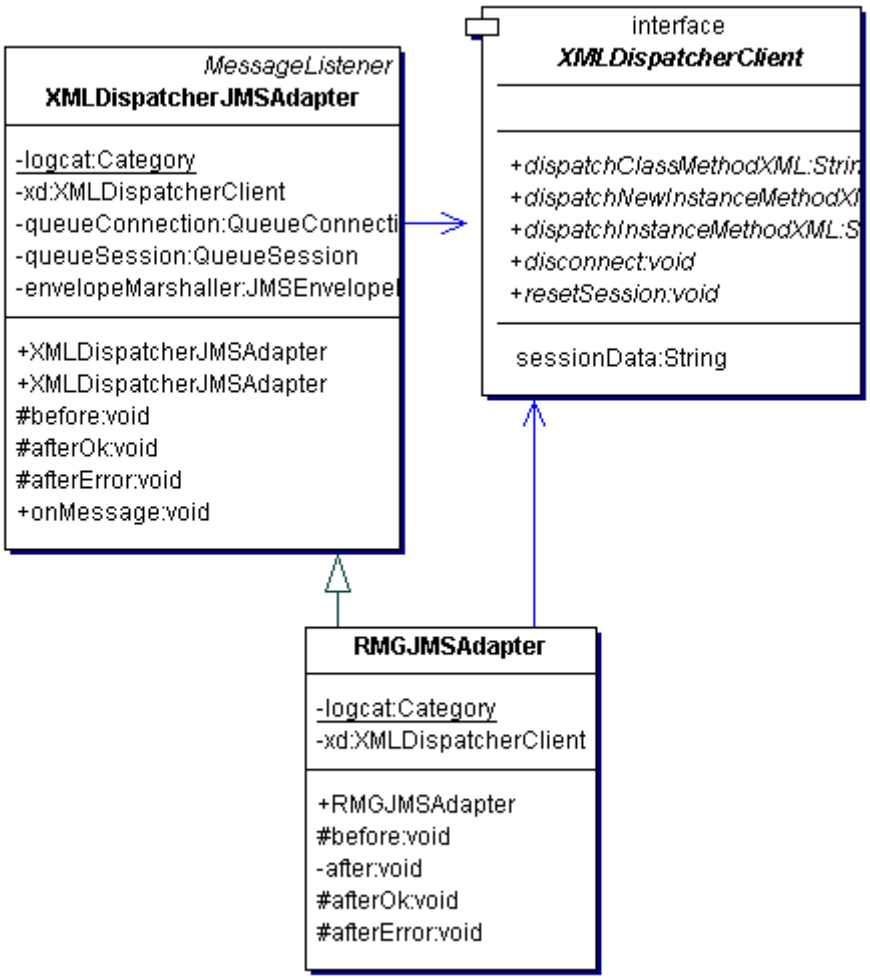


5.1.5.2. Asynchronous part

The second part of the request involves the processing of the message as if it came from a client.

It should be noted that the execution of this second part of the request is of a lower priority than the first part; for the first part, the user is waiting for a response from the server, while for the second part, the user knows that it is a long request and it is being processed. This means that synchronous requests (such as those for which it is necessary to obtain contextual information during navigation in the nomenclature) and the synchronous parts of asynchronous requests can be given preferential treatment.

This message is received using the method "onMessage"; this method is defined in the interface "MessageListener" (J2EE framework) and is implemented in "XMLDispatcherJMSAdapter" from which "RMGJMSAdapter" inherits (see the class diagram below).



The execution of the "onMessage" method can be broken down into the following steps:

— unmarshalling of the message;

— invocation of the method "before"; this method enables the class derived from the "XMLDispatcherJMSAdapter" to process the message before dispatching; in our case, "RMGJMSAdapter" executes the following operations:

  — extraction of the "ProcessingReport" identifier from the message

  — activation of the method "started" from the "ProcessingReport" object;

— depending on the type of method specified in the message wrapper (instance method, class method), dispatch of the correct method, "dispatchInstanceMethodXML" in our example, by specifying the

object class, the object method, the object identifier and the structure including the parameters of the request. This leads to the following steps:

- In the example chosen, it is the method "checkout_authoring_syncwrapper" that is called up;
- It executes the method "checkout_authoring" ("AbbDocumentImpl");
- This method returns a result that is packaged as an attachment to a processing report ("StructProcessingReportAttachedResult").

— if the request ends without an exception, the result is marshalled;

— invocation of the method "afterOK"; unlike the method "before", this method enables the class derived from the "XMLDispatcherJMSAdapter" to process the message before dispatch; in our case, "RMGJMSAdapter" executes the following operations:

- activation of the method "succeeded" of the object "ProcessingReport"; the effect of this is to attach the result to the "ProcessingReport" and consider that the processing is finished. The client can then interrogate the processing report object and obtain the attached result.

5.1.5.3. Controlling the number of concurrent asynchronous requests

Some business operations are considered as batch operations and may take quite a long time to complete (for instance, operations like cross document reports): it is desirable to control the number of such concurrent operations in order keep enough server processing power for more interactive operations.

Some user requests may result in hundreds or thousands of server requests; for instance when a user requests the translation of tens of documents in 20 different languages, the server needs to prepare and send hundreds of documents for translation. In order to keep the server useable for interactive requests, it is necessary to control the number of such concurrent requests.

By default, all asynchronous requests share the same message queue, and the maximum number of beans handling such asynchronous requests is controlled by the `MDB_Conf_Default` configuration entry in `jboss.xml` application configuration file (typically, there is a provision for 15 concurrent asynchronous requests). It is not desirable that very lengthy requests share the same message queue, because lengthy requests could take all the slots and prevent quicker asynchronous requests to execute.

The best solution to this controlling problem would be to add some sort of 'background task manager' that would execute outside of the application server and issue server calls depending on configuration parameters. This solution would make it possible to dynamically modify the server load.

But the actual solution to this problem is based on pure application server functionality and has been chosen because it required less effort. The point is to multiply the number of message queues by type of request. There is currently one additional message queue for `rtf-read-only` reports and another message queue for the preparation of translation text. These are respectively described in `ejb-jar.xml` as `RMGJMSAdapterMDB_T1` and `RMGJMSAdapterMDB_TRANSLATION` `message-driven` entries that specify a message selector (example: `taskSelector = 'amd-single-translation'`).

The link between the `message-driven` entry and the container configuration is done through the `message-driven` entry in the `jboss.xml` application file; this container configuration specifies a different number of concurrent beans. Of course, these additional queues refer to the same bean code.

How are such asynchronous requests dispatched in one or another message queue? The response lies in the `RMGJMSClient` (extending the `XMLDispatcherJMSClient`).

— when creating a`ARMGJMSClient`, the`taskSelectorValue`attribute is emptied;

— in the`pDispatch`method, we get a chance to look at the request and its parameters and decide in which message queue the request should be posted; the decision is store by assigned the`taskSelectorValue`value;

— in the hook that is called just before inserting the message in the JMS request queue (the`before`method), a`taskSelector`property is added to the message;

— this`taskSelector`property will then be used by the container to control the creation of beans based on the configuration parameters.

5.1.6. Business objects

5.1.6.1. Classes

In the SEI-BUD system, business objects are expressed in XML and respond to a model expressed by a DTD. This modelling forces the developers to work to the same methodology and allows the XOo f framework to process object definitions automatically in order to generate HTML or LaTeX documentation for them and principally to make the developer's task easier by generating code automatically. It should be noted that in the SEI-BUD system, this code is Java for EJB (but it could also be, for example, VB code for COM objects).

```
<!ELEMENT class (descr+, doc*, classmethods?, defaultinterface?, fsm?)>
<!ATTLIST class name CDATA #REQUIRED>
<!ELEMENT classmethods (event+)>
<!ELEMENT defaultinterface (extends*,event*)>
<!ELEMENT extends EMPTY>
```

An XOx f business object consists of a text description and a documentation part (which may include text, images, lists, etc.). The name of the object or class is given as an attribute and will be used when generating the code. Any class methods ("`classmethods`") are then specified; these are methods that do not require the creation of an object in order to be used. The "`defaultinterface`"part specifies the instance methods and the "`fsm`"part enables a diagram of states/transitions to be specified, modelling and controlling the dynamic aspect of the object.

```
<!ELEMENT event (descr+,rqst?,rply?,doc*)>
<!ATTLIST event
name CDATA #REQUIRED
visibility (public|private) "public"
special (constructor|destructor) #IMPLIED>
<!ATTLIST extends
interface CDATA #REQUIRED
interfacepath CDATA ".">
```

A class method or instance method is represented by the construction "`event`". An instance method can also specify that it is inheriting from another class: the "`interface" attribute of the element`"`extends`"; this attribute refers to an object that must be defined in another XML specification document (the attribute "`interfacepath`"enables the access path to this document to be set where there is any potential ambiguity). The attribute "`visibility" is used to`indicate whether the method is public or private; a private method would be used, for example, if it was used by another object but was not on view to clients. The attribute "special" is used to indicate, in particular to the code generator, that this method is to be processed in a special way.

From a documentation point of view, a method must be described briefly or, optionally, must be the subject of detailed documentation. A method can have an input parameter and an output parameter.

```
<!ELEMENT rqst EMPTY>
<!ELEMENT rply EMPTY>
```

```
<!ATTLIST (rqst|rply)
class CDATA #REQUIRED
classpath CDATA "."
list (y|n) "n"
optional (y|n) "n"
validated (y|n) "y">
```

The input and output parameters of methods are specified in a similar way. The attribute refers to an XOo f structure name (see the definition below); this structure is described in a separate XML document. The attribute "`classpath`"is used to set an access path to this definition in the event of any ambiguity. The attribute "`optional`" indicates whether it is a collection of structures. The attribute determines whether the parameter is compulsory or optional. The last attribute ("`validated`") indicates whether the syntax of the parameter should be validated (we will see further on that it is also possible to programme validation rules).

```
<!ELEMENT fsm (doc*, nstate,(mstate|state)+)>
<!ELEMENT nstate (transition*)>
<!ATTLIST nstate
name CDATA #FIXED "nihil">
<!ELEMENT mstate (descr*,doc*,transition*,(mstate|state)+)>
<!ELEMENT state (descr+,doc*,transition*)>
<!ATTLIST state
name CDATA #REQUIRED>
<!ELEMENT transition EMPTY>
<!ATTLIST transition
event CDATA #REQUIRED
nextstate CDATA #IMPLIED
impl CDATA #IMPLIED>
```

The different instance methods are structured using a diagram of states/transitions ("fsm" for finite state machine): these methods correspond to transitions in the diagram, and the XOo f framework checks dynamically that only methods corresponding to the transitions permitted in the current state are authorised. The state "`nstate`" ("nihil state") is a special state: the transitions that start from that state are triggered by a construction event.

A transition is triggered by an event (attribute "event") and can specify the state that follows (attribute "nextstate"); the attribute "impl" specifies the name of the instance method to be executed.

The element "state" is a normal state, while the element "mstate" should be considered a "macro state", which is to be considered a write facility.

5.1.6.2. Data structures manipulated by classes

The input and output parameters of the class and instance methods are expressed formally in a similar way to the simplest XML schemas. The aim is to achieve a simple mapping towards various programming languages and therefore to be able to generate code easily, to be able to manipulate it easily without seeing the XML, and to be able to document and validate it.

```
<!ELEMENT struct (descr+,doc*,(vfield|gfield|vlfield|glfield|xmlfield)*,validate*)>
<!ATTLIST struct
class CDATA #REQUIRED
baseclass CDATA #IMPLIED
baseclasspath CDATA ".">
```

The "`struct`" element is the root element of a data structure specification. The attribute "`class`" is the name of the data structure; this name is taken as the name of the input or output parameter of the class or instance methods. The attribute "`baseclass`" can specify the name of a structure from which to inherit; the structure derived in this way inherits from all the fields in the base structure. The attribute "`baseclasspath`" takes the absolute or relative URL of the directory of the base structure (for which the file naming convention is "baseclass.xml"). The file specifying this structure must fulfil the naming

convention "class.xml" where "class" is the name of the structure referred to by the attribute "class".

The content of a structure consists of a description, optional documentation, and a number of field definitions and validation rules for the complete structure (i.e. rules for validation between fields).

```
<!ELEMENT vfield (descr+,
(tstring|tint|tdecimal|tboolean|tcode|tdatetime|ttime|
tdate|tbinary), default?,doc*,validate*)>
<!ELEMENT default (#PCDATA)>
<!ATTLIST vfield
name CDATA #REQUIRED
mandatory (y|n) "y"
serialize (attribute|element|pcdata) "element">
```

The element "vfield" is used to define a simple field (as opposed to a compound field). The attribute "name" takes the field name; "mandatory" indicates whether the field is mandatory or optional. The attribute "serialize" specifies how the field must be serialised in XML: the possible options are:

— as an element;

— as an attribute of the parent element;

— as an item of data of the parent element; this is only allowed in one field of the structure.

The content of a simple field is determined by a description, one of the different preconfigured base types (see details further on), a default value that is optional for the field ("default") if the field is not mandatory, documentation and possibly some validation rules. The content of the default value must respect XML representation rules for the type of field in question, and must also respect all the constraints for the type of field.

```
<!ELEMENT vlfield (descr+,
(tstring|tint|tdecimal|tboolean|tcode|tdatetime|ttime|
tdate|tbinary),doc*,validate*)>
<!ATTLIST vlfield
name CDATA #REQUIRED
maxOccur CDATA #IMPLIED
minOccur CDATA "0">
```

The field "`vlfield`" is for specifying a field made up of a list of simple fields. The attributes "`minOccur`" and "`maxOccur`" determine, where appropriate, the minimum and maximum number of occurrences of the simple field in the list.

```
<!ELEMENT gfield (descr+,doc*,validate*)>
<!ATTLIST gfield
name CDATA #REQUIRED
mandatory (y|n) "y"
class CDATA #REQUIRED
classpath CDATA ".">
```

The field "`gfield`" is used to declare a compound field, by making reference to a data structure using the attributes "`name`", "`class`" and "`classpath`".

```
<!ELEMENT glfield (descr+,doc*,validate*)>
<!ATTLIST glfield
name CDATA #REQUIRED
maxOccur CDATA #IMPLIED
minOccur CDATA "0"
class CDATA #REQUIRED
classpath CDATA ".">
```

The field "`glfield`" is like the field "`gfield`", except for the fact that it is used to declare a list of identical data structures. As with the field "vlfield", the attributes "`minOccur`" and "`maxOccur`" determine, where appropriate, the minimum and maximum number of occurrences of the simple field in the list.

```
<!ELEMENT xmlfield (descr+,doc*,validate*)>
<!ATTLIST xmlfield
name CDATA #REQUIRED
mandatory (y|n) "y">
```

The field "xmlfield" is a container for any well-formed XML data. The field value is an element, the name of which is given by the attribute "". Once it has been deserialised, it is transformed into a DOM "DocumentFragment" in the target programming language.

```
<!ELEMENT validate (#PCDATA)>
<!ATTLIST validate
language (vb|java|python) #REQUIRED>
```

The element "validate" is destined to contain validation rules in the form of code depending on the target programming language. The method for accessing field values also depends on the programming language used. The SEI-BUD system does not use this mechanism.

5.1.6.3. Base types

The known base types for the XOo f framework are: tstring, tint, tdecimal, tboolean, tcode, tdatetime, ttime, tdate and tbinary.

```
<!ELEMENT tstring (choices?)>
<!ATTLIST tstring
maxLen CDATA #IMPLIED
minLen CDATA "1"
regexp CDATA #IMPLIED>
```

"tstring" is used for strings of Unicode characters. It may have a minimum length constraint "minLen" and a maximum length constraint "maxLen". It may also have a regular expression constraint (not used for SEI-BUD).

```
<!ELEMENT choices (choice+)>
<!ELEMENT choice (choice.value,choice.descr+,choice.doc*)>
<!ELEMENT choice.value (#PCDATA)>
<!ELEMENT choice.descr (#PCDATA)>
<!ATTLIST choice.descr
xml:lang (%langs;) #IMPLIED>
<!ELEMENT choice.doc (%text;)>
<!ATTLIST choice.doc
xml:lang (%langs;) #IMPLIED>
```

Constraints can be placed on valid values using the element "choices". Each value has a description, which may be multilingual, and documentation. If there are a large number of possible values, it is better to use the type "tcode".

```
<!ELEMENT tint (choices?)>
<!ATTLIST tint
maxVal CDATA #IMPLIED
minVal CDATA #IMPLIED>
```

"tint" represents an integer value from 32 bits. Once again, a minimum value "minVal" and a maximum value "maxVal" can be specified.

```
<!ELEMENT tdecimal (choices?)>
<!ATTLIST tdecimal
maxVal CDATA #IMPLIED
minVal CDATA #IMPLIED
fractionDigits CDATA #REQUIRED>
```

"tdecimal" is used for decimal values. The attribute "" indicates the maximum number of decimals, while "minVal" and "maxVal" represent the minimum and maximum values.

```
<!ELEMENT tboolean EMPTY>
The type "tboolean" takes the value "true" or "false".
<!ELEMENT tcode EMPTY>
```

```
<!ATTLIST tcode
name CDATA #REQUIRED>
```

"tcode" is used to represent a reference to a parameter table designated by the value of the attribute "". The meaning of this attribute depends on the application.

```
<!ELEMENT tdatetime EMPTY>
<!ELEMENT ttime EMPTY>
<!ELEMENT tdate EMPTY>
```

"tdatetime" is used to specify a date and time; "ttime" specifies a time in a day and "tdate" specifies a date.

```
<!ELEMENT tbinary EMPTY>
<!ATTLIST tbinary
encoding (base64) "base64">
```

Finally, the type "tbinary" is used to specify an item of binary data that is not encoded in Unicode, but in "base64".

5.1.7. Bean locking strategy

The SEI-AMD system has the following characteristics that influence what the best locking strategy is:

— Most operations are performed on the amendment level

— Some of these operations are fast, but others can last several seconds

— Amendments are usually not shared between authors

— When amendments are public, there are more read operations than write operations

— Report creation iterates over a list of amendments, report creation takes a long time

— Milestone reports are made when there is no more authoring activity

In an EJB container only one instance of a bean can exist. With instance, an entity with a unique primary key is meant.

Especially the reports pose problems. In the default locking strategy, all amendments that are to be integrated in the report will get locked during the creation of the report and will block all further access to them. This implies that another report that includes those amendments will have to wait until the first report is finished.

There are several strategies to deal with locking-problems:

— Let all read-operations fall outside the transaction boundary. This means that on read operations the beans aren't locked. An implication is that two read operations in the same transaction could give different results.

— Configure separate beans for read-only operations and read/update operations. The clients must then address the correct beans. In our case this would not work well with the XOo f interfaces, unless all specifications were duplicated too.

— Split big transactions into smaller transactions. This resembles a lot the first option, but is programmed in the application code. This complicates the process flow and code, especially if the smaller transactions are dispatched asynchronously.

— Let all transactions use their own instance of the bean. This is a special configuration of the JBOSS EJB Container. Every transaction creates its own instances of the beans, which avoids locking problems. Since the WebInt client application does not use any locks on the data (it uses an

optimistic strategy), there's no functional implication towards the users. This option does not allow the reuse of cached beans between transactions.

With Jboss 2.4.4, this locking strategy is specified in the "jboss.xml" application configuration file. In the`container-configuration`section, the following lines:

```
<interceptor>org.jboss.ejb.plugins.EntitySynchronizationInterceptor</interceptor>
<interceptor>org.jboss.ejb.plugins.EntityInstanceInterceptor</interceptor>
<locking-policy>org.jboss.ejb.plugins.lock.QueuedPessimisticEJBLock</locking-policy>
<commit-option>A</commit-option>
```

are replaced with:

```
<interceptor>org.jboss.ejb.plugins.EntityMultiInstanceInterceptor</interceptor><interceptor>or
g.jboss.ejb.plugins.EntityMultiInstanceSynchronizationInterceptor</interceptor>
<locking-policy>org.jboss.ejb.plugins.lock.MethodOnlyEJBLock</locking-policy>
<commit-option>B</commit-option>
```

5.2. Business logic

See annex : annex-MRF-BLO

5.3. Security

There are two types of security device within SEI-BUD:

— **Authentication device**. The aim of the authentication device is to ensure that only recognised users access the application.

— **Authorisation device**. The authorisation device ensures that an authenticated user only interacts with data over which he has certain rights.

The principles behind these devices and their implementation are explained in the sections that follow.

5.3.1. Authentication

5.3.1.1. Principles

A mechanism similar to HTTP basic authentication, i.e. based on a username and password, is used for authentication. This authentication mechanism is not very secure because the password is not encoded in any way. Its use is considered acceptable in an environment such as SEI-BUD.

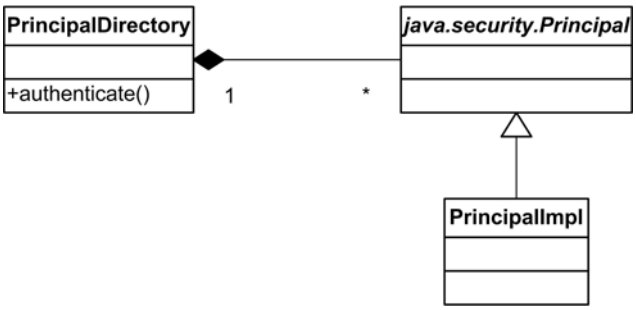
5.3.1.2. `xef.security`package

The package`xef.security`implements the functions of the SEI-BUD authentication system, namely:

— User directory management;

— The authentication logic.

The classes of the`security`package used for authentication are shown in the diagram below:



The***Principal***interface is defined in the package`java.security`. This interface represents the abstract notion of "identified entity". Within the SEI-BUD framework, the identified entities are the users and "*daemons*" ("watchdog", translation update process of the Translation Service). The***Principal***interface is implemented by the class ***PrincipalImpl*** of the package`java.security`, which defines the attributes of an identified user: the identifier, password and a description.

In the current version, the classes of the package`security`do not give a method for updating authentication system data. The directory of users is made persistent in a file in XML format; this file can be modified using an ordinary text editor (the client has not asked for an interface for managing the user directory). Changes are detected and incorporated dynamically by the system.

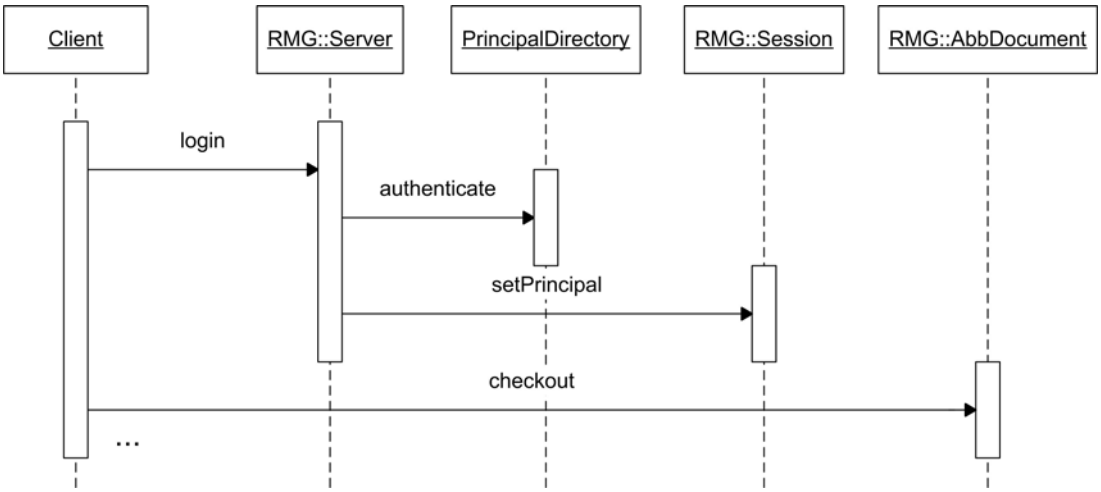
N.B.: An alternative to this "minimalist" approach would be to interface SEI-BUD with an enterprise directory (LDAP). The usual directory administration tools would be used to manage users. A directory of this kind has apparently not yet been established at an inter-institutional level.

The class ***PrincipalDirectory*** carries out a DOM parsing of the file used to store the authentication data (this is the same file that is used to define the group hierarchy (see "Authorisation")) and instantiates the*Principal*objects. The method*PrincipalDirectory ::authenticate()*is used to authenticate a user on the basis of his user identifier and password.

5.3.1.3. Implementation

The authentication system implemented in the package`xef.security`is used by the Repository Manager (RMG) to implement authentication in SEI-BUD. Before being able to invoke the services of the RMG, the client creates a session by invoking the RMG method*Server ::login()*. This method in turn invokes the method*PrincipalDirectory ::authenticate()*to validate the user's authentication data (identifier and password). A*Principal*object is instantiated and associated with the session*RMGSession*. The user identifier is thus made available to the access control system, which intervenes at the start of processing of each request.

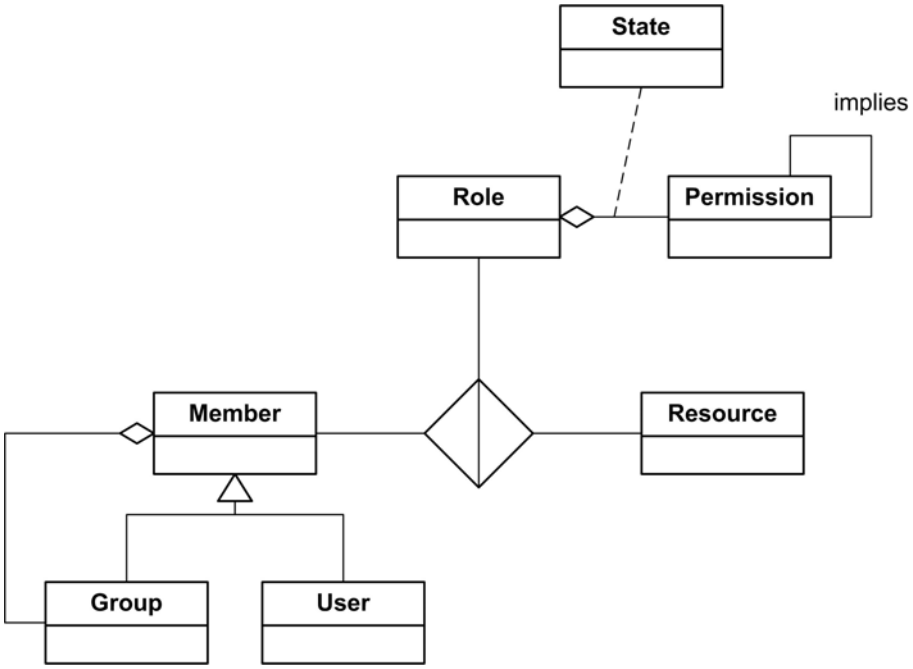
The call sequence for authentication can be represented by the following sequence diagram:



5.3.2. Authorisation

5.3.2.1. Principles

The authorisation system[1] ensures that an authenticated user only interacts with data over which he has certain rights. The system implemented in SEI-BUD uses the concepts of resource, role and member. These concepts and their interactions are represented in the diagram below[2]:



Resources

---

[1] Also known as "access control". This is a less accurate name because it covers both authentication and authorisation.

[2] This diagram aims to show the concepts used in the authorisation system. It is not an accurate representation of the system implementation (see "Implementation" section).

The authorisation system implemented in SEI-BUD can be described as "**resource based**". Systems of this kind allow user permissions to be specified for each resource individually. The permissions granted to a particular user may vary from one resource to another. The Windows file system is an example of a system with resource-based access control.

Since the resources to which access control relates can be of various kinds, the permissions that apply can also vary from one resource to another.

In SEI-BUD, the budgetary publications are the resources to which access control relates. Each publication consists of a set of documents that are the different language versions of the publication. The language versions can themselves be edited using several separate views (figures, remarks, etc.). It must be possible to define permissions for each view and each language version individually.

N.B.: The need to define permissions at the level of the budget fragment has been identified on several occasions. Some publications include contributions from several institutions; each contribution is a separate part. Currently, SEI-BUD does not allow permissions to be specified at fragment level. Aside from the technical difficulties, access control at fragment level would require greater management.

Permissions and roles

In SEI-BUD, authorisations are defined by granting permissions to users. In other systems, authorisations are defined more directly by conferring rights on interface methods (operations). There are two advantages to the use of permissions. A permission can cover a group of methods, which can prove necessary when several methods are used together (the methods "checkout" and "checkin", for example). The second advantage is that permissions can be linked in a relationship of "implication". For example, write permission must of necessity imply read permission.

The authorisation system implemented in SEI-BUD can be described as "**role based**". This type of system allows a role, i.e. a set of permissions, to be associated with each user (or group of users). The notion of role can be used to express the rights of each user precisely and in terms that closely resemble the terms of business. The roles defined in the authorisation system generally reflect the responsibilities held within the organisation. In SEI-BUD, for example, there are "author" and "translator" roles.

By combining the principles of the resource-based and role-based systems, the authorisation system implemented within SEI-BUD means that a role can be attributed to each user. This role can vary from one resource to another.

Members

Sometimes several users assume the same role with respect to a group of resources. The notion of user groups was introduced to avoid having to repeat for each resource the list of users assuming that role.

A group is therefore a set of users grouped according to the common role they assume with respect to a set of resources. Groups can be assembled in turn to form a hierarchy of groups.

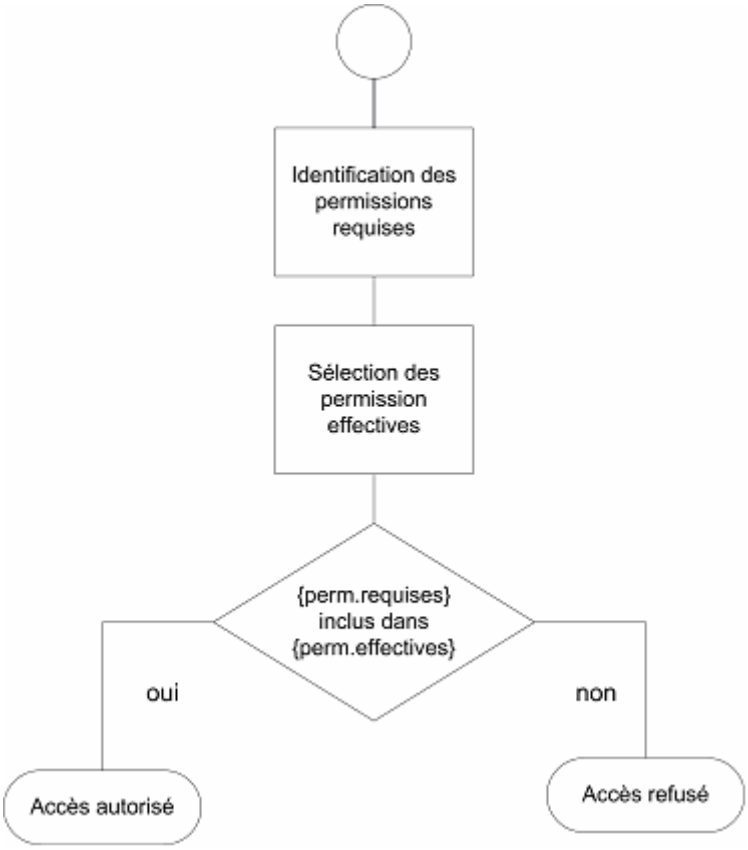
The notion of member is a generic notion that encompasses the notions of user and of group. Roles are assumed by members, i.e. either by individual users or by groups of users.

Dynamic access control

Budgetary publications are prepared in a number of successive stages. Generally the following four stages are identified: the authoring phase, the translation phase, the correction phase and the layout and printing phase[3]. A user will only participate in certain clearly defined phases. In this sense, access control is considered dynamic because it takes account of the current state of the object to be edited. The rights of a user change as the publication moves through the different stages of preparation.

Algorithm

The access control algorithm is represented by the flowchart below.



Key:

Identification des permissions requises = Identification of required permissions

Sélection des permissions effectives = Selection of actual permissions

{perm.requises} inclus dans {perm.effectives} = {required.perm} included in {actual.perm}

oui = yes

non = no

Accès autorisé = Access authorised

---

[3]For the publication of Volume 0, several authoring and translation phases take place: budget estimate director's version, budget estimate director general's version, etc.

Accès refusé = Access refused

There are three steps to the algorithm:

Identification of required permissions. The first step is to establish a match between the operation being carried out (invocation of a method from the interface) and the permissions required. The set of permissions and the operations available are fixed for any particular application. The match between the operations and the required permissions can therefore be defined statically.

Selection of actual permissions. The second step is to select the permissions the user actually has with respect to a resource. These permissions will either have been granted to the user directly, or to a group of which the user is a direct member, or to a group of which the user is a member because of the hierarchical organisation of groups. The selection of actual permissions must also take account of the current state of the resource (see "Dynamic access control").

Access diagnosis. The last step is to compare the required permissions with the actual permissions. Access is authorised when the required permissions are a subset of the actual permissions held. Any comparison of these two sets of permissions must take account of relationships of implication that exist between the permissions.

5.3.2.2. `xef.security`package

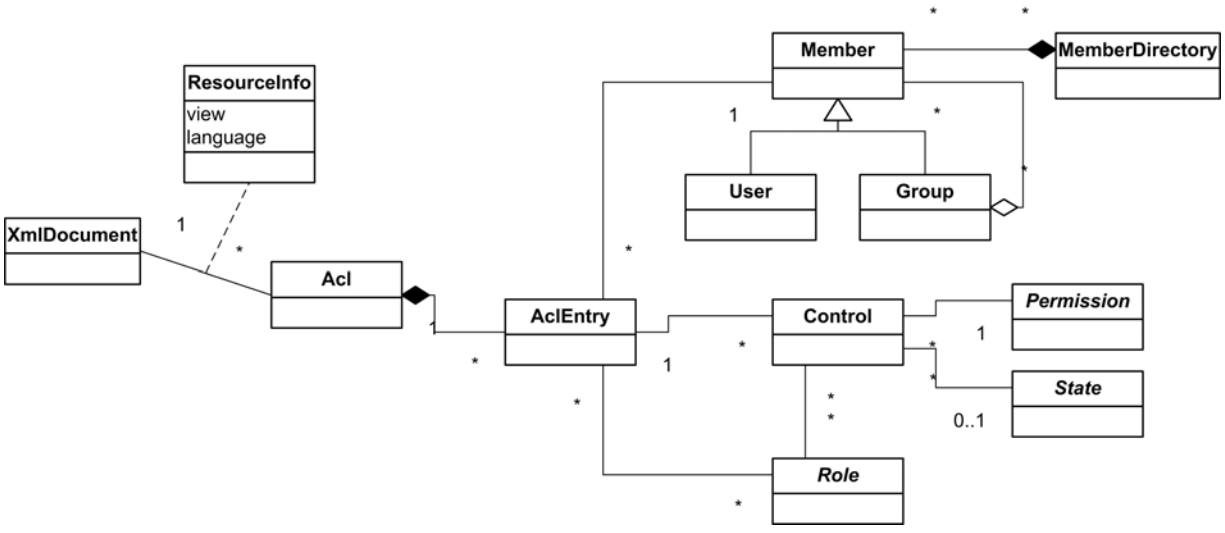
The principles described in the previous section can largely be implemented generically. The package`xef.security`groups together the generic functions of SEI-BUD access control, namely:

— Management of the hierarchy of users and groups;

— The allocation of permissions and roles;

— The access control logic.

The principal classes of the `security`package are shown in the diagram below:



Some classes do not have a corresponding entry in the principle diagram in the previous section (*Acl, AclEntry, Control, etc.*). These are "technical classes" introduced during the detailed design phase.

The three-way association between a resource, a member and a role is implemented by the objects*Acl*and*AclEntry*. The class ***AclEntry*** *(Access Control List Entry)*associates a member

(*Member*class) with one or more roles (*Role*class). Each instance of this class constitutes an entry in the list implemented by the class ***Acl****(Access Control List)*. Finally, each document resource is associated with an instance of the class*Acl*. By way of a reminder, in SEI-BUD resources are made up of views (remarks, figures, etc.) of the language versions of budgetary publications (the class*XmlDocument*and the association class*ResourceInfo*).

The definition of roles brings in the class ***Control***, which associates a permission (*Permission*class) with a state (*State*class)[4]. Association with a state is optional; this means that permissions can be allocated independently of state. An*AclEntry*object can be directly associated with one or more*Control*objects to allocate permissions without using roles.

The classes ***Role*** and ***Permission*** are abstract classes that constitute extension points of the security framework. Beyond the few basic permissions ("read", "write", etc.) implemented in the class ***BasicPermission***, the definition of permissions is unique to each application. To the extent that each document class can have specific methods, permissions (and roles) can vary from one document class to another. Permissions and roles are defined by extending the classes*Permission*and*Role*[5].

In the current version, the classes of the package`security`do not provide a method for updating access control data. Data are held in files in XML format and can be modified using an ordinary text editor. Changes are detected and incorporated dynamically[6].

To facilitate management, the ACLs of the resources corresponding to the different language versions and views of a publication are grouped in a single file. This file respects the`acl.dtd`grammar. The filter XSLT`acl.xsl`is used by the*Acl*class to extract the XML data from an ACL. This filter uses two parameters: a language code to identify the language version and a view identifier. When the XML data in an ACL have been isolated, a DOM parsing enables them to be read in order to instantiate the*Acl*object and its*AclEntry*entries (method*Acl ::fromDOM*).

The data defining the hierarchy of groups and users are also held in an XML file. This file respects the`principal-directory.dtd`grammar. The class ***MemberDirectory*** carries out a DOM parsing to read the data from this file and instantiate the objects***User*** and ***Group***.

5.3.2.3. Implementation

"Prior" and "post" access control

The access control system implemented by the package`xef.security`is used by the Repository Manager (RMG) to implement two access control strategies: "post" control and "prior" control.

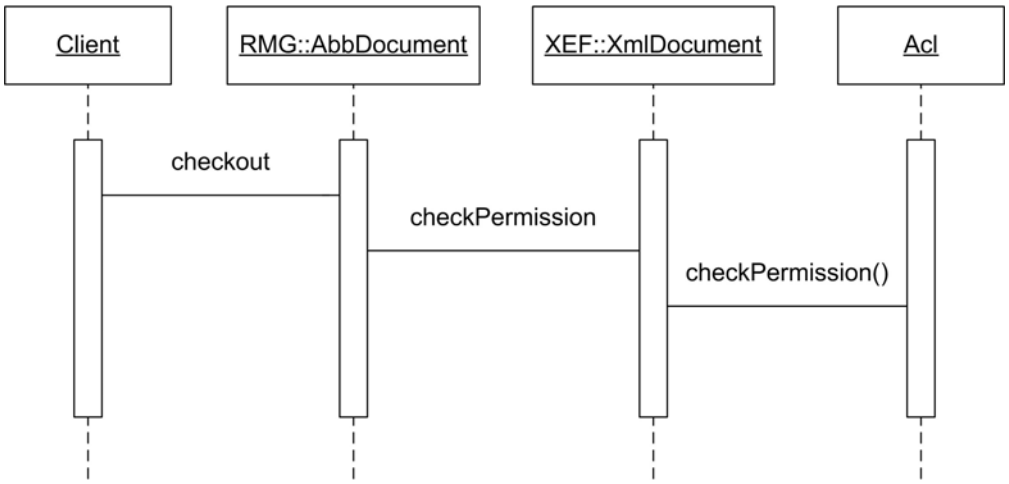
The diagram below shows how a**"post" access control**is implemented.

---

[4]The class*State*is not part of the`security`package. It belongs to the package`lifecycle`, which implements lifecycle management.

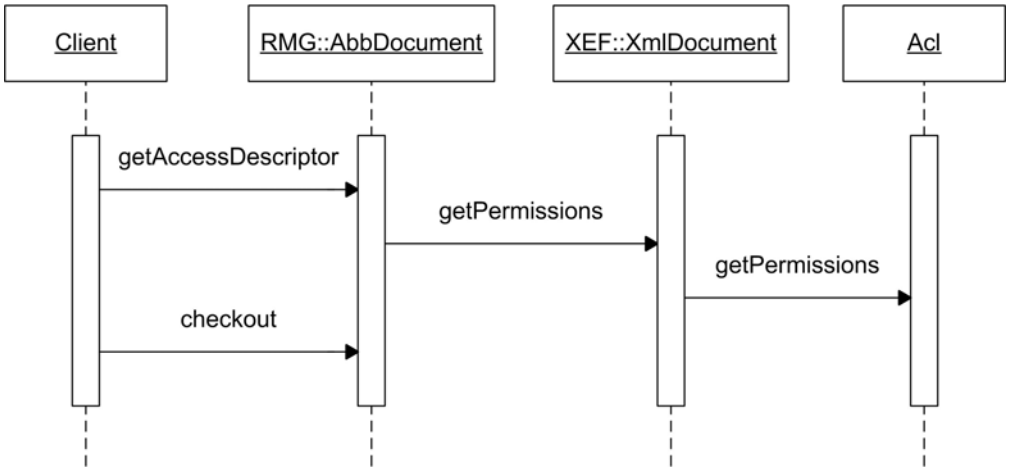
[5]A mechanism for defining permissions and roles by means of configuration is not necessary. The definition of permissions and roles is linked with the operations offered by the resources. These operations are fixed for a particular version of the application.

[6]To date, users have not asked for an interface for managing access control.



The access control system method*Acl ::checkPermission()*is invoked explicitly at the start of processing of each request[7]. As explained in the section on dynamic access control ("Principles", "Dynamic Access Control"), the permissions granted to a user change depending on the state of the resource. The current state of the resource is determined during execution of the method*XmlDocument ::checkPermission()*by invoking the "state manager" (`repository.state`package). It is then made into a parameter when the method*Acl ::checkPermission()*, which implements the access control algorithm, is called up.

The package`xef.security`is also used to implement a**"prior" access control**. The sequence of calls for the "prior" access control can be represented by the diagram below:



Instead of controlling access when a service is called up, the client only offers the services the user has the right to invoke. To implement "prior" access control, the RMG's method*getAccessDescriptor()*provides the client with a list of the services accessible to the user (taking account of the permissions granted and the current state of the resource).

The method*Acl ::getPermissions()*is used to implement this "prior" access control strategy. The RMG converts the list of permissions granted to the user into a list of accessible services.

---

[7]SEI-BUD uses an explicit access control mechanism: the application invokes the access control service itself. In contrast, a middleware offers an implicit access control service, i.e. implemented by the middleware itself.

N.B.: "post" access control is normally no longer necessary when the "prior" access control strategy is implemented. As not all SEI-BUD clients implement "prior" control (command line clients, for example), "post" access control is still needed.

Permission and role classes

The classes*Role*and*Permission*of the package`xef.security`are abstract classes that constitute extension points of the security framework. Beyond the few basic permissions ("read", "write", etc.) implemented in the class*BasicPermission*, the definition of permissions is unique to the application.

Currently, the SEI-BUD application manages two document classes: ABB publications and "Volume 0" publications. The same services are available for these two document classes and a unique permission class has been used (the class ***PermissionClass***,`rmg.bo`package). The permissions defined by this class (and those inherited from the class*BasicPermission*) are shown in the table below:

Permission	**Implies**	**Description**
read		Consultation
write	read	Reservation, updating
write-authoring	write	Reservation, updating (with carryover to other languages)
write-translation	write	Reservation, updating (without carryover to other languages)
write-nonvalid	write	Updating, possibly with use of the validation service
unlock		Cancellation of reservations by other users
diff-authoring	read	Revision marks ("author" mode)
diff-translation	read	Revision marks ("translator" mode)
diff-correction	read	Revision marks ("corrector" mode)
remove		Deletion of a publication.
send-poetry		Use of Commission Translation Service
end-phase		Phase closure
start-authoring-phase		Cancellation of phase closure

Currently two classes of roles have been implemented in the`rmg.bo`package:***AbbRoleClass*** and ***Vl0RoleClass***. The lifecycles of ABB and "Volume 0" publications differ significantly, so roles have been defined separately for each class of document.

5.4. The REUSE mechanism

5.4.1. Principles

The REUSE mechanism enables the effective "reuse" of structured information "within" and "between" publications.

The aim is to be able to insert tags in a document that indicate that "at this point" there is information that can be "reconstructed" from other information in the same document or even in other documents managed by the SEI-BUD system.

The current implementation of this mechanism in the SEI-BUD system requires the following considerations to be borne in mind:

1) The SEI-BUD system currently has no help tool for introducing REUSE tags: this is done in XML, not by the authors but by the publication supervisor.

2) When document fragments containing REUSE tags are being edited, these tags have to be retained: the author must be aware that REUSE information exists but should not be concerned with the technical details of the tagging. In the Word-based editing tool, this tagging is put in a hidden field presented in a particular way.

3) Documents extracted with the REUSE tagging resolved are "read-only" documents, i.e. they can no longer be used for updates.

4) Replacing the REUSE tagging may lead to errors if the tags are incorrect; there is actually – for the moment at least – no mechanism in the SEI-BUD system for ensuring that all the references contained in the REUSE tags can always be resolved.

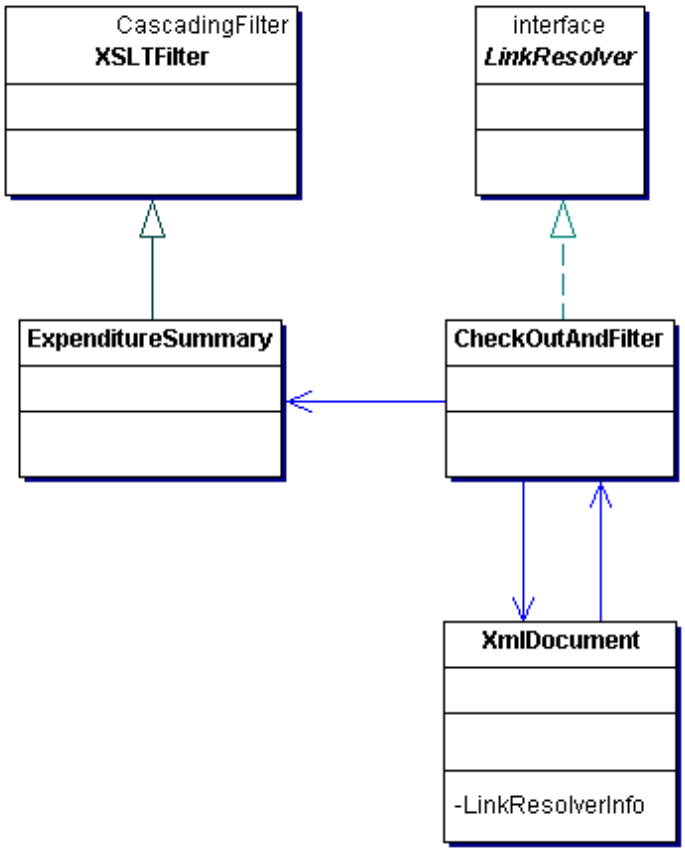
5) The decision as to whether or not to resolve the REUSE tags can be made in two ways:

   a) either by specifying it explicitly in the parameter "`CheckoutRqst`" ("`resolveInclusions`") of the method "`checkout_readOnly`" of the business object "`AbbDocument`";

   b) or by specifying that the links must be resolved ("`resolve-links`") in the export format to be used (configuration file "`document-repository-config.xml`").

5.4.2. Technical details of REUSE implementation

A simplified class diagram shows the following relationships (in the specific case of a REUSE giving an expenditure summary):



The class "`XmlDocument`" of the Xef framework has two "`checkout`" methods with different signatures:

1) the first is used by the RMG tier and receives as a parameter, notably, a format and its parameters;

2) the second is currently only used by the REUSE mechanism and receives a filter instead of a format as a parameter.

Among other things, these two methods execute the private method "`export`" of the class "`XmlDocument`"; this method receives both a format and a filter as parameters, one of which may be zero. The method "`export`" determines from the explicitly specified parameter or the format given as a parameter whether it is necessary to resolve the REUSE links. If this is the case, it is the private method "`retrieveAndResolveDocumentFragment`" that is executed.

The method "`retrieveAndResolveDocumentFragment`" uses "`retrieveDocumentFragment`" to obtain a reference to a DOM node corresponding to the requested fragment. It then determines whether the fragment in question contains links to be resolved (if not, it is not in fact necessary to clone the fragment before modifying it). If there are links to be resolved, the fragment is cloned and run through recursively "top/down" by the method "`resolveElementContent`" to locate and process the "reuse" elements.

The method "`getTargetLinkContent`" is used to obtain a document fragment corresponding to the resolution of the link. This document fragment is imported into the document under construction and replaces the "reuse" tag.

To resolve a link definition from its parameters, the method "`getTargetLinkContent`" instantiates a "`LinkResolver`" from the parameter "class" specified in the "reuse" tagging and executes the "`resolve`" method. The result of this resolution is a stream that is parsed and converted into a document fragment.

The class "`CheckOutAndFilter`" implements the interface "`LinkResolver`" and constitutes a default resolution method which is part of the Xef framework. First the method selects the target publication using the "`DocumentRepository`" given as a parameter. This resolution method can work in several ways:

1) if a format is specified, it invokes the public method "`checkout`" from the class "`Xmldocument`";

2) if a style sheet is specified, it creates an "`XSLTFilter`" and calls the variant "`checkout`" from the class "`Xmldocument`" with this filter as a parameter;

3) if a filter is specified (by a class name), it instantiates this filter before executing the method "`checkout`" with this filter as a parameter;

4) otherwise the "checkout" is executed with the default format, which is the "`global`" or "`repository`" format.

5) The class "`CheckOutDiffAndFormat`" implements the interface "`LinkResolver`" and it is used for tracking changes between two documents. First the method selects two publications using the "`publication`" and "`oldpublication`" given as the parameters. Then it invokes the public method "`checkout`" from the class "`Xmldocument`" for the both publications. It instantiates the "`diffformatter`" class which implements "`IDiffFormatter`" interface and executes the public method "`run`" with both publications as parameters. The result is document which contains changes between two initial publications.

The class "CheckOutOJAndFilter" implements the interface "LinkResolver" and it is used to apply filter on a list of publications. Method selects all publications from the "publications" list given as a parameter. Then it invokes the public method "checkout" from the class "Xmldocument" for all publications. It merges "section_name" from all publications into single publication. It instantiates the "filter" class which implements "Filter" interface and executes the public method "run" with merged publication as a parameter.

5.4.3. REUSE library for Volume 0 publication

5.4.3.1. Summary table of expenditure by title (policy).

Use: in Document II of Volume 0 (Table II.1).

Linkresolver:com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter

Filter name:com.softwareag.belgium.seibud.vl0.LinkAndReuse.ExpenditureSummary

Parameters:

1) publication: name of document including expenditure (e.g.: ABBAP2004VOL4)

2) xpath: location of the expenditure fragment (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="publication" value="ABBAP2004VOL4"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.vl0.LinkAndReuse.ExpenditureSummary"/>
</reuse-link>
```

5.4.3.2. Summary table of expenditure by financial perspective (or category) heading.

Use: in Document II of Volume 0 (Table II.2).

Linkresolver:com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter

Filter name:com.softwareag.belgium.seibud.vl0.LinkAndReuse.ExpenditureSummaryByFP

Parameters:

1) publication: name of document including expenditure (e.g.: ABBAP2004VOL4)

2) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="publication" value="ABBAP2004VOL4"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.vl0.LinkAndReuse.ExpenditureSummaryByFP"/>
</reuse-link>
```

5.4.3.3. Summary table of a title (policy) by chapter (activity) and human resources.

Use: in the analysis of expenditure by activity in Document II of Volume 0 (Table II.3).

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.vl0.LinkAndReuse.PolicySummary`

Parameters:

1) publication: name of document involving title (e.g.: ABBAP2004VOL4)

2) xpath: title xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1]/nmc-title[2])

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="publication" value="ABBAP2004VOL4"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]/nmc-title[2]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.vl0.LinkAndReuse.PolicySummary"/>
</reuse-link>
```

5.4.3.4. Summary table of a title (policy) by chapter (activity), by article in a chapter and human resources.

Use: in the analysis of expenditure by activity in Document II of Volume 0 (Table II.3).

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.vl0.LinkAndReuse.PolicyAndChapterSummary`

Parameters:

1) publication: name of document involving title (e.g.: ABBAP2004VOL4)

2) xpath: title xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1]/nmc-title[26])

3) alias: last part of the chapter alias (default: 01)

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="publication" value="ABBAP2004VOL4"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]/nmc-title[26]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.vl0.LinkAndReuse.PolicyAndChapterSummary"/>
</reuse-link>
```

5.4.3.5. Table of expenditure line by line.

Use: in Document IV of Volume 0.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.vl0.LinkAndReuse.BudgetLineByLine`

Parameters:

1) publication: name of document including expenditure (e.g.: ABBAP2004VOL4)

2) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="publication" value="ABBAP2004VOL4"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.vl0.LinkAndReuse.BudgetLineByLine"/>
</reuse-link>
```

5.4.3.6. Table of revenue line by line.

Use: in Document IV of Volume 0.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.vl0.LinkAndReuse.BudgetLineByLine`

Parameters:

— publication: name of document including revenue (e.g.: ABBAP2004VOL1)

— xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-revenue[1])

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="publication" value="BUDAP2004VOL1"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[2]/nmc-revenue[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.vl0.LinkAndReuse.BudgetLineByLine"/>
</reuse-link>
```

5.4.3.7. Enlargement table of expenditure line by line.

Use: in Document IV of Volume 0.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.vl0.LinkAndReuse.EnlargementLineByLine`

Parameters:

1) publication: name of document including expenditure (e.g.: ABBAP2004VOL4)

2) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="publication" value="ABBAP2004VOL4"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
```

```
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.vl0.LinkAndReuse.EnlargementLineByLine"/>
</reuse-link>
```

5.4.3.8. General Statement of revenue table

Use: in “Estimate of Revenue and Expenditure of the European Parliament” report

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutOJAndFilter`

Filter name:`com.softwareag.belgium.seibud.vl0.LinkAndReuse.ContributionStatement`

Parameters:

1) publications: list of publication parameters
2) publication: name of (abb-like) document containing revenue and expenditure (e.g.: the Estimate document of Parliament BUDAP2007VOL2)
3) section_name: name of the section element containing revenue and expenditure (nmc-section)
4) xpath: xpath of document fragment containing revenue and expenditure (e.g.: /)
5) exp_xpath: expenditure xpath (e.g.: //nmc-expenditure)
6) rev_xpath: revenue xpath (e.g.: //nmc-revenue)

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class"
value="com.softwareag.belgium.xef.impl.filters.CheckOutOJAndFilter"/>
<reuse-param-list name="publications">
<reuse-param name="publication" value="BUDAP2007VOL2"/>
<reuse-param name="publication" value="BUDAP2007VOL5"/>
</reuse-param-list>
<reuse-param name="section_name" value="nmc-section"/>
<reuse-param name="xpath" value="/"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.vl0.LinkAndReuse.ContributionStatement"/>
<reuse-param name="rev_xpath" value="//nmc-revenue"/>
<reuse-param name="exp_xpath" value="//nmc-expenditure"/>
</reuse-link>
```

5.4.3.9. Include an abb document

Use: in “Estimate of Revenue and Expenditure of the European Parliament” report

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.abb.Abb2Vl0`

Parameters:

1) publication: Name of the (abb-like) document to import (e.g.: the Estimate document of Parliament BUDAP2007VOL2)
2) xpath: xpath of the document fragment to import (e.g.: /)

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="publication" value="BUDAP2007VOL2"/>
<reuse-param name="xpath" value="/"/>
<reuse-param name="filter" value="com.softwareag.belgium.seibud.abb.Abb2Vl0"/>
</reuse-link>
```

5.4.3.10. Include “Track changes” between abb documents

Use: in “Estimate of Revenue and Expenditure of the European Parliament” report

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutDiffAndFormat`

Diffformatter name:`com.softwareag.belgium.seibud.abb.AbbVl0DiffFormatter`

Parameters:

1) publication: name of the abb-like document (e.g.: the Estimate document of Parliament BUDAP2007VOL2)

2) oldpublication: name of the abb-like document to compare with (e.g.: the Budget document of Parliament BUDD2006VOL2)

3) version: The specific version (label or number) of the abb-like document (default= current)

4) oldversion: The specific version (label or number) of the old abb-like document (default=current)

5) xpath: xpath of document fragment to compare (e.g.: /)

6) showDataDiff: Display changes within figures: yes/no

7) showIdentical: Display identical parts: yes/no

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class"
value="com.softwareag.belgium.xef.impl.filters.CheckOutDiffAndFormat"/>
<reuse-param name="publication" value="BUDAP2007VOL2"/>
<reuse-param name="oldpublication" value="BUDD2006VOL2"/>
<reuse-param name="xpath" value="/"/>
<reuse-param name="diffformatter"
value="com.softwareag.belgium.seibud.abb.AbbVl0DiffFormatter"/>
<reuse-param name="showDataDiff" value="yes"/>
<reuse-param name="showIdentical" value="no"/>
</reuse-link>
```

5.4.4. REUSE library for ABB publication

5.4.4.1. Summary table of the amount of a "horizontal" item (XX …) by title (policy)

Use: in items in Chapter XX 01 of Volume 4.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.abb.LinkAndReuse.PoliciesAmounts`

Parameters:

1) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

2) alias: item alias (e.g. XX 01 01 01)

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.abb.LinkAndReuse.PoliciesAmounts"/>
<reuse-param-list name="aliases"><reuse-param name="alias" value="XX 01 01 01"/>
</reuse-param-list>
</reuse-link>
```

5.4.4.2. Summary table of the human resources of a "horizontal" item (XX …) by title (policy)

Use: in items in Chapter XX 01 of Volume 4.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.abb.LinkAndReuse.PoliciesHResources`

Parameters:

1) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

2) alias: item alias (e.g. XX 01 01 01)

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.abb.LinkAndReuse.PoliciesHResources"/>
<reuse-param-list name="aliases">
<reuse-param name="alias" value="XX 01 01 01"/>
</reuse-param-list>
</reuse-link>
```

5.4.4.3. Breakdown of reserves of non-differentiated appropriations

Use: in Title “Reserve” of Volume 4.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.abb.LinkAndReuse.NDiffReservesBreakdown`

Parameters:

1) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

2) alias: alias of the destination (e.g. 31 02 40 01)

For example:

```
<reuse-link reuse="reuse-link">
```

```
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.abb.LinkAndReuse.NDiffReservesBreakdown"/>
<reuse-param-list name="aliases">
<reuse-param name="alias" value="31 02 40 01"/>
</reuse-param-list>
</reuse-link>
```

5.4.4.4. Breakdown of reserves of differentiated appropriations

Use: in Title "Reserve" of Volume 4.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.abb.LinkAndReuse.DiffReservesBreakdown`

Parameters:

1) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

2) alias: alias of the destination (e.g. 31 02 41 01)

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.abb.LinkAndReuse.DiffReservesBreakdown"/>
<reuse-param-list name="aliases">
<reuse-param name="alias" value="31 02 41 01"/>
</reuse-param-list>
</reuse-link>
```

5.4.4.5. Enlargement table of expenditure line by line.

Use: in the Enlargement annex of Volume 4.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.abb.LinkAndReuse.EnlargementLineByLine`

Parameters:

1) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.abb.LinkAndReuse.EnlargementLineByLine"/>
</reuse-link>
```

5.4.4.6. Summary table of Heading 5 of the financial perspectives.

Use: in the Heading V annex to Volume 4.

Linkresolver:`com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter`

Filter name:`com.softwareag.belgium.seibud.abb.LinkAndReuse.CategoryAmounts`

Parameters:

1) xpath: expenditure xpath (e.g.:/abb[1]/nmc-section[1]/nmc-expenditure[1])

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="class" value="com.softwareag.belgium.xef.impl.filters.CheckOutAndFilter"/>
<reuse-param name="xpath" value="/abb[1]/nmc-section[1]/nmc-expenditure[1]"/>
<reuse-param name="filter"
value="com.softwareag.belgium.seibud.abb.LinkAndReuse.CategoryAmounts"/>
</reuse-link>
```

5.4.5. REUSE library for AMD CAT/POL reports

5.4.5.1. Principles of computation of the sums of the Amendments figures

To compute the AMD CAT/POL amounts, the following 4 steps are executed:

1) The integration of the figures from the budget lines and from the amendments.

2) The application of corrections on the integration results.

3) The computation of the sums of the integration results.

4) The generation of the CAT/POL report using the computed sums and a template document.

Below, the two last steps are detailed.

5.4.5.2. The computation of the sums of the integration results

The computation of the sums depends on the following selections:

— The type of reading: Council L1, EP L1, Council L2 or EP L2. This type allows to select the budget types of the computed sums.

— The ABB VOL4 publication. The computation uses only the metadata part of this publication and extracts data about the margins by categories/politics and about the gross national incomes.

The integration results can be represented as a big table that stores the figures of the following different types of reading and their corresponding figures by years and by types of budget (we suppose here that we prepare the budget of 2007) :

— Council L1 (FIGURES_BEFORE_PARLIAMENT):
  — 2006: budget of current year
  — PDB2007: preliminary draft of current year
  — DB2007: amended figures of current year
— EP L1 (FIGURES_BEFORE_COUNCIL):
  — 2006: budget of current year
  — PDB2007: preliminary draft of current year
  — DB2007: draft of current year

    - PEL2007: amended figures of current year
- Council L2 (FIGURES_AT_COUNCIL):
    - 2006: budget of current year
    - DB2007: draft of current year
    - PEL2007: PEL1 figures of current year
    - COL2 2007: amended figures of current year
- EP L2 (FIGURES_AFTER_COUNCIL):
    - 2006: budget of current year
    - PEL2007: PEL1 figures of current year
    - COL2 2007: COL2 figures of current year
    - PEL2 2007: amended figures of current year

Each figure of this big table contains the following data:

- The type of figures (COMP or NCOMP)
- The commitment figure (com)
- The payment figure (pay)

Each computed sum is identified by an unique figure name. This name is structured as a concatenation of different sub strings separated by '_'.

Below, we give some figure name examples:

- 'exp_sum_dbnm1_pay_5' : in the politics '5', sums up all the payments (sub string 'pay') of the draft of previous year (sub string 'dbnm1')
- 'exp_diff_pel1-dbn_com_4' : in politics '4', subtracts the commitments figure 'exp_sum_pel1_com_4' - 'exp_sum_dbn_com_4'

All the computed sums can be classified as follow:

- The maximum commitments by total, by category or by category/politics. The figure name has one of these formats:
    - exp_max_<BUDGET>_com
    - exp_max_<BUDGET>_com_<CATPOL>
- The basic sums which sums up the figures by the category/politics. The figure name has one of these formats:
    - exp_sum_< BUDGET >_<CP>
    - exp_sum_< BUDGET >_<CP>_<CATPOL>
    - exp_sum_< BUDGET >_<CP>_<CATPOL>_<ALIAS_1>
    - exp_sum_< BUDGET >_<CP>_<CATPOL>_< ALIAS _1>_< ALIAS_2>
    - exp_sum_< BUDGET >_<CP>_< COMPNCOMP>
- The difference between the basic sums. The figure name has one of these formats:

- exp_diff_<BUDGET (left operand)>-<BUDGET (right operand)>_<CP>…

— The margin commitments by total, by category or by category/politics The figure name has one of these formats:

- exp_marg_< BUDGET >_com
- exp_marg_< BUDGET >_com_<CATPOL>

The sub string < BUDGET> has one of theses values:

— pdbnm1: preliminary draft budget of the previous year

— dbnm1: draft budget of the previous year

— bnm1: budget of the previous year

— pdbn: preliminary draft budget of the current year

— dbn: draft budget of the current year

— pel1: PEL1 figures of the current year

— col2: COL2 figures of the current year

— pel2: PEL2 figures of the current year

The sub string < CP> has one of theses values:

— com: commitments

— pay: payments

The sub string <CATPOL> contains the politics code. A dot at the end of the string means: the partial sums on the code will only be computed for the parts separated by dots and not for parts built of arbitrary substrings.

The politics code is a concatenation of numbers separated by the '.' character like this:

— <n1>. … .<n_m>.<n_m+1>

When a figure is treated with a CAT/POL like '<n1>. … .<n_m>.<n_m+1>', its payments and its commitments are sum up on all the following ancestor CAT/POL code :

— <n1>. … .<n_m>.<n_m+1>

— <n1>. … .<n_m>

— …

— <n1>

5.4.5.3. The generation of the CAT/POL report using the sum results and an template document

This template can use two types of re-use:

1) ComputeValue: extract the value given by its figure name.

2) ComputeDifference: extract two values given by their figure names and subtract them.

The re-use ComputeValue uses the following Linkresolver:

— com.softwareag.belgium.seibud.amd.LinkAndReuse.ComputeValue

The re-use ComputeValue needs the following parameters:

— figname: gives the figure name

— emp: is the emphasis. The parameter is optional. The possible values are:

  — bold or italic

— format: The parameter is optional. The possible values are:

  — M: the format is "###,###,###,###,###,###,###,###,###,##0" and the value is divided by 1000000.

  — M2: the format is "###,###,###,###,###,###,###,###,###,##0.00" and the value is divided by 1000000.

  — P: the format is "##0.0" and the value is multiplied by 100.

  — P2: the format is "##0.00" and the value is multiplied by 100.

The re-use ComputeDifference uses the following Linkresolver:

— com.softwareag.belgium.seibud.amd.LinkAndReuse.ComputeDifference

The re-use ComputeDifference needs the following parameters:

— figname1: gives the figure name of the left operand of the subtraction.

— figname2: gives the figure name of the right operand of the subtraction.

— emp: is the same parameter as ComputeValue.

— format: is the same parameter as ComputeValue.

For example:

```
<reuse-link reuse="reuse-link">
<reuse-param name="figname" value="exp_ratio_pdbn-gni_perc_pay"/>
<reuse-param name="format" value="P2"/>
<reuse-param name="emp" value="italic"/>
<reuse-param name="class"
value="com.softwareag.belgium.seibud.amd.LinkAndReuse.ComputeValue"/>
</reuse-link>
<reuse-link reuse="reuse-link">
<reuse-param name="figname1" value="exp_sum_dbnm1_com_4"/>
<reuse-param name="figname2" value="exp_sum_dbnm1_com_4.0.9"/>
<reuse-param name="emp" value="bold"/>
<reuse-param name="class"
value="com.softwareag.belgium.seibud.amd.LinkAndReuse.ComputeDifference"/>
</reuse-link>
</entry>
```

5.5. The tables used by the SEI-BUD server

In addition to the tables used by the XML repository, the following tables are created by the scripts"logdb_oracle.sql" and "logdbinit.sql" (see the "Installation Guide (GIL)"):

TLOG_DATATYPE

TLOG_MSGCONTENT

TLOG_INFOTYPE

TLOG_INFOTYPECHOICE

TLOG_RQST

TLOG_INFORQST

TLOG_LEVEL

TLOG_MSG

TLOG_INFOMSG

Once the server has started up, other tables are created:

RMG_ABBDOCUMENTS

RMG_ABBDOCUMENTLOCKS

RMG_POETRYIDENTIFIER

XEF_LOCK

XEF_STATEDB

XEF_VERSION_BRANCH

XEF_VERSION_LABEL

XMLDOCUMENT

XMLINSTANCE

PR_PROCESSINGREPORTS

PR_ATTACHEDRESULTS

5.5.1. Logging tables

5.5.1.1. The table TLOG_DATATYPE

This table gives the different types of data used in logged values (for example:`tstring`, `tnumeric`,`tdate`,`tchoice`, which are data types mentioned in the XOo f specifications).

Name	Type	Remarks
datatype	varchar2	name of a data type

5.5.1.2. The table TLOG_INFOTYPE

This table lists the different attributes for which values are stored: each attribute has a type defined in the table TLOG_DATATYPE (examples of attribute names:`class-name`,`method-name`,`document-name`).

Name	Type	Remarks
name	varchar2	(information) attribute name
descr	varchar2	description of this attribute
datatype	ref	reference to the datatype column of the table TLOG_DATATYPE

5.5.1.3. The table TLOG_INFOTYPECHOICE

This table lists the various possible values for an attribute name.

Name	Type	Remarks

name	ref	reference to the name column of the table TLOG_INFOTYPE.
val	varchar2	possible value for a choice of values

5.5.1.4. The table TLOG_RQST

This table contains one row per request; each row provides a request identifier and the date on which that request was entered into the SEI-BUD system.

Name	Type	Remarks
id	varchar2	request identifier (primary key)
ddate	varchar2	request date

5.5.1.5. The table TLOG_INFORQST

This table implements a dictionary of information concerning requests to the SEI-BUD system.

Name	Type	Remarks
name	Ref	reference to the name column of the table TLOG_INFOTYPE
val	varchar2	request attribute value
idrqst	Ref	reference to the identifier column of the table TLOG_RQST

5.5.1.6. The table TLOG_LEVEL

This table gives the different log levels, along with a description and a colour (unused in the SEI-BUD system). For example, for SEI-BUD, the following levels exist: "`Debug`", "`Info`", "`Warning`", "`User Error`", "`Application Error`", "`System Error`".

Name	Type	Remarks
lvl	number	log level (primary key)
descr	varchar2	description of log level
color	number	colour code

5.5.1.7. The table TLOG_MSG

This table contains one entry per logged message.

Name	Type	Remarks
id	varchar2	primary key
idrqst	Ref	reference to the identifier column of the table TLOG_RQST
ddate	date	message date
lvl	Ref	reference to the lvl column of the table TLOG_LEVEL
text	varchar2	message text
hascontent	number	boolean value (0 or 1) indicating whether the message has content

5.5.1.8. The table TLOG_INFOMSG

This table can contain additional information in the form of attributes for each message (in the same way as for requests); however, the table is not used in SEI-BUD at present.

Name	Type	Remarks
name	Ref	reference to the name column of the table TLOG_INFOTYPE
val	varchar2	value
idmsg	Ref	reference to the identifier column of the table TLOG_MSG

5.5.1.9. The table TLOG_MSGCONTENT

This table gives the contents of attachments associated with a message; there is a maximum of one attachment per message. For example, for an error message, this table could contain a detailed diagnosis of the context of the error in the form of a stack trace.

Name	Type	Remarks
idmsg	Ref	reference to the identifier column of the table TLOG_MSG
mime	varchar2	type of content for the "content" value
content	long raw	binary content
compressed	number	boolean value (o or 1) indicating whether the content of the "content" column is compressed.

5.5.2. The RMG tables

5.5.2.1. The table RMG_ABBDOCUMENTS

This table gives all the publications that can be managed by the SEI-BUD system.

Name	Type	Remarks
objid	varchar2	unique publication identifier (primary key)
state	varchar2	publication state
name	varchar2	publication name
type	varchar2	type of publication (e.g. ABB)
year	varchar2	year of publication
step	varchar2	stage of publication
volume	varchar2	publication volume
instance	varchar2	instance name (n/a)

5.5.2.2. The table RMG_ABBDOCUMENTLOCKS

This table gives all the active locks for all the publications that can be managed by the SEI-BUD system. These locks correspond to reservations. Access to this table is managed by the class "`AbbDocumentLock`".

Name	Type	Remarks
objid	varchar2	unique publication identifier (primary key)
view_	varchar2	view in which the reservation was made
language	varchar2	reservation language
fragmentid	varchar2	fragment identifier (versioned Xpath)
alias	varchar2	fragment alias
locktoken	varchar2	lock identifier (based on the request identifier)
userid	varchar2	username
datetime	date	date and time the lock was obtained

5.5.2.3. The table RMG_POETRYIDENTIFIER

This table gives the information needed to manage document fragments sent for translation via the Poetry system. This table is managed by the class "`PoetryIdentifier`".

Name	Type	Remarks
documentname	varchar2	document name
language	varchar2	language (part of primary key)
fragmentid	varchar2	fragment identifier (versioned Xpath)
poetryrefnum	number	Poetry reference number (part of primary key)
poetryversionnum	number	fragment version number (part of primary key)
poetrypartnum	number	fragment part number (part of primary key)

5.5.2.4. The table PR_PROCESSINGREPORTS

This table gives the processing report information for all the publications managed by the SEI-BUD system.

Name	Type	Remarks
processingreportid	varchar2	unique processing report identifier (primary key)
state	varchar2	processing report state

subject	varchar2	name of the method responsible for the creation of the processing report (e.g.:`checkin_syncwrapper`)
target	varchar2	name of the user who will receive the processing report
datetime	date	date and time of creation
body	varchar2	not used

5.5.2.5. The table PR_ATTACHEDRESULTS

This table gives the attachments for all processing reports.

Name	Type	Remarks
attachedresultid	varchar2	unique attachment identifier (primary key)
processingreportid	varchar2	unique processing report identifier
name	varchar2	attachment name (e.g.: result, error...)
mime	varchar2	mime type (e.g. text/xml)
data	long raw	binary content

5.5.3. The XEF tables

5.5.3.1. The table XEF_LOCK

This table is used by the class "`Lock`" of the package "`com.softwareag.belgium.xef.impl.repository`" to manage the persistence, insertion, removal and selection of locks on parts of publications.

Name	Type	Remarks
token	varchar2	unique lock identifier (primary key)
documentname	varchar2	publication name
branchname	varchar2	branch name
xpath	varchar2	versioned XPath
viewname	varchar2	view name
language	varchar2	language
lmode	number	lock mode (digital value corresponding, for example, to the modes "author", "translator", etc.)
status	number	lock status ("active", "commit")

5.5.3.2. The table XEF_STATEDB

This table gives the information needed to manage the states of a publication fragment (the state of a fragment depends on the state of its branches). This table is used by the class "`StateDb`" of the package "`com.softwareag.belgium.xef.impl.repository.state`".

Name	Type	Remarks
documentname	varchar2	publication name
branch	varchar2	branch name
xpath	varchar2	versioned XPath
state	varchar2	name of state (e.g.: "`authoring`", "`correction`", "`translation`"

5.5.3.3. The table XEF_VERSION_BRANCH

This table gives the processing report information for all the publications managed by the SEI-BUD system.

Name	Type	Remarks
document	varchar2	publication name
branch	varchar2	branch name
lifecycle	varchar2	name of a lifecycle management class (e.g.: "`com.softwareag.belgium.seibud.rmg.bo.AbbLifeCycle`")

5.5.3.4. The table XEF_VERSION_LABEL

This table gives the labels that can be used as a mnemonic designation for a particular version of part of a publication. This table is managed by the class "`VersionLabel`" of the package "`com.softwareag.belgium.xef.impl.repository`".

Name	Type	Remarks
label	varchar2	label name
documentname	varchar2	publication name
branch	varchar2	branch name
xpath	varchar2	versioned XPath
sitdbversionmap	varchar2	string of characters representing a dictionary giving a sitdb version according to a language code

5.5.3.5. The table XMLDOCUMENT

This table gives all the publications managed by the SEI-BUD system. This table is used by the class "`XmlDocument`" of the package "`com.softwareag.belgium.xef.impl.repository`".

Name	Type	Remarks
name	varchar2	publication name
class	varchar2	publication class (e.g.: "abb")

5.5.3.6. The table XMLINSTANCE

This table gives all the different language versions for each publication managed by the SEI-BUD system. This table is used by the class "`XmlDocument`" of the package "`com.softwareag.belgium.xef.impl.repository`". It can be used to find out the XML Repository instance name from the publication name and language code.

Name	Type	Remarks
document	varchar2	publication name
language	varchar2	language codes
name	varchar2	XML repository instance name

5.5.4. The tables used for Amendments

See annex : annex-MRF-tabAmd

5.6. The DTDs

5.6.1. Introduction

The SEI-BUD system implements "reservation/update" (or "consultation") cycles in different formats. The SEI-BUD system offers the following editing formats:

For editing the budget (ABB DTD):

1) XML (structure view) for editing the budget nomenclature

2) XML (figures view) for editing budget figures

3) RTF (remarks view) for the Word-based editing tool used by authors and translators

4) XML (remarks view) for the Adept Editor editing tool used by correctors at the Publications Office

5) HTML for printing out the differences.

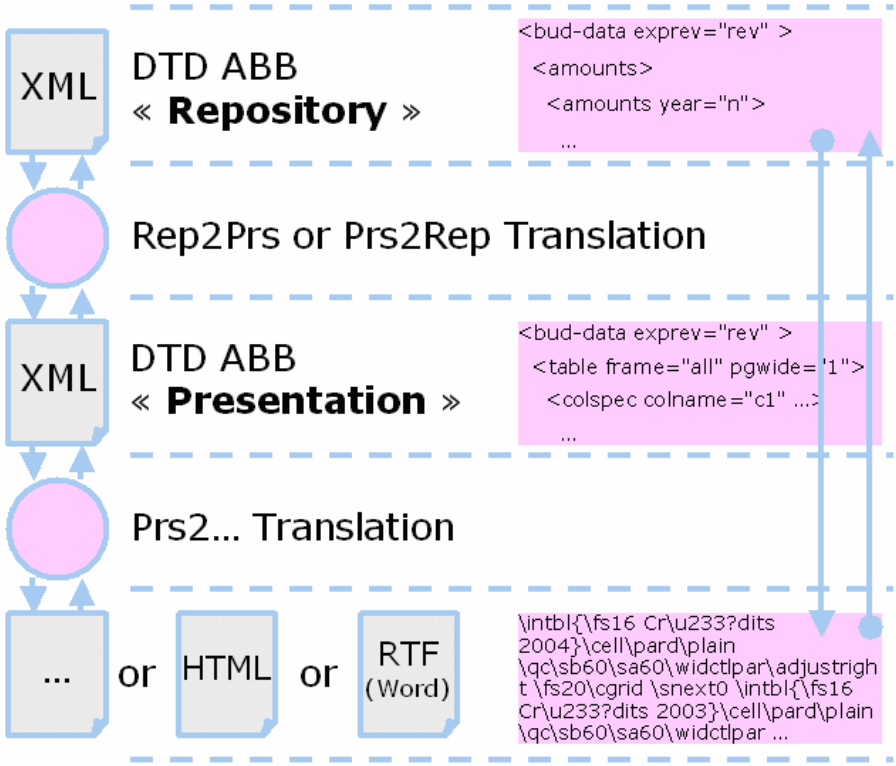
For editing the Budget Estimate and Volume 0 (VL0 DTD):

— RTF (remarks view) for the Word-based editing tool used by authors and translators

— XML (remarks view) for the Adept Editor editing tool used by correctors at the Publications Office

— HTML for printing out the differences.

In order to rationalise conversions between the repository format (ABB DTD XML) and the proprietary or specific formats such as Word, the SEI-BUD system implements a strategy that involves using a "pivot" format, presentation DTD, as the intermediate format.



This strategy involves putting together in an initial conversion phase all the conversions relating to the "presentation" of the information needed for the production of any "finished product". For example, irrespective of the finished product (Word, HTML, etc.), the budget figures are always "presented" in the form of a budget table with 3, 4, 5 or 6 columns. Conversion of the budget figures' own semantic (<bud-data> element) into a CALS table (<table> element) is carried out on conversion to the "presentation" format.

There are very few differences between the "Repository" and "Presentation" DTDs. That is why these two DTDs are actually "defined" in a single module.

5.6.2. The ABB DTD

5.6.2.1. General structure and organisation

The ABB DTD is organised (structured) in the following way:

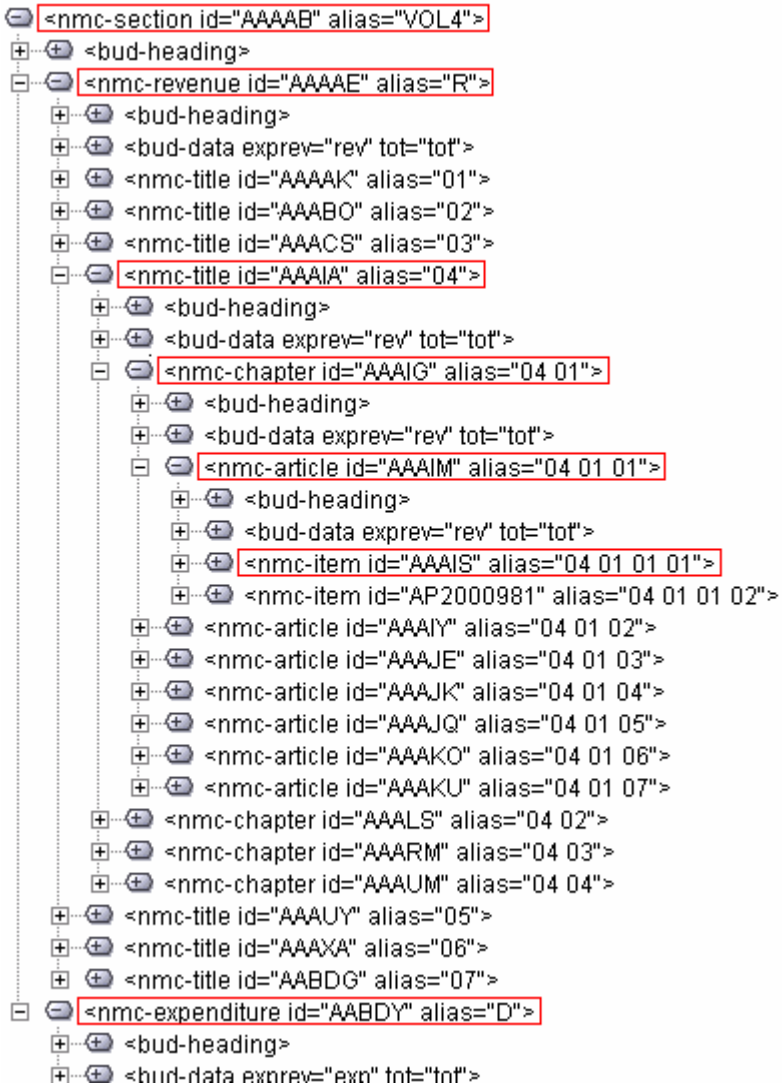
abb.dtd	"main"
abb-nmc.mod	Contains the definitions relating to the general structure of a budget document (revenues, expenditures, titles, chapters, articles, items, etc.) and the content of these structural elements (figures, remarks, legal basis, reference acts, etc.).

abb-fig.mod	Contains the definitions relating to the structure of the budget figures (figures part, schedules, human resources).
abb-btx.mod	Contains the definitions relating to the basic text, such as the definition of paragraphs, lists, etc.
abb-tbl.mod	Contains the definitions relating to the table model. That definition "includes" the DTD relating to the CALS table models (calstblx.dtd).
calstblx.dtd	DTD based on the CALS table model.

5.6.2.2. Budget nomenclature

General structure of a budget document

Basically, the general structure of a budget document looks like this:



Generally, a budget document is therefore organised into a "revenues" section, an "expenditures" section, "budgetary" annexes (the offices) and "basic text" type annexes. The revenues and expenditures sections are themselves organised into titles containing chapters containing articles containing items.

The structure of each budget line is as follows:

— The heading (bud-heading element) of the budget line;

— An introduction, if applicable (bud-intro element);

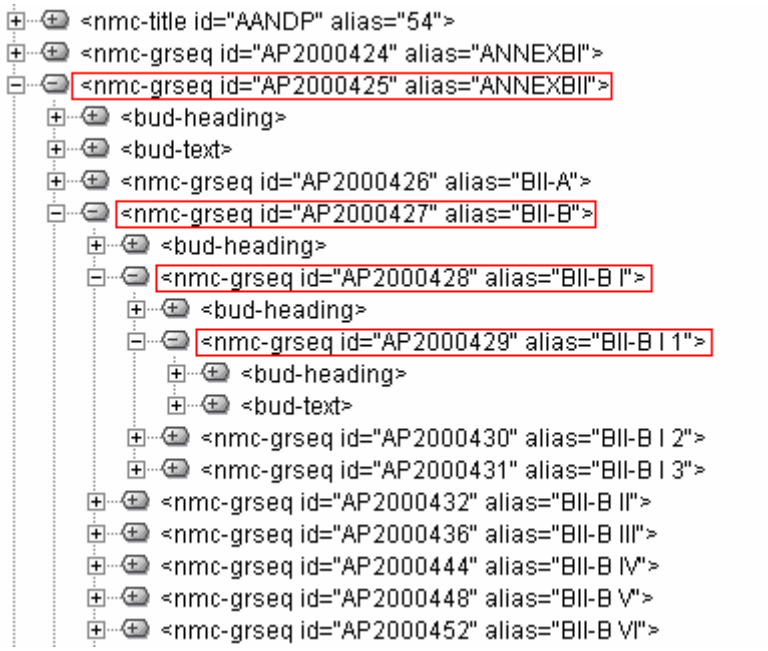
— The budget figures (bud-data element);

— Any budget remarks (bud-remarks element);

— Any legal basis (bud-legal element);

— Any reference acts (bud-reference element).

The structure of an annex (nmc-grseq element) is in basic text; it is organised using a hierarchy of nmc-grseq elements.



Specific case of Volume 1

In order to ensure coverage of Volume 1 (General Statement of Revenues) as well (see also Volume 9), an element <nmc-sectpart> is added to the DTD. The model for the root element of a budget document is therefore:

```
<!ELEMENT nmc-section (bud-heading, bud-intro?, bud-remarks?, bud-legal?, (nmc-sectpart+ |
(nmc-revenue?, nmc-expenditure?)), (%appendix.class;)*)>
<!ATTLIST nmc-section
%ext.nomenc.unit.attrib;
>
```

The attributes of elements of the budget nomenclature

The attributes of elements of the budget nomenclature are:

For the "Repository" DTD:

```
<!ENTITY % ext.nomenc.unit.attrib "
alias CDATA #IMPLIED
id ID #REQUIRED
state (hidden) #IMPLIED"
>
```

For the "Presentation" DTD:

```
<!ENTITY % ext.nomenc.unit.attrib "
alias CDATA #IMPLIED
id ID #IMPLIED
state (hidden) #IMPLIED"
>
```

Attribute	Description
alias	The alias of a budget line (apart from the elements <nmc-grseq> and <nmc-annex>) is the identifier of a budget line (e.g.: "02 01 03 04"). This information should normally be mandatory ("REQUIRED"). In the DTD it is defined as being "IMPLIED", in order to allow a certain amount of flexibility when a budgetary instance is being created. For elements that are not strictly "budget lines" (nmc-sectpart, nmc-grseq, nmc-annex), the attribute "alias" acts as a "prefix" to the element heading.
id	This attribute is an "internal" identifier within the SEI-BUD system. This information should not normally be used and/or modified by any tool or player outside the system. This is why the attribute is "REQUIRED" for the "Repository" DTD and "IMPLIED" for the "Presentation" DTD.
state	This attribute is not used at present. This attribute was carried over from the BUD DTD of version 3 of the SEI-BUD system. It was used to "activate" or "deactivate" a budget line within the framework of the production of SABs (Supplementary and Amending Budgets) and ALs (Amending Letters). At present, SEI-BUD v4 does not cover the production of SABs and ALs. We envisage treating SABs and ALs as amendments to a publication; these publications would therefore be handled within the framework of the Unified system.

5.6.2.3. The "content" of a budget line

As stated above, the structure of each budget line is as follows:

1) The heading (bud-heading element) of the budget line;

2) An introduction, if applicable (bud-intro element);

3) The budget figures (bud-data element);

4) (the budget figures for enlargement (bud-data-extra element));

5) Any budget remarks (bud-remarks element);

6) Any legal basis (bud-legal element);

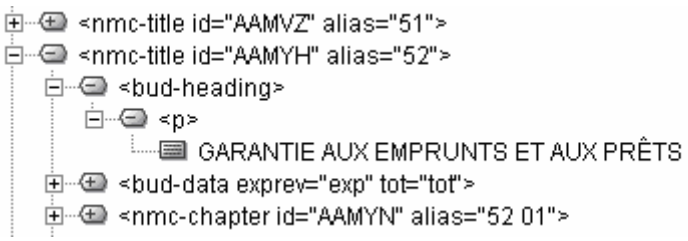
7) Any reference acts (bud-reference element).

The structure of an annex (nmc-grseq element) is in basic text; it is organised using a hierarchy of nmc-grseq elements.

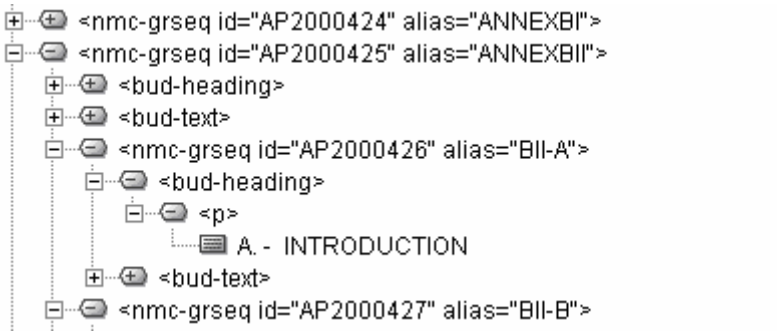
The heading <bud-heading>

The element <bud-heading> is the element used for structuring headings of elements in the budget nomenclature, as for the "titles" of the basic text annexes. The model of a heading is "p" (one single "paragraph").

Example of the structure of the heading of a budget line:



Example of the structure of the heading of a title in a "basic text" annex:



The introduction <bud-intro>

The element <bud-intro> is specific to the ABB. This element is used to structure a portion of text that follows on immediately from the heading of a budget line and precedes the portion reserved for budget figures.

Initially, only the "expenditure headings" that in the ABB correspond to "policies" should allow the introduction of this type of text (brief description of the "policy").

However, the DTD allows this type of text to be inserted into all elements of the budget nomenclature.

The model for this element is "basic text"; its content may therefore consist of "paragraphs", "lists" and "tables". In practice, its content will be one or possibly more paragraphs.

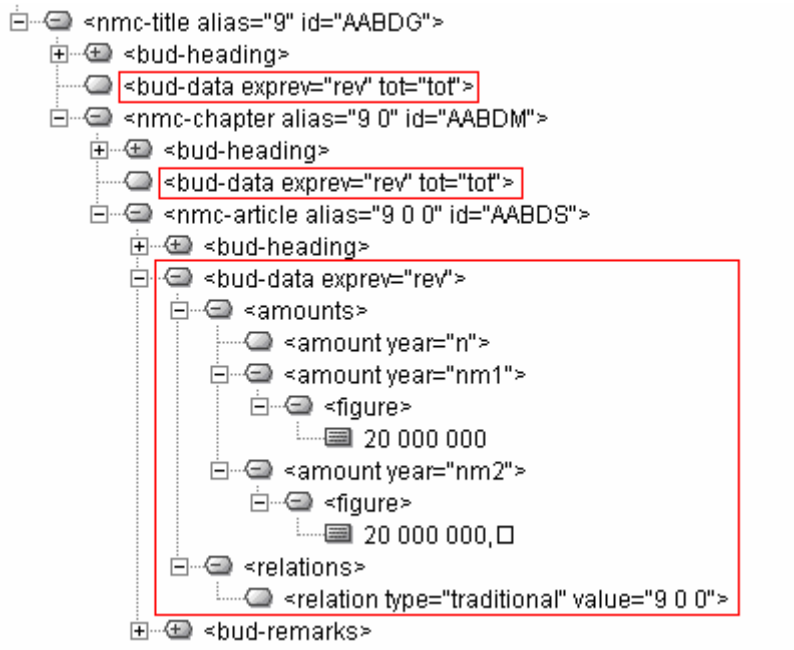
The budget figures <bud-data>

The element <bud-data> is used to contain the figures in budget lines.

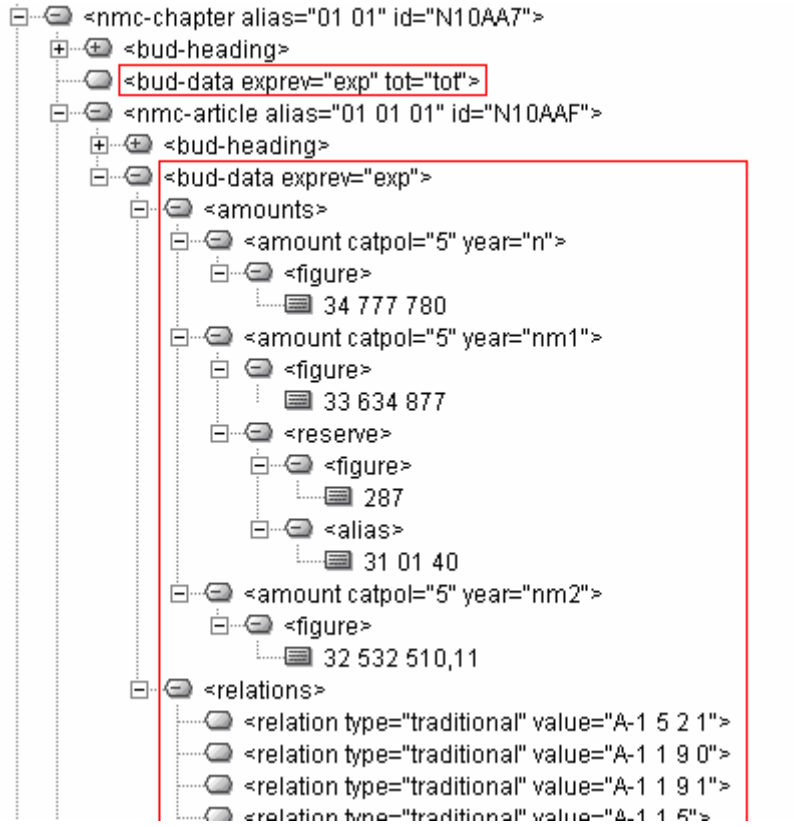
Generally, the element <bud-data> is**mandatory**in all "purely" budget lines: nmc-revenue, nmc-expenditure, nmc-title, nmc-chapter, nmc-article, nmc-item. The attributes of the element <bud-data> are:

Attribute	Description
exprev	The attribute "exprev" is mandatory; it is used to specify whether a budget line is part of the "revenues" section or the "expenditures" section of the budget. N.B.: Obviously all <bud-data> in all "child" budget lines in the "revenues" section will have an "exprev=rev" attribute, while the <bud-data> in all "child" budget lines in the "expenditures" section will have an "exprev=exp" attribute. This repetition is necessary. The repository allows access to any fragment of a budget publication. It is therefore essential to be able to determine for all budget lines whether they should be treated as "revenues" lines or "expenditures" lines.
tot	Generally, a <bud-data> element of the type "total" (tot=tot) will be empty; the exception to this is <bud-data> from the "expenditure titles" (nmc-title), which will contain the figures for human resources <hresources>. Really, the attribute "tot" is slightly redundant. The value of the attribute can be determined by looking at whether or not the budget line contains "child" budget lines.
horizontal	This attribute (horizontal=true) should be used to define the "horizontal" budget figures, i.e. lines (sub-items) used in Chapter XX 01.

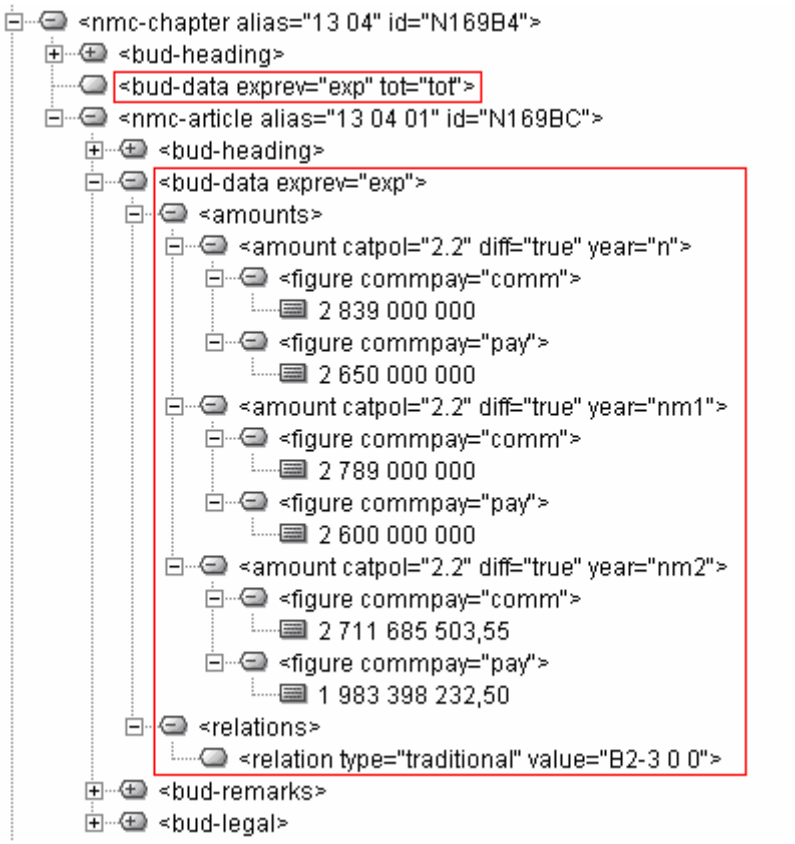
Example of <bud-data> for the "revenues" section:



Example of <bud-data> for the "expenditures", "non-differentiated appropriations" section:



Example of <bud-data> for the "expenditures", "differentiated appropriations" section:



The model for the element <bud-data> is as follows:

```
<!ELEMENT bud-data (amounts?, hresources?, schedules?, relations?)>
<!ATTLIST bud-data
exprev (exp|rev) #IMPLIED
tot (tot) #IMPLIED
horizontal (true) #IMPLIED
>
```

This element is therefore made up of:

1) The element <amounts>: this contains all the figures;

2) The element <hresources>: this contains all the data relating to human resources;

3) The element <schedules>: this contains all the data relating to budgetary schedules;

4) The element <relations>: this contains various information regarding the construction of Chapter XX 01 that makes it possible to link a budget line in the ABB with a budget line in the traditional version of the budget.

The element <amounts>

The model for the element <amounts> is as follows:

```
<!ELEMENT amounts (amount)* >
<!ELEMENT amount (figure+, reserve?)? >
<!ATTLIST amount
aele (true) #IMPLIED
catpol CDATA #IMPLIED
comp (true) #IMPLIED
delegation (true) #IMPLIED
```

```
diff (true) #IMPLIED
peco (true) #IMPLIED
year (n | nm1 | nm2 | delta | new) #IMPLIED
>
<!ELEMENT figure (#PCDATA) >
<!ATTLIST figure
commpay (comm | pay) #IMPLIED
>
```

An <amounts> element is therefore made up of several <amount> elements. In fact, an <amounts> element consists of three <amount> elements:

1) one for the year N (year="n"),

2) one for the year N-1 (year="nm1"),

3) and one for the year N-2 (year="nm2"),

(N.B.: currently, the attributes "delta" and "new" are not used.)

Each <amount> element for the year N or N-1 or N-2 could then be made up of:

For "revenues": one <figure> element giving the amount for the financial year (year N or N-1 or N-2).

For "expenditures":

1) In the case of non-differentiated appropriations: one <figure> element (giving the appropriation for the year N or the appropriation for the year N-1 or the execution for the year N-2).

2) In the case of differentiated appropriations: two <figures> elements (giving the amounts committed (commpay=comp) and paid (commpay=pay) of the appropriations for the year N or of the appropriations for the year N-1 or the execution of the year N-2).

To summarise, attributes of the element <amount> should be used as follows:

Attribute	Description
year	Used to define the year to which the element <amount> relates ("n" for the year N, "nm1" for the year N-1 and "nm2" for the year N-2; N is the current financial year). The attribute values "delta" and "new" are not used.
diff	Diff=true means that the element <amount> for the year referred to by the attribute "year" is of the type "differentiated appropriations". For revenues, this attribute cannot be used; for expenditures, it is considered by default that budget figures are of the type "non-differentiated appropriations".
comp	This attribute (compulsory) is used to define, (for expenditure lines,) whether the appropriation in question (for the year N or N-1 or N-2) is "compulsory expenditure" or "non-compulsory expenditure" (CE/NCE). The default is "non-compulsory". The aim is to use this information for the automatic production of what is known as the "CE/NCE Annex".
catpol	This attribute (category.policy) is used to produce other types of "figures" reports (presentation by "categories and policies" or financial perspectives).
aele	Not used at present. The aim is to use this information for the automatic production of what is known as the "AELE Annex".
peco	Not used at present. The aim is to use this information for the automatic production of what is known as the "PECO Annex".
delegation	No longer used at present (was used for the 2003 ABB).

The element <hresources>

The model for the element <hresources> is as follows:

```
<!ELEMENT hresources (hresource)* >
```

```
<!ELEMENT hresource (figure, relations?)? >
<!ATTLIST hresource
type (operation | research | support | othersupport | linguistic | tot | epstat | policy |
pending) "operation"
year (n | nm1 | nm2 | delta | new) "n"
>
```

This element is used to define human resources by "policy".

The attributes have the following meanings:

type:

operation = Establishment plan staff: operational

research = Establishment plan staff: research

support = Support staff ex-A-7

othersupport = Support staff other

linguistic = Linguistic services

tot = total

year:

n = year N

nm1 = year N-1

nm2 = year N-2

(delta = sab/al amount, not used at present)

(new = new amount)

The element <schedules>

The model for the element <schedule> is as follows:

```
<!ELEMENT schedules (schedule)* >
<!ELEMENT schedule (figure, note*)? >
<!ATTLIST schedule
type (cso | cma | nm1 | n | tot) #IMPLIED
year (nm1 | n | np1 |np2 | npx) #IMPLIED
>
```

This element is used to define the budgetary schedules (Volume 4 expenditures).

The element <relations>

The model for the element <relations> is as follows:

```
<!ELEMENT relations (relation)* >
<!ELEMENT relation EMPTY >
<!ATTLIST relation
pp CDATA #IMPLIED
type (admin | traditional | policy) "admin"
```

```
value CDATA #IMPLIED
>
```

The logic for using the attributes is as follows:

If type = "traditional":

Then value = alias in the traditional nomenclature

If type = "admin":

Then value =

XX 01 01 01 staff-institution (remuneration, transfers)

XX 01 01 02 staff-delegation (remuneration, transfers)

XX 01 02 01 external-institution (auxiliary, agency, temporary)

XX 01 02 02 external-delegation (local, training, other)

XX 01 02 11 other-institution (interpretation, translation, mission, conference, meetings, studies, training)

XX 01 02 12 other-delegation (mission, training)

XX 01 03 01 buildings-institution (acquisition, other, equipment, services)

XX 01 03 02 buildings-delegation (acquisition, equipment)

XX 01 05 01 staff-research

XX 01 05 02 external-research

XX 01 05 03 other-research

To summarise, this element is used to define certain relationships, in particular:

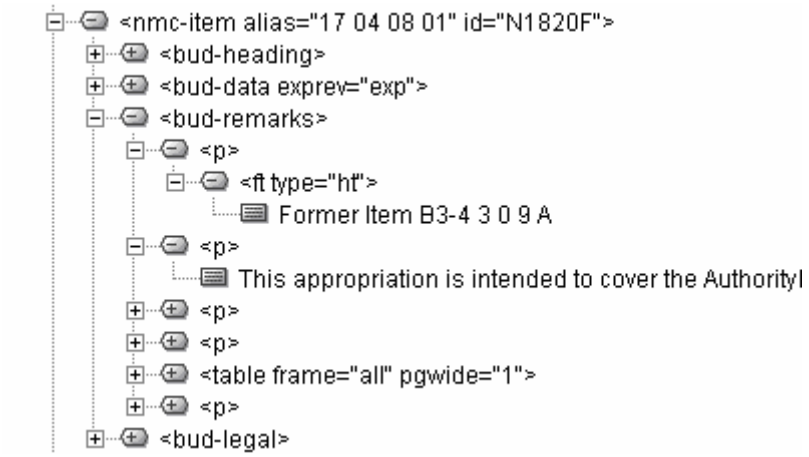
References to budget lines (aliases) in the old nomenclature;

"Internal" references, for example to Chapter XX 01.

Remarks <bud-remarks>

The element <bud-remarks> is used to contain budget remarks.

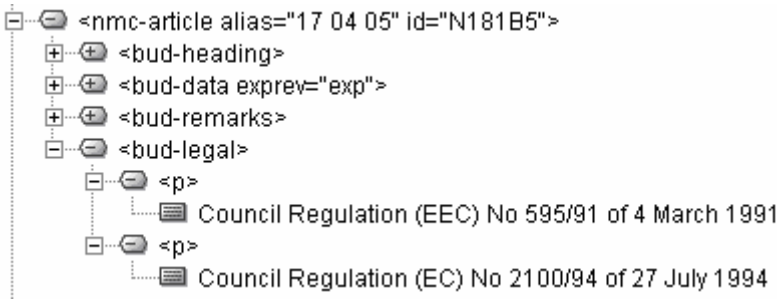
The template for this element is "basic text"; its content may therefore consist of "paragraphs", "lists" and "tables".



Legal basis <bud-legal>

The element <bud-legal> is used to contain the legal basis of a budget line.

The template for this element is "basic text"; its content may therefore consist of "paragraphs", "lists" and "tables". In practice, its content only consists of paragraphs. In certain exceptional cases, it may be that lists are used to give some structure to the text.



Reference acts <bud-reference>

The element <bud-reference> is used to contain everything that used to be in the legal basis of a budget line but which is not strictly a legal basis. This distinction between "legal basis" and "reference act" was introduced for the production of the 2004 ABB.

The template for this element is "basic text"; its content may therefore consist of "paragraphs", "lists" and "tables". In practice, its content only consists of paragraphs. In certain exceptional cases, it may be that lists are used to give some structure to the text.

5.6.2.4. "Basic text" (BTX)

General information

By "basic text" we mean everything used to structure basic text, namely:

1) Paragraphs (p)

2) Lists

3) Tables

4) The "inline" structuring of content (ft, qt, etc.)

5) REUSE

The structuring of "basic text" is relatively simple. Its purpose is more to enable sufficient structure to be given to the finished products (RTF, HTML, "paper", etc.) than to provide any semantic depth. A full description of the way basic text is structured lies outside the scope of this document.

Tables (CALS)

CALS is used for "marking" tables. The conversion filters to the finished products (RTF, HTML, etc.) do not take account of all the embellishments that CALS marking allows. For example, the system – at present – does not allow for the creation of "tables within tables" (Word 97 limitation).

REUSE

"REUSE" is a functionality that was introduced for the production of the 2004 ABB. The aim is to be able to insert in a budget document tags indicating that "at this point" there is information that may be "reconstructed" from other information present in the instance.

For example, the "summary" tables in Chapter XX 01 are really "REUSEs" that will be resolved through the use of budget figures present in the Volume 4 expenditure.

In the SEI-BUD system, "REUSEs" are "resolved" (REUSE tag replaced by the "result") on consultation in the case of "resolved" formats (for example, the Word draft).

5.6.3. "Volume 0" DTD

The VL0 DTD is a "basic text" DTD and it differs from the ABB DTD in that it does not have any particular semantic for marking pure budget data (in particular, the budget figures).

In the VL0 DTD, the concept of separation between the "Repository" DTD and the "Presentation" DTD is retained, even though these DTDs are identical here. In fact, because – for example – there is no specific semantic for marking figures, the issue of the "presentation" of this information does not arise.

5.6.3.1. General structure and organisation

The VL0 DTD is organised (structured) in the following way:

vl0.dtd	"main"
Vl0-hier.mod	Groups together the definitions relating to the general structure of a document (parts, sect1, sect2, sect3, preamble, annex, etc.).
Vl0-btx.mod	Groups together the definitions relating to the basic text, such as the definition of paragraphs, lists, etc.
Vl0-tbl.mod	Gives the definitions relating to the table model. That definition "includes" the DTD relating to the CALS table models (calstblx.dtd).
calstblx.dtd	DTD based on the CALS table model.

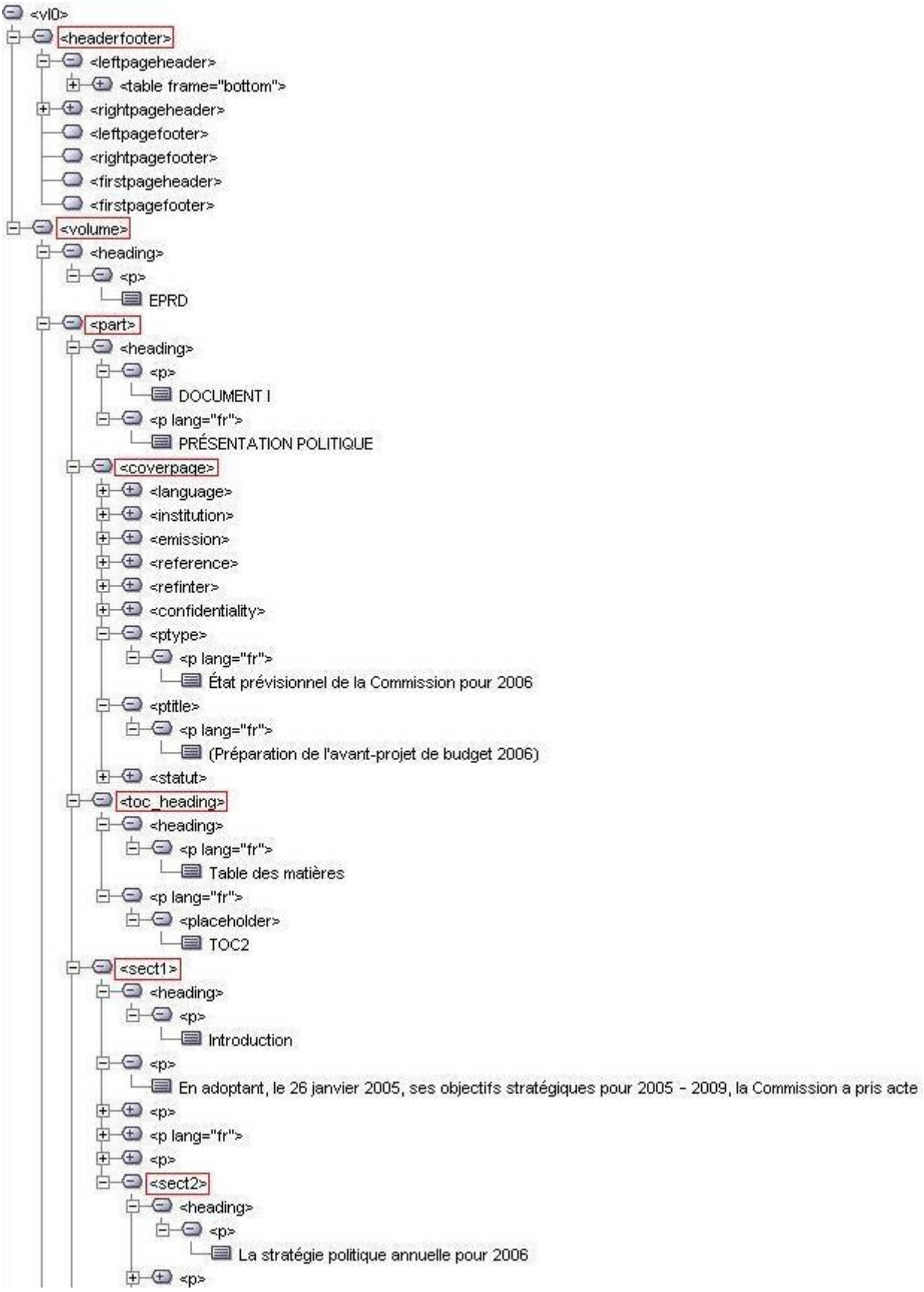
5.6.3.2. The general structure of a VL0

A vl0 document is a volume organised into "parts", consisting of "sect1", consisting in turn of "sect2", etc.

Header and footer can be defined for the whole document.

The volume and the parts can have a cover page, as well as a placeholder for the table of content. Resolution of placeholders is performed in the Word-based editing tool.

Typically, the structure (organisation) of a document of this type looks like this:


<vl0>
<headerfooter>
<leftpageheader>
<table frame="bottom">
<rightpageheader>
<leftpagefooter>
<rightpagefooter>
<firstpageheader>
<firstpagefooter>
<volume>
<heading>
<p>
EPRD
<part>
<heading>
<p>
DOCUMENT I
<p lang="fr">
PRÉSENTATION POLITIQUE
<coverpage>
<language>
<institution>
<emission>
<reference>
<refinter>
<confidentiality>
<ptype>
<p lang="fr">
État prévisionnel de la Commission pour 2006
<ptitle>
<p lang="fr">
(Préparation de l'avant-projet de budget 2006)
<statut>
<toc_heading>
<heading>
<p lang="fr">
Table des matières
<p lang="fr">
<placeholder>
TOC2
<sect1>
<heading>
<p>
Introduction
<p>
En adoptant, le 26 janvier 2005, ses objectifs stratégiques pour 2005 - 2009, la Commission a pris acte
<p>
<p lang="fr">
<p>
<sect2>
<heading>
<p>
La stratégie politique annuelle pour 2006
<p>

5.6.3.3. "Basic text" (BTX)

General information

By "basic text" we mean everything used to structure basic text, namely:

1) Paragraphs (p)

2) Lists

3) Tables

4) Pictures

5) The "inline" structuring of content (ft, qt, placeholder, etc.)

6) REUSE

The structuring of "basic text" is relatively simple. Its purpose is more to enable sufficient structure to be given to the finished products (RTF, HTML, "paper", etc.) than to provide any semantic depth. A full description of the way basic text is structured lies outside the scope of this document.

Tables (CALS)

CALS is used for "marking" tables. The conversion filters to the finished products (RTF, HTML, etc.) do not take account of all the embellishments that CALS marking allows. For example, the system – at present – does not allow for the creation of "tables within tables" (Word 97 limitation).

REUSE

"REUSE" is a functionality that was introduced for the production of the 2004 ABB. The aim is to be able to insert in a document tags indicating that "at this point" there is information that may be "reconstructed" from other information present in this instance or in another instance (including an ABB instance).

For example, the "summary" tables for policies in Part II of Volume 0 are really "REUSEs" that are resolved through the use of budget figures present in the Volume 4 expenditure.

In the SEI-BUD system, "REUSEs" are "resolved" (REUSE tag replaced by the "result") on consultation (of certain formats such as the Word draft).

5.6.4. DTD's used for amendments

The system uses several DTD's for amendments; one storage DTD and several "production" DTD's; the "production" DTD's are used to model temporary intermediate XML documents before their actual conversion to a final output format:

1) the amendment document DTD for storage; this DTD is used for textual parts of the amendment;

2) the Full Text ("FT") DTD;

3) the Reduced Text ("RT") DTD;

4) the Translation Text (TT") DTD;

5) DTD's used for various reports.

This section only describes the storage DTD; the reader is referred to ad-hoc sections for the description of "production" DTD's.

The AMD storage DTD is used for textual content: it does not include all metadata, nor the details of amendment transactions.

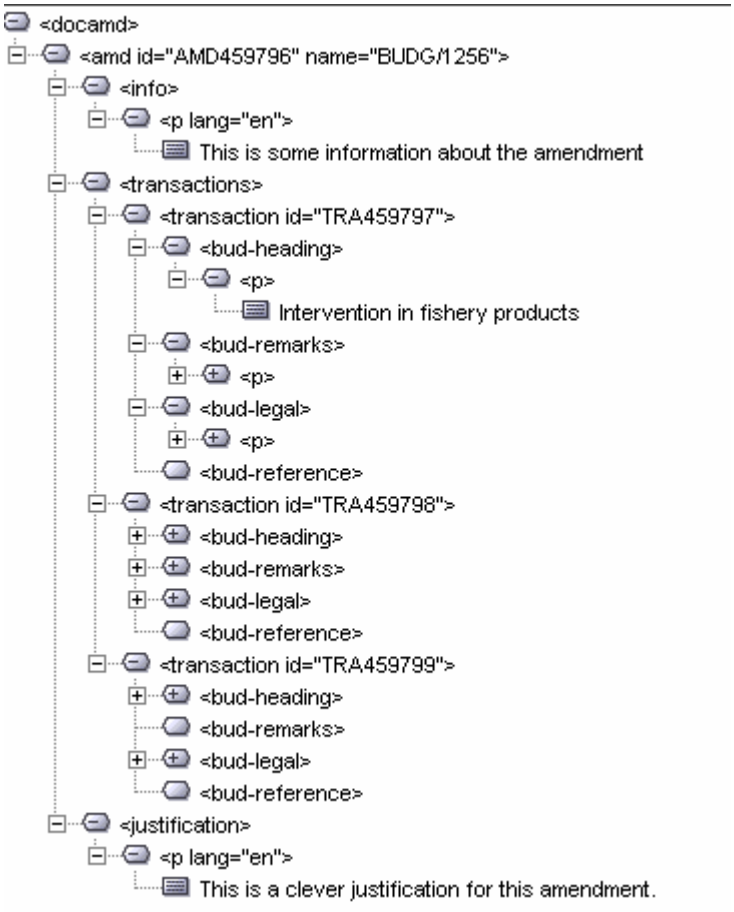
5.6.4.1. General structure and organisation

The AMD DTD is organised (structured) in the following way:

amd.dtd	"main"
amd-hier.mod	Groups together the definitions relating to the general structure of an amendment document.
abb-btx.mod	Groups together the definitions relating to the basic text, such as the definition of paragraphs, lists, etc. This part is shared with the ABB DTD as its name indicates.
abb-tbl.mod	Gives the definitions relating to the table model. That definition "includes" the DTD relating to the CALS table models (calstblx.dtd). This part is also shared with the ABB DTD.
calstblx.dtd	DTD based on the CALS table model. This part is also shared with the ABB DTD

5.6.4.2. The general structure of amendment document

Basically, the structure of a linguistic version of a stored amendment document looks like this:



The document has two attributes: the amendment unique identifier (`id`attribute) and the amendment label (`name`attribute). It is divided into three main parts: the information (`info`element), the transactions (`transactions`element) and the justification (`justification`element).

Each transaction has an identifier (`id`attribute) and contains the four main textual parts of a budget item: the heading (`bud-heading`element), the remarks (`bud-remarks`element), legal bases (`bud-legal`element) and reference acts (`bud-reference`element). These textual parts together with other elements such as`bud-intro`and`bud-text`have the same model as described for the ABB DTD. The reader is referred to the annexes for the complete source of this DTD.

5.6.4.3. "Basic text" (BTX)

The reader is referred to the ABB DTD for a description of the basic text components.

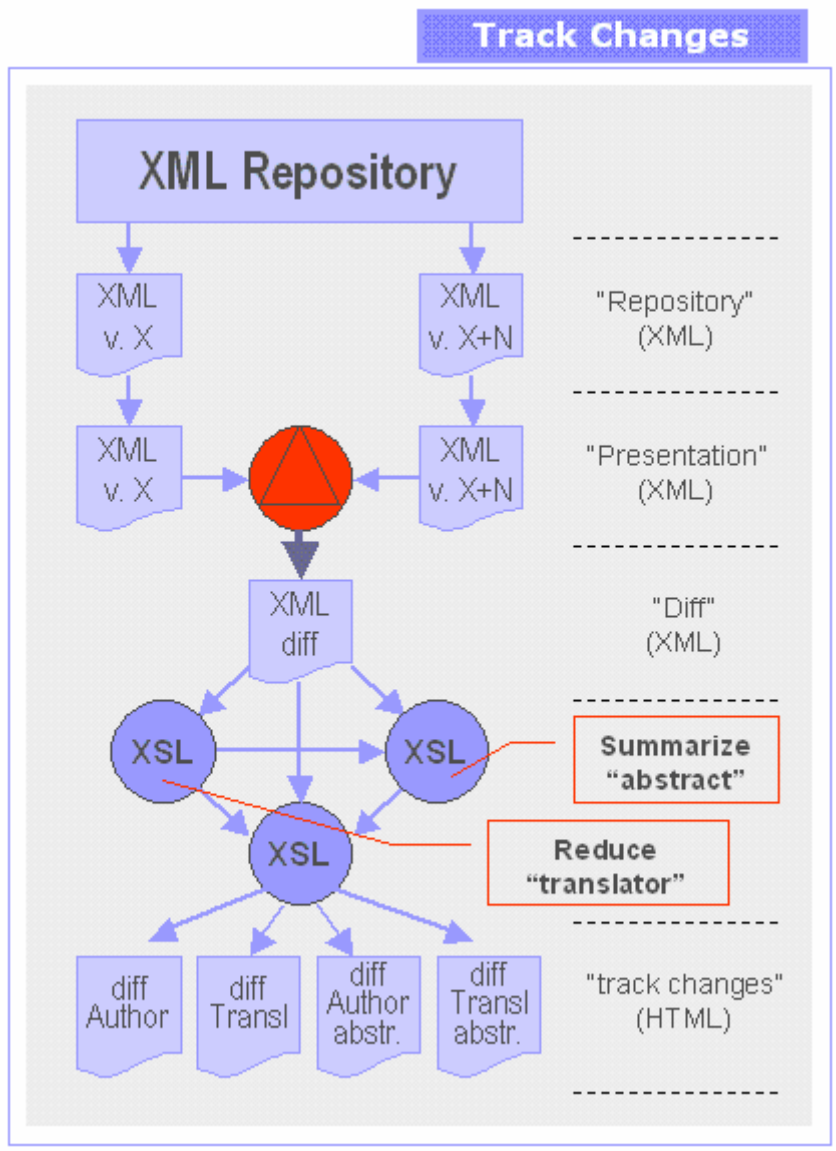
5.7. Conversion filters and tools

5.7.1. Producing the "differential" (change-marked text)

This section describes the tools implemented for the production of the different types of differential (one type of differential for each type of person involved in production).

5.7.1.1. The different types of differential

The production of the differential (printout of differences) is based on the difference calculation tool (xdiff) as described in the previous section. The different types of difference printouts (author, corrector, translator) are obtained in a similar way.



Background:

1) The "Author" differential shows all modifications (to nomenclature, figures, headings and remarks), irrespective of the editing tool used. Modifications appear there in the form of insertions and removals of words.

2) The "Translator" differential shows the modified text that is to be translated. It does not report paragraphs that have been removed completely or modifications made to figures. However, it does report modifications made to schedule notes. Modifications appear in the form of insertions of paragraphs (any paragraph modified by an author automatically replaces the equivalent paragraph in all language versions).

3) The "Corrector" differential shows the modified text that is to be corrected. Like the "Translator" differential, it does not report paragraphs that have been removed completely or modifications made to figures. However, it does report modifications made to text, including schedule notes. Modifications appear in the form of insertions and removals of words.

There is an additional option that enables all data in the unmodified budget lines to be included; by default, budget lines that have not been modified are indicated in the following way:

Article 19 02 01 — European Union Analysis and Evaluation Centre/Conflict Prevention Network

+ + + no diffs + + +

Article 19 02 02 — Institutes specialising in relations between the European Union and non-EU countries

+ + + no diffs + + +

In order to give a more concrete illustration of the different types of differential, some examples are given below.

Example of an author differential

Article 19 02 03 — Information programmes for non-member countries

Appropriations 2005		Appropriations 2004		Outturn 2003	
Commitments	Payments	Commitments	Payments	Commitments	Payments
7 ~~6~~ 450 000	6 ~~5~~ 717 000	5 455 000	5 455 000	4 914 331,70	4 943 960,91

The likely schedule of payments vis-à-vis commitments is as follows :

Commitments		Payments				
		2004	2005	2006	2007	Subsequent years
Pre-2004 commitments still outstanding						
Commitment appropriations made available again and/or carried over from 2003	250 000	250 000				
Appropriations 2004	250 000		250 000			
Appropriations 2005						
Total	500 000	250 000(1)	250 000(2)			

(1) Un crédit de 50 000 euros est inscrit au chapitre 31 02 41.

(2) Un crédit de 50 000 euros est inscrit au chapitre 31 02 41.

Commentaires

This is a new paragraph.

In its communication of 11 April 2003~~February 2000~~ to the European Parliament, the Council, the Court of Auditors, the Economic and Social Committee and the Committee of the Regions on strategic objectives 2000 to 2005 Shaping the new Europe (OJ C 81, 21.3.2000, p. 1), the Commission laid down its strategic priorities for the period 2000 to 2005, which include efforts to give Europe 'a stronger voice in the world of tomorrow~~a stronger voice in the world~~.

The main guidelines for 2004~~2003~~ are:

- ~~to develop a strategic approach by refocusing information measures and matching them to the priorities of the European Union's external policy in order to set out and promote a consistent and dynamic image of that policy;~~
- to develop regional coordination of the information programmes of Commission delegations,
- ~~to collaborate with the Member States in organising joint activities in non-member countries;~~
- to make greater use of new technologies in order to spread information quickly and in targeted fashion (Internet, electronic mail),

Example of a translator differential

Article 19 02 03 — Information programmes for non-member countries

Appropriations 2005		Appropriations 2004		Outturn 2003	
Commitments	Payments	Commitments	Payments	Commitments	Payments
7 450 000	6 717 000	5 455 000	5 455 000	4 914 331,70	4 943 960,91

The likely schedule of payments vis-à-vis commitments is as follows :

Commitments		Payments				
		2004	2005	2006	2007	Subsequent years
Pre-2004 commitments still outstanding						
Commitment appropriations made available again and/or carried over from 2003	250 000	250 000				
Appropriations 2004	250 000		250 000			
Appropriations 2005						
Total	500 000	250 000(1)	250 000(2)			

(1) Un crédit de 50 000 euros est inscrit au chapitre 31 02 41.

(2) Un crédit de 50 000 euros est inscrit au chapitre 31 02 41.

Commentaires

This is a new paragraph.

In its communication of 11 April 2003 to the European Parliament, the Council, the Court of Auditors, the Economic and Social Committee and the Committee of the Regions on strategic objectives 2000 to 2005 Shaping the new Europe (OJ C 81, 21.3.2000, p. 1), the Commission laid down its strategic priorities for the perioc 2000 to 2005, which include efforts to give Europe 'a stronger voice in the world of tomorrow.

The main guidelines for 2004 are:

- to develop regional coordination of the information programmes of Commission delegations,
- to make greater use of new technologies in order to spread information quickly and in targeted fashion (Internet, electronic mail),

Example of a corrector differential

Article 19 02 03 — Information programmes for non-member countries

Appropriations 2005		Appropriations 2004		Outturn 2003	
Commitments	Payments	Commitments	Payments	Commitments	Payments
7 450 000	6 717 000	5 455 000	5 455 000	4 914 331,70	4 943 960,9

The likely schedule of payments vis-à-vis commitments is as follows :

Commitments		Payments				
		2004	2005	2006	2007	Subsequent years
Pre-2004 commitments still outstanding						
Commitment appropriations made available again and/or carried over from 2003	250 000	250 000				
Appropriations 2004	250 000		250 000			
Appropriations 2005						
Total	500 000	250 000(1)	250 000(2)			

(1) Un crédit de 50 000 euros est inscrit au chapitre 31 02 41.

(2) Un crédit de 50 000 euros est inscrit au chapitre 31 02 41.

Commentaires

This is a new paragraph.

In its communication of 11 April 2003 to the European Parliament, the Council, the Court of Auditors, the Economic and Social Committee and the Committee of the Regions on strategic objectives 2000 to 2005 Shaping the new Europe (OJ C 81, 21.3.2000, p. 1), the Commission laid down its strategic priorities for the perio 2000 to 2005, which include efforts to give Europe 'a stronger voice in the world of tomorrow.

The main guidelines for 2004 are:

- to develop regional coordination of the information programmes of Commission delegations,
- to make greater use of new technologies in order to spread information quickly and in targeted fashion (Internet, electronic mail),

5.7.1.2. The differential production process

The different stages in the production of a differential are as follows:

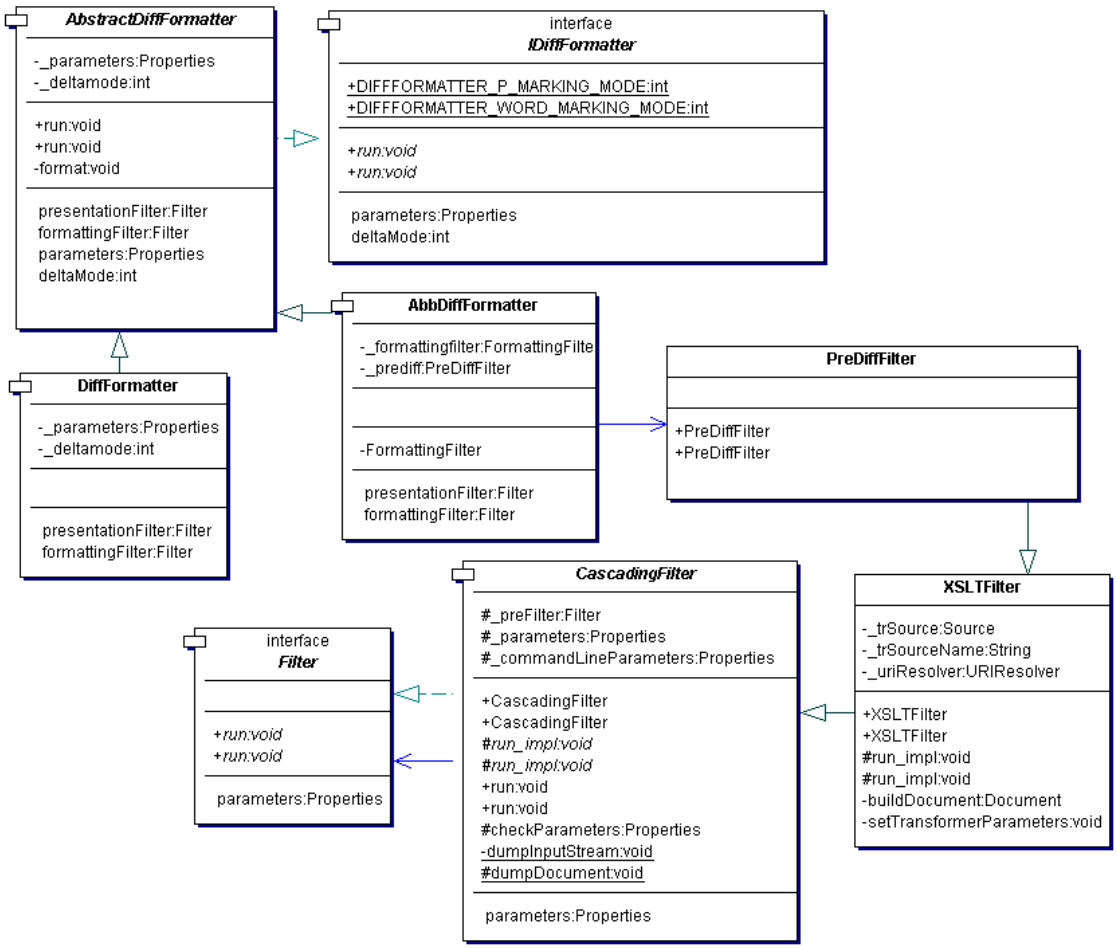
1) Determination of the version numbers to be compared, according to the type of differential mentioned in the request ("initial" version or "current" version).

2) Extraction in consultation mode of the two document fragments to be compared.

3) Formatting of the two documents in line with a "differential formatter" as specified in the request and the characteristics of which are detailed in the configuration file "document-repository-config.xml" (see for example the parameter "diff-formatter" of the format "htm-diff-text-abb").

4) The result of the differential is packaged and may also be compressed before being returned to the client.

The differential formatter is used in the filtering framework that is part of Xef. A typical filter formatter is that used for the HTML output of the author differential of the ABB publication. A simplified class diagram of this formatter is given in the figure below:



The class "`AbbDiffFormatter`" is an extension of the abstract class "`AbstractDiffFormatter`", which is part of Xef; this abstract class implements the interface "`IDiffFormatter`". This interface has four methods: one method to determine the differential mode, one method to set the parameters and two methods to execute the formatting (method with stream or with DOM document).

The class "`AbbDiffFormatter`" uses the method "run", available by default in "AbstractDiffFormatter". This method carries out the following steps:

1) determination of a presentation filter ("`PreDiffFilter`"); this filter is obtained by interrogation of the class "`AbbDiffFormatter`";

2) execution of the presentation filter on both of the document fragments for comparison; because the results are DOM documents, they are serialised in a stream;

3) execution of the method "`format`", which instantiates a "Delta" class responsible for producing a comparison using the tool "xdiff" and executes its "run" method with the two previous streams as input parameters; this method produces a result in the form of an input stream;

4) determination of a result formatting filter ("`FormattingFilter`") and execution of this filter on the previous result;

In our case, the presentation filter ("`PreDiffFilter`" class) inherits from an "`XSLTFilter`" class that is part of the Xef framework. It consists of a single XSLT filter ("pre_diff.xslt").

The result formatting filter ("`FormattingFilter`") is a sub-class of the class "AbbDiffFormatter" and inherits from the class "`CascadingFilter`", which is part of the Xef framework. This filter can show or hide the budget figures ("`hide-datadiff.xslt`"); it can also show or hide the items that have not changed ("`hide-identical.xslt`"). Depending on the options, zero or two XSLT filters are cascaded before the presentation filter "`prs2htm.xslt`" (described in another section).

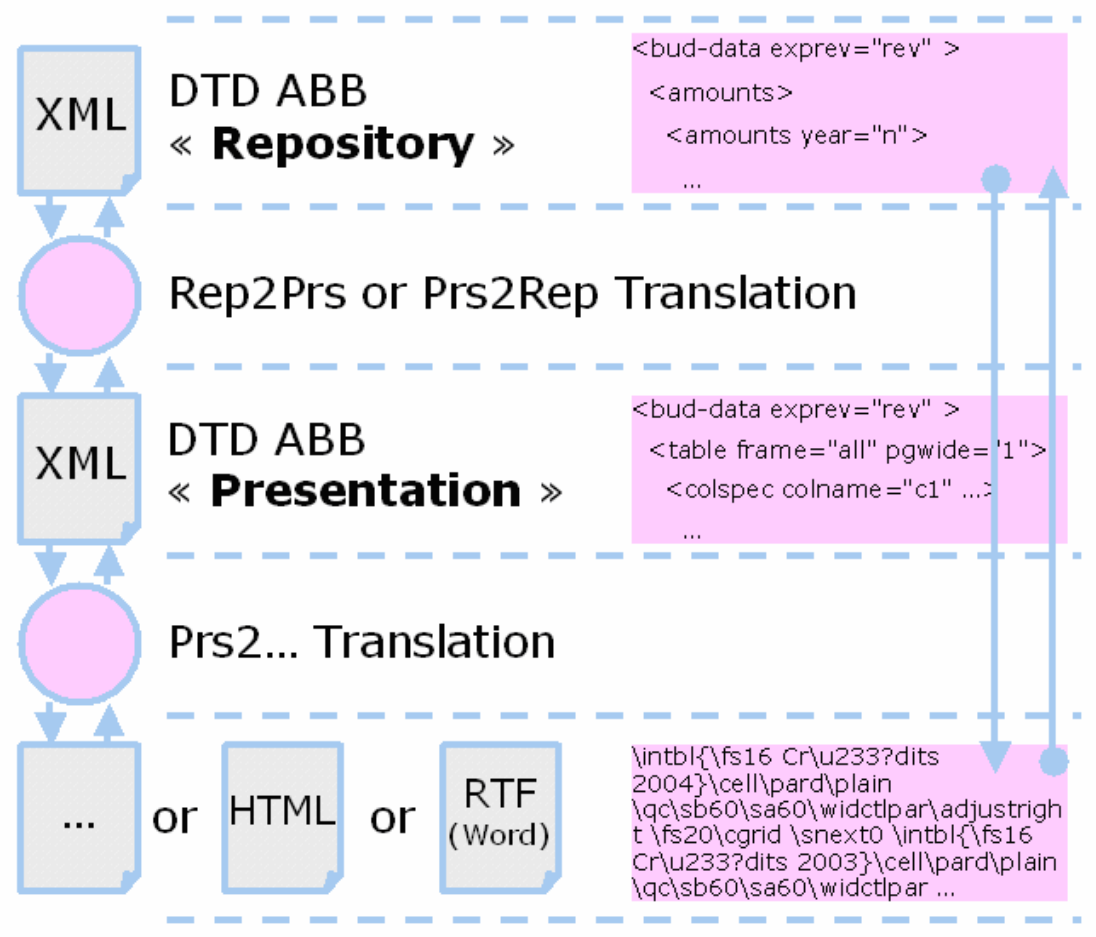
The filter "`pre_diff.xslt`" is relatively trivial: it is specific to the ABB publication and adds anchor points for certain elements (bud-remarks, bud-leagl, etc.) in order to force synchronisation by the difference calculation programme ("xdiff"). These anchor points are subsequently removed by the presentation filter "`prs2htm.xslt`".

The filter "`hide-datadiff.xslt`" is used in the case of a translator differential to remove the differences located within the budget figures; it also removes the removal differences.

The filter "`hide-identical.xslt`" examines each of the nomenclature elements to determine whether or not they contain differences. If not, their content is replaced by a standard text explaining that they do not contain any differences.

5.7.2. "Repo" to "presentation" conversion

The aim of the intermediate presentation format is principally to reduce the complexity of conversions (through the needless repetition of conversion rules) by passing through a sort of pivot format representing the constructions associated with presentation.



This conversion could lose some of the semantics of the "repo" document. Thus the figures for a budget item could be structured in an anonymous table. If this table is taken out of context, it will no longer be possible to determine whether it contains budget figures or is simply an anonymous table.

Going through a pivot representation means that conversions to the various output formats (RTF, HTML, etc.) can be simplified. To take the same example, the translation of a table to RTF could be reused both for an "anonymous" table and a table coming from the budget figures. In addition, as this pivot representation is close to a "physical presentation" format, it will be easier to make the transformation.

The principles applied are therefore modularity, reuse and factorisation.

It should be remembered that this conversion is not two-way: some information (such as budgetary amounts) is impoverished and cannot then be converted by a reverse filter.

This conversion is implemented by the class "`Abb2prs`". This class is derived from the class "`XSLTFilter`", which is part of the Xef framework.

The filter requires two mandatory parameters: the document language and the budget year.

5.7.2.1. The style sheet "abb2prs.xslt"

By default, this XSLT style sheet copies in its output everything it receives as input. Example of use:

```
java com.icl.saxon.StyleSheet -o {xml output file} {xml input file} abb2prs.xslt year={yyyy}
lang={lg}
```

This style sheet also includes several other style sheets:

1) "abb2prs-fig.xslt" converts budgetary amounts into CALS tables, including in them the standard headers for the language of the document being processed; different table structures are produced according to whether or not the amounts are expenditures, differentiated, etc.

2) "abb2prs-fig-extra.xslt" processes budgetary amounts broken down for the countries before enlargement into CALS tables; in practice, if breakdowns are available, one table is created for the 15 countries before enlargement and another table for the 10 additional countries; this style sheet performs arithmetical operations as well as assembling the amounts into tables.

3) "abb2prs-sch.xslt" converts the schedules into CALS tables, including in them the standard headers for the language of the document being processed;

4) "abb2prs-hrs.xslt" converts figures relating to human resources into CALS tables;

5) "abb2prs-uti.xslt" groups together several rules common to conversions into CALS tables, in particular the output of the various attributes relating to a table cell, and the output of column specifications;

6) "stdtxt-templates.xslt" groups together various character strings added during presentation (table headers, etc.), generally depending on the document language; this style sheet is generated automatically by the module "stdtxt".

5.7.3. "Presentation" to "repo" conversion

This conversion is used during the process of updating certain parts of the publication, such as the remarks. The full data cannot be converted to "repo" format, so the budgetary amounts are not retrieved by means of this conversion.

This conversion is implemented by the class "Prs2abb". This class is derived from the class "XSLTFilter", which is part of the Xef framework.

The filter does not require any parameters and uses the style sheet "prs2abb.xslt".

This style sheet includes the style sheet "prs2abb-sch.xslt", which is used to convert a schedule in the form of a CALS table into "repo" format.

All CALS tables used for budgetary amounts and human resources are ignored.

5.7.4. "Repo" to RTF conversion

This conversion is implemented by the class "Abb2rtf", which derives from the class "XSLTFilter", which is part of the Xef framework.

This filter uses the conversion from "repo" to presentation; since this causes impoverishment, as mentioned in the description of the filter "abb2prs", not all the data (in particular the budgetary amounts) can undergo the reverse conversion.

This filter requires a mandatory parameter: the document language.

This conversion filter cascades three filters:

1) "abb2wrx" converts "repo" format into the intermediate presentation format "wrx" in preparation for a further conversion to RTF;

2) "utf82rtf" performs a code conversion from UTF8 to a representation specific to RTF format (using the construction "?);

3) the style sheet "xml2rtf" is used to convert the presentation format "wrx" into RTF.

5.7.4.1. The filter "abb2wrx"

This filter cascades two filters, one of which is "abb2prs" (described above), which converts "repo" format to "presentation" format. The second filter is the style sheet "prs2wrx".

The format "wrx" (for**W**o**R**d**X**ml) is formally described by the DTD "abb": it is a configuration of this DTD by marked sections.

This style sheet uses a single parameter, which is the language of the document to be processed. By default, all XML constructions are copied to the output, except the attributes for which an empty value is specified. The XML envelopes of the budgetary amounts ("bud-data", "bud-data-extra") are copied with neither their content nor their attributes.

The style sheet includes the other style sheets listed below:

1) "prs2wrx-quote.xslt" contains the definitions for quote marks in the different languages supported;

2) "prs2wrx-btx.xslt" carries our several conversions:

3) conversion of "qt" elements into characters; this conversion depends on the document language;

4) conversion of "ft" elements into "emphasis", "glossterm", "subscript", "superscript" and "uppercase" elements

5) lists are also converted (in particular by removing the introductions "int.li" and closures "close.li");

6) "prs2wrx-table.xslt" converts tables into a form that is easier to translate into RTF; among other things, this filter translates "colspec" elements by carrying over the cell width information to the cells themselves (attribute "hpos"); it also converts the way cells merged vertically are represented;

7) "prs2wrx-grseq.xslt" converts "grseq" elements into "sect1", "sect2", etc. elements depending on the level at which they are nested; for these elements, the attributes "id" and "alias" are removed;

8) "prs2wrx-rmatt.xslt" removes the attributes "id" and "alias" from elements other than "grseq".

5.7.4.2. The filter "utf82rtf"

This filter extends the class "CommandLineFilter" of the Xef framework. It controls a small programme written in "C". This programme receives two parameters: an input filename encoded in UTF8 and an output filename.

This programme converts any UTF8 characters that are not purely ISO 646, using the RTF escape sequence "? followed by the decimal integer code representing the corresponding code in UCS-2. Thus the character Ā will be converted to "Ā". The decimal integer can in fact be followed by a "fallback" or substitute character if the character referred to cannot be displayed; in this case, a question mark is used.

Some characters ("meta-characters" in RTF syntax) are translated another way: these are the characters "{", "}", "" and the hyphen and non-breaking-hyphen.

5.7.4.3. The style sheet "xml2rtf"

The templates in this style sheet are grouped into two parts: on the one hand, templates giving RTF code definitions, and on the other, templates that are attached declaratively to the XML elements of the presentation and that use the initial RTF code definitions.

This style sheet is fairly complex and controls several other style sheets. The following style sheets are generated automatically using a SAG tool that enables the extraction of certain information from an RTF document template (it is also possible to produce these style sheets manually using copy and paste):

1) "`doc_head0.xsl`" contains the definition of the first part of the RTF header, located before the default language specification of the document;

2) "`doc_head1.xsl`" contains the part of the RTF header that lies between the default language definition and the definition of the styles;

3) "`doc_sty.xsl`" contains the part of the RTF header relating to the definition of the styles

4) "`doc_lst.xsl`" contains the part of the RTF header relating to the definition of the lists

5) "`doc_lstov.xsl`" is the same, but for the "listoverridetable", i.e. the table of list definitions obtained by modifying a basic list;

6) "`doc_head2.xsl`" contains the part of the RTF header that includes the information that appears in the document properties;

7) "`doc_tmpl.xsl`" contains the part of the header containing a reference to the document template ("template");

8) "`doc_head3.xsl`" contains the last part of the RTF header;

9) "`doc_styles.xsl`" contains a series of XSLT templates including the RTF information to be generated when a style is used; every style, paragraph style or character string style has its own template.

The style sheet "`xml2rtf`" first issues the RTF header, then applies the templates to the document content, and then issues the RTF end-of-document code. It also contains the templates used to generate the document properties, and several templates called up before and after each paragraph is output.

The style sheet "`xml2rtf`" also includes the following style sheets:

1) "app.xsl" primarily includes the conversion code for the document type "abb": this code is used to convert the nomenclature elements, floating elements within paragraphs (bold, italic, etc.) and table notes and titles;

2) "xml2rtf_list.xsl" groups together the generic processing of lists (numerical and non-numerical lists).This style sheet has two parts. The first is activated when the RTF header is produced: it has to produce a list definition for all the lists used ("listtable" and "listoverridetable") – to do this, it must look through the whole document. Spaces for different identifiers are used for numbered and non-numbered lists;The second part contains templates activated on the list elements found in the document itself. The processing of list elements obviously depends on the level at which they are nested.

3) "xml2rtf_tables.xsl" covers the processing of the tables

4) "xml2rtf_reuse.xsl" processes the reuse tags ("reuse-link") by copying the tagging itself into a private hidden field and by displaying this field in grey with a note to show that it is reuse information.

5) "xml2rtf_lang.xsl" contains the language-dependent templates (document language code, character strings to be used to separate different parts of the document).

5.7.5. RTF to "repo" conversion

This conversion is implemented by the class "Rtf2abb", which derives from the class "XSLTFilter", which is part of the Xef framework.

The filter uses the "presentation" to "repo" conversion described earlier.

This conversion filter cascades the following filters:

1) "Rtf2piv" extracts certain information from an RTF document and produces an XML (pivot) file;

2) "piv2tok.xsl" is a style sheet used to convert the XML pivot into an intermediate conversion file;

3) "Tok2wrx" is a parser that processes the intermediate file just produced and creates the presentation format "wrx"

4) "wrx2prs.xslt" is a style sheet that makes it possible to go from "wrx" format (for WoRd Xml) to the presentation format

5) "prs2abb.xslt" is the style sheet for converting presentation format to "repo" format; it is described in an earlier section.

5.7.5.1. The filter "Rtf2piv"

This filter extends the class "CommandLineFilter" of the Xef framework. It controls an executable programme that is a SAG tool used in many projects, developed by SAG; this programme is available both in Windows and in Solaris.

The activation syntax of this programme is as follows:

```
rtf2piv rtf_file pivot_file–cfg=param_file
```

where:

1) `rtf_file`is the RTF source filename

2) `pivot_file`is the name of the XML document to be produced

3) `cfg_file`is the name of a parameter file (".ini" format)

The following is an example of an activation command:

```
rtf2piv doc1.rtf doc1.piv–cfg=rtf2piv.ini
```

A second activation syntax is used to produce a parameter file that is auto-documented by default:

```
rtf2piv –cfgtemplate=rtf2piv.ini
```

5.7.5.1.1. Configuration of the RTF to pivot conversion

N.B.: most of the parameters take the Boolean value "0" to suppress the facility or "1" to activate it. Some parameters depend on other parameters being activated (e.g. if you do not want to keep the images, all the other parameters relating to image extraction irrelevant). The default values are given for information.

Parameters relating to images

1) **LeaveTempfile=0**the value "1" can be used to delete temporary image files (such as ".bmp" or ".wmf")

2) **ConvertPictToMono=1**enables colour images to be converted to black and white images in TIFF format (this requires a library that is not normally supplied with the executable rtf2piv).

3) **KeepPictures=1**the value "0" is used to not extract any images

4) **ConvertToPs=1**to convert images into PostScript (available only in Windows); this option also requires the installation of a PostScript printer (named RTF2PIV) configured to print to file.

5) **ConvertToTiff=1**is used to convert bitmap images to TIFF format.

6) **ConvertMetafileToBmp=0**enables metafiles to be converted to Windows bitmaps (".bmp" format)

7) **ConvertMetafileToJPG=0**enables metafiles to be converted to JPG

8) **ConvertMetafileToPNG=0**converts metafiles to PNG

9) **Pict_Format=1**adds the attribute "FORMAT", which specifies the image format

10) **KeepObjects=0**enables objects included in the document to be extracted to separate files

11) **Parameters relating to the formatting of character strings**

12) **CS_Bold=1**extracts information in bold

13) **CS_Italic=1**extracts information in italics

14) **CS_Script=1**extracts "superscript" and "subscript" information

15) **CS_Font=1**extracts font names

16) **CS_Size=1**extracts font sizes

17) **CS_Style=1**extracts character styles

18) **CS_Caps=0**extracts capitalisation information

19) **CS_Hidden=0**extracts hidden information

20) **CS_More=0**extracts certain additional information such as the different underlining combinations.

Parameters relating to paragraph formatting

1) **P_Li=1**extracts left indentation (can be a negative integer)

2) **P_Ri=1**extracts right indentation

3) **P_Align=1**extracts alignment information

4) **P_Style=1**extracts paragraph styles

5) **Parameters relating to tables**

6) **Cell_Border=0**extracts cell border information

7) **Cell_VMerge=0**extracts vertical merge information

8) **Cell_VAlign=0**extracts cell alignment information

9) **Row_Header=0**extracts row header information

10) **Row_Align=0**extracts row alignment information

Parameters relating to lists

It should be noted that Word lists are represented as strings of paragraphs. Lists can even be orthogonal. Each list has a unique identifier. Section titles are also represented as lists.

The output of the programme "rtf2piv" does not support interlaced lists – lists must have a tree structure. The programme "rtf2piv" uses left indentation to determine both the level of nesting of list items and whether a paragraph is part of the current list item or of a list item at a lower level of nesting.

Additionally, the programme closes all lists normally when a paragraph with an "outline level" appears (i.e. a paragraph likely to appear in the table of contents).

1) **UseLists=0**is used to extract list tagging

2) **P_ListInfo=0**extracts additional information by taking the following attributes from the "LIST" item:

    a) LISTID is the unique numerical identifier of the list

b) LISTTYPE is a numerical code representing the type of list (see RTF documentation)

c) LISTNUMB is used to distinguish between numbered and non-numbered lists

d) STARTAT gives the starting value for a numbered list

3) **UseListLevels=STYLE**makes it possible to specify a paragraph style for which a list level will be used in order to determine the level of nesting; "STYLE" is a style name to be specified.

4) **NumCloseListStyles=2**makes it possible to specify a consecutive number of paragraph styles that will close all the open lists.

5) **CloseListStyle0=STYLE1**specifies a paragraph style name

6) **CloseListStyle1=STYLE2**specifies a paragraph style name

7) **NumNoCloseListStyles=2**is used to specify a consecutive number of paragraph styles that will not close open lists (the paragraphs in question with therefore be included in the current list item)

8) **NoCloseListStyle0=STYLE1**specifies a paragraph style name

9) **NoCloseListStyle1=STYLE2**specifies a paragraph style name

Miscellaneous parameters

1) **UseFontNum=0**is used to specify a font reference that will point to a font table rather than the font name

2) **UseStyleNum=0**is used to specify a style reference that will point to a style table rather than the style name. The reference will point to the paragraph styles table or the character styles table, as appropriate.

3) **UseBookmarks=0**is used to extract information about bookmarks

4) **KeepFldinst=0**is used to extract information about fields (FLD1)

5) **KeepFldrslt=0**is used to extract information about result fields (FLD2)

6) **UsePrivateFields=0**is used to extract the content of private fields

7) **UseUlink=1**is used to extract links

8) **UseSectionBreaks=0**is used to extract section break information

9) **UTF8OutputCharset=1**specifies UTF8 output encoding

10) **XMLOutput=1**forces XML output instead of SGML output (differences in terms of empty elements and character escape)

11) **SurroundData=0**is used to enclose character strings in a tag (DATA) giving information about the location of the string in the RTF document. It should be noted that the paragraph number given presupposes that all constructions are visible; in fact, the use of this information to position MS Word is dependent upon the display options.

12) **StartEndData=0**extracts character positions

13) **GenText=1**extracts text generated automatically by MS Word, such as section numbering

14) **UseMetadata=0**extracts the document properties

15) **WarnInvalidCharacters=0**is used to generate a warning message in the output document if a character cannot be converted using the output encoding (generally if a font has a code that is not supported by the programme).

16) **ExtractPnText=0**extracts the "pntext" information, i.e. list numbering or bullets

17) **IncludeEmptyParagraphs=0**extracts the empty paragraphs

18) **TablesWithinLists=0**allows tables to be inserted into list items (by default, all open lists are closed when a table appears).

19) **SectionBreaksCloseLists=0**indicates that section breaks close all open lists

20) **ParsWithCharStyles=0**specifies that character formatting modification information must not be output unless it is different from the information specified in the paragraph style.

21) **AddEncodingDeclaration=0**specifies that the output must include a statement of the encoding used at the start of the document.

5.7.5.1.2. DTD for the output of the RTF to pivot conversion

```
<!ELEMENT WP-DOC (FONTTABLE?, PSTYLES?, CSTYLES?, (%TEXT-ITEM;)*)
+(SECTION|METADATA|PAGEBREAK|BM-START|BM-END|LINE)>
```

The root element is the element "WP-DOC". It may contain a font table, a paragraph style table and a character style table, followed by a repetition of elements of content represented by the parameter entity "TEXT-ITEM". It should be noted that a number of exceptions from inclusion are also allowed: these are, for the most part, empty elements representing events that are orthogonal to the main structure.

The various parameter entities used as a shortened notation are as follows:

```
<!ENTITY % FLDSTUFF "(FLD1|FLD2)">
<!ENTITY % TEXT-ITEM "PIC|OBJECT|TABLE|P|%FLDSTUFF;|FOOTNOTE|LIST">
<!ENTITY % PARA-ITEM "CS|FOOTNOTE|%FLDSTUFF|PIC|OBJECT|LIST|ULINK">
<!ENTITY % FLD-CONTENT "DATA|PIC|OBJECT|TABLE|P|FOOTNOTE|LIST|CS|FLD1|FLD2">
```

Optional tables are defined as follows:

```
<!ELEMENT FONTTABLE (FONT*)>
<!ELEMENT FONT EMPTY>
<!ATTLIST FONT NAME CDATA #REQUIRED ID NUMBER #REQUIRED>
```

The font table is a dictionary giving a font name and numerical identifier.

```
<!ELEMENT PSTYLES (STY*)>
<!ELEMENT CSTYLES (STY*)>
<!ELEMENT STY EMPTY>
<!ATTLIST STY NAME CDATA #REQUIRED ID NUMBER #REQUIRED>
```

The same is true of the two style tables.

```
<!ELEMENT PAGEBREAK EMPTY>
<!ELEMENT SECTION EMPTY>
<!ELEMENT LINE EMPTY>
```

The element "PAGEBREAK" specifies a page break, the element "SECTION" indicates a section break, and the element "LINE" replaces a vertical tab.

```
<!ELEMENT PIC EMPTY >
<!ATTLIST PIC
FILE1 CDATA #IMPLIED FORMAT1 CDATA #IMPLIED
FILE2 CDATA #IMPLIED FORMAT2 CDATA #IMPLIED
XEXT CDATA #IMPLIED YEXT CDATA #IMPLIED
WIDTH CDATA #IMPLIED HEIGHT CDATA #IMPLIED
SCALEX CDATA #IMPLIED SCALEY CDATA #IMPLIED>
```

The element "PIC" refers to one or more files containing the image definition. This element has several attributes:

1) "FILE1" specifies a filename containing the image definition and "FORMAT1" the image format ("PNG", "JPG", "WMF", "BMP", etc.)

2) FILE2 specifies a filename with an alternative format, if necessary, and FORMAT2 gives its format

The other attributes give the nominal image dimensions and the enlargement factor.

```
<!ELEMENT OBJECT EMPTY >
<!ATTLIST OBJECT
FILE CDATA #IMPLIED CLASS CDATA #IMPLIED
FILE2 CDATA #IMPLIED FORMAT2 CDATA #IMPLIED
METAFILE CDATA #IMPLIED
XEXT CDATA #IMPLIED YEXT CDATA #IMPLIED
WIDTH CDATA #IMPLIED HEIGHT CDATA #IMPLIED
SCALEX CDATA #IMPLIED SCALEY CDATA #IMPLIED>
```

The element "OBJECT" gives the information for an object included in the document (Excel spreadsheet, Visio diagram, etc.). It has the following attributes:

The attribute "FILE" refers to the filename containing the serialised object;

The attribute "CLASS" specifies the object class (for example "MSGraph.Chart.8");

The attributes "FILE2" and "FORMAT2" refer to the corresponding image of this object;

The attribute "METAFILE" refers to the corresponding "metafile" file.

This element gives the same size information as the element "PIC" for the object image.

```
<!ELEMENT TABLE (ROW|%FLDSTUFF;)* >
<!ELEMENT ROW (CELL|%FLDSTUFF;)* >
<!ATTLIST ROW
GAPH NUMBER #IMPLIED
LEFTPOS CDATA #IMPLIED
HEADER CDATA #IMPLIED
ALIGN CDATA #IMPLIED>
```

The element "TABLE" consists of a series of rows. Each row is made up of a series of cells and can have the following attributes:

1) GAPH specifies half the distance between cells

2) LEFTPOS indicates the left position of the row

3) HEADER indicates whether or not it is a header row

4) ALIGN specifies the alignment ("L", "R" or "C")

```
<!ELEMENT CELL ( PIC | P | LIST | %FLDSTUFF; )* >
<!ATTLIST CELL
HPOS CDATA #IMPLIED
RowIndex NUMBER #IMPLIED ColumnIndex NUMBER #IMPLIED
TOP CDATA #IMPLIED BOTTOM CDATA #IMPLIED
LEFT CDATA #IMPLIED RIGHT CDATA #IMPLIED
VALIGN CDATA #IMPLIED
VMERGE CDATA #IMPLIED>
```

A table cell can contain in particular paragraphs, lists and images, and has the following attributes:

1) "HPOS" gives the position of the right border of the cell;

2) "RowIndex" and "ColumnIndex" give the numerical row and column indices;

3) "TOP", "BOTTOM", "LEFT" and "RIGHT" specify whether the cell has these borders;

4) "VALIGN" indicates the cell alignment ("B" for "bottom", "M" for "middle" and "T" by default)

5) "VMERGE" indicates vertical merge and can take the values "FIRST" or "NEXT".

```
<!ELEMENT P ( (DATA,ENDDATA?) | %PARA-ITEM; )* >
<!ATTLIST P
STYLE CDATA #IMPLIED
ALIGN NAME #IMPLIED
LI CDATA #IMPLIED RI CDATA #IMPLIED FI CDATA #IMPLIED
LEVEL NUMBER #IMPLIED
TOP NUMBER #IMPLIED BOTTOM NUMBER #IMPLIED
LEFT NUMBER #IMPLIED RIGHT NUMBER #IMPLIED
DOUBLE NUMBER #IMPLIED
LISTID NUMBER #IMPLIED LISTTYPE CDATA #IMPLIED
LISTNUMB NUMBER #IMPLIED LISTLEV NUMBER #IMPLIED
STARTAT NUMBER #IMPLIED HR NUMBER #IMPLIED>
```

Paragraph elements can have the following attributes:

1) "STYLE" gives the style name of the paragraph;

2) "ALIGN" specifies its alignment and can take the values "L" for left alignment, which is the default value, "R" for right alignment, "C" for centred and "J" when the paragraph is justified;

3) "LI", "RI" and "FI" respectively give left and right indentation, and the indentation of the first line of the paragraph;

4) "LEVEL" indicates the sub-numerical level of nesting;

5) "TOP", "BOTTOM", "LEFT" and "RIGHT" respectively specify whether the paragraph has a border at the top, bottom, left or right;

6) "DOUBLE" specifies whether the border is double or not;

The other attributes relate to list information (see "LIST" element) and are given when the lists are not tagged (see configuration of the programme "`rtf2piv`").

```
<!ELEMENT CS - - ( (DATA,ENDDATA?) | %PARA-ITEM; )* >
<!ATTLIST CS
STYLE CDATA #IMPLIED HIDDEN NUMBER #IMPLIED
FONT CDATA #IMPLIED SIZE NUMBER #IMPLIED
BOLD NUMBER #IMPLIED ITALIC NUMBER #IMPLIED
FULLCAPS NUMBER #IMPLIED SMALLCAPS NUMBER #IMPLIED
SUBSCRIPT NUMBER #IMPLIED SUPERSCRIPT NUMBER #IMPLIED
```

```
STRIKE NUMBER #IMPLIED UL NUMBER #IMPLIED
ULDB NUMBER #IMPLIED ULDASH NUMBER #IMPLIED
ULD NUMBER #IMPLIED
SHADPERC CDATA #IMPLIED SHADFILL CDATA #IMPLIED>
```

The element "CS" is used to tag character strings with different formatting characteristics. Each character string can be encapsulated in a "DATA" element. The presentation attributes for a character string are as follows:

1) "STYLE" gives the style name

2) "HIDDEN" indicates whether the text is hidden

3) "FONT" gives a font name (or an index in a font table)

4) "SIZE" is an integer giving the font size

5) "BOLD" indicates whether the character string is in bold

6) "ITALIC" indicates whether the character string is in italics

7) "FULLCAPS" specifies whether the string is completely in normal capitals

8) "SMALLCAPS" specifies whether the string is completely in small capitals

9) "SUBSCRIPT" indicates whether the string is in subscript

10) "SUPERSCRIPT" indicates whether the string is in superscript

11) "STRIKE", "UL", "ULDB", "ULDASH", "ULD" indicate respectively whether the character string is struck through, underlined, double underlined, underlined with a dashed line, or underlined with a dotted line.

12) "SHADPERC" specifies the percentage of shading

13) "SHADFILL" specifies the shade filling (RGB value such as "0:255:255")

```
<!ELEMENT FLD1 - - (%FLD-CONTENT;)* >
<!ELEMENT FLD2 - - (%FLD-CONTENT;)* >
<!ELEMENT BM-START EMPTY>
<!ATTLIST BM-START NAME CDATA #REQUIRED>
<!ELEMENT BM-END EMPTY>
<!ATTLIST BM-END NAME CDATA #REQUIRED>
<!ELEMENT FOOTNOTE (%TEXT-ITEM;)* >
```

The elements FLD1 and FLD2 give respectively the content of field instruction ("") and the calculated result of the field ("").

Bookmarks are represented by a start element and an end element giving the name of the bookmark; to process these elements, they need to be paired on the basis of their name.

The element "FOOTNOTE" is tagged at the place where the note appears.

```
<!ELEMENT METADATA (PROPERTY*)>
<!ELEMENT property (DATA)>
<!ATTLIST property
NAME CDATA #IMPLIED
TYPE (BUILTIN|CUSTOM|OTHER) "BUILTIN">
```

The element "METADATA" gives a dictionary containing the various document properties. These properties are classed into three groups.

```
<!ELEMENT DATA (#PCDATA)>
<!ATTLIST DATA NP NUMBER #IMPLIED START NUMBER #IMPLIED>
<!ELEMENT ENDDATA EMPTY>
<!ATTLIST ENDDATA END NUMBER #IMPLIED>
```

The element "DATA" is used to surround all character strings. The attribute "NP" gives the corresponding MS Word paragraph number; the attribute "START" gives the position of the first character in the string in relation to the start of the document. The element "ENDDATA" is an empty element that follows the element "DATA" and gives the position of the last character in the string.

```
<!ELEMENT LIST (LISTITEM+)>
<!ATTLIST LIST
LISTID NUMBER #IMPLIED LISTTYPE CDATA #IMPLIED
LISTNUMB NUMBER #IMPLIED LISTLEV NUMBER #IMPLIED
STARTAT NUMBER #IMPLIED HR NUMBER #IMPLIED>
<!ELEMENT LISTITEM (%TEXT-ITEM;)*>
<!ATTLIST LISTITEM NUM CDATA #IMPLIED>
```

Lists are tagged with the elements "LIST" and "LISTITEM". The attribute "NUM" of the element "LISTITEM" gives the text used to number the list. The element "LIST" uses the following attributes:

1) LISTID is the unique numerical identifier of the list

2) LISTTYPE is a numerical code representing the type of list (see RTF documentation)

3) LISTNUMB is used to distinguish between numbered and non-numbered lists

4) STARTAT gives the starting value for a numbered list

5) HR indicates whether the list is numbered according to a hierarchy (e.g.: 1.4.3)

```
<!ELEMENT ULINK (DATA|%FLD-CONTENT;)*>
<!ATTLIST ULINK
BMK CDATA #IMPLIED
HREF CDATA #REQUIRED>
```

The element "ULINK" is used to tag the links; the attribute "HREF" gives the link (URL for example), while the attribute "BMK" is the name of a bookmark within the linked document.

5.7.5.2. The style sheet "piv2tok.xsl"

This style sheet translates the pivot format produced by "rtf2piv" into "tokens", which will form the "terminals" (in compiler jargon) for the parser at the next pass. These tokens are effectively start and end tags, but not all are present.

This style sheet processes the following constructions:

1) generation of a root tag including the default language, which is stored in an additional document property ("metadata");

2) removal of any constructions produced by the Translator's Workbench tool used by translators;

3) removal of all the document properties (the default language is retrieved from the first tag);

4) translation of the "PIC" elements into "graphic" elements and retrieval of the attributes;

5) translation of "reuse" constructions;

6) processing of highlighting (bold, italic);

removal of keywords used to delimit certain parts of the document The style sheet also includes the two style sheets listed below:

1) "gentok.xsl" is a style sheet containing a series of very simple conversion instructions such as:

```
<xsl:template match="P[@STYLE='Figures']">
<xsl:text disable-output-escaping="yes"><bud-data></xsl:text>
<xsl:apply-templates/>
</xsl:template>
```

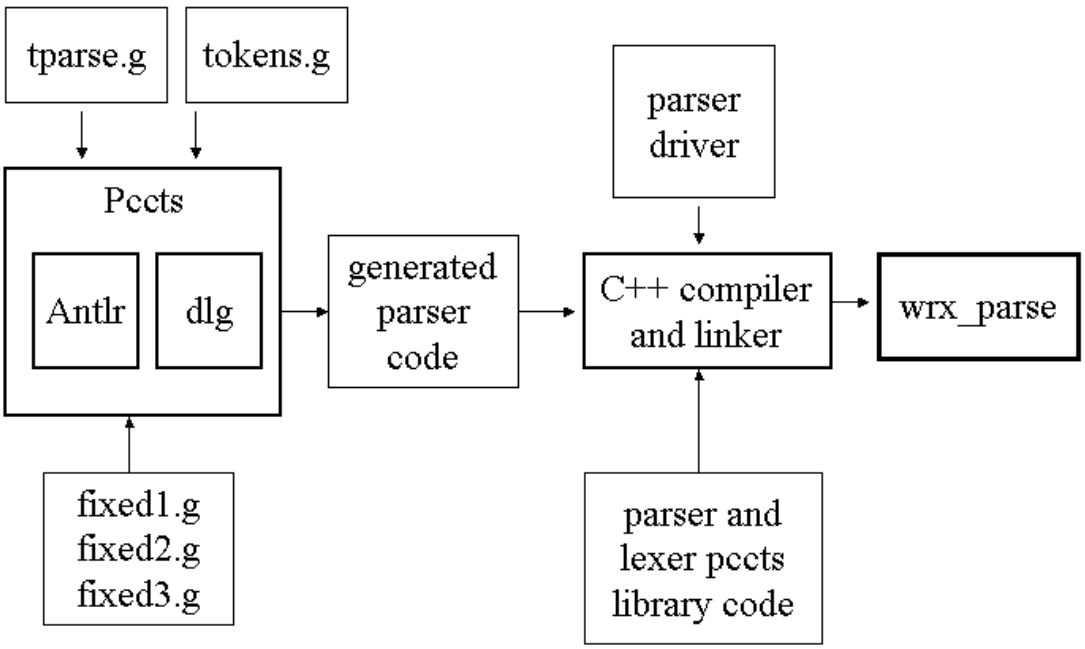
1) "piv2tok-table.xsl" processes table constructions; the titles and subtitles of tables positioned before the table are moved into the table, vertical merges of cells are resolved.

5.7.5.3. The filter "Tok2wrx"

This filter extends the class "`CommandLineFilter`" of the Xef framework. It controls an executable programme ("`wrx_parse`") responsible for parsing or compiling what was produced by the style sheet "`piv2tok.xsl`". This programme can receive two arguments in its input, which are the input and output filenames (by default, these are "`piv2tok.out`" as the input filename and "`piv2tok.xml`" as the output filename). This programme returns an execution status comprising an integer which is the erroneous paragraph number (+ 1) in the event of an error.

This programme is in fact generated from a compiler production suite: PCCTS ("the Purdue Compiler Construction Tool Set"). PCCTS is a set of tools from the public domain used to facilitate the construction of compilers and other processing systems. This system bears partial comparison with the tools Unix YACC and LEX or their equivalents GNU Bison and Flex.

The "`wrx_parse`" programme production can be represented as in the following diagram:



The two PCCTS programmes used are "ant" and "dlg"; "ant" is a specification in the form of several files; it generates a number of C++ files; it also generates a lexical analyser specification file, which is processed by "dlg" to obtain C++ code.

The files that undergo PCCTS processing can be split into two groups: the "fixed" files are independent of the target DTD and contain common lexical and grammatical specifications. The files "tparse.g" (grammatical definition) and "tokens.g" (terminal definition) depend on the target DTD and can be produced either manually or from the DTD using a SAG tool.

The C++ code generated by PCCTS is then compiled, as are the PCCTS utility code and a "driver" that enables the activation arguments such as input file and output file to be processed. This produces an executable programme in native code (supported platforms: Solaris and Windows).

By way of example, here is an extract from the file "tparse.g" relating to the element "bud-remarks" for the "abb" publication:

```
bud_remarks: S_bud_remarks <<write("<bud-remarks>"); >>
( p
| reuse_link
| list
| table
)* <<write("</bud-remarks>"); >>
;
#token S_bud_remarks "<bud-remarks>"
```

The content of the non-terminal "bud_remarks" starts with a terminal "S_bud_remarks", the definition of which is "<bud-remarks>" (this is an XML syntax, but another syntax could have been chosen for the intermediate file generated by the style sheet "piv2tok.xsl"). Then between chevrons, there is some code to be executed after recognising "S_bud_remarks"; this code simply writes the "bud-remarks" starting tag in the output. Then, in brackets, there is a series of non-terminals that form a repeatable choice group. Lastly, the end tag "bud-remarks" is given. Note that in this case, the intermediate file generated by "piv2tok.xsl" does not issue a terminal corresponding to the end tag. The end of the construction "bud_remarks" is inferred by the parser.

5.7.5.4. The style sheet "wrx2prs.xslt"

This style sheet converts a document in the intermediate format "wrx" to the presentation format "prs".

By default, all the elements are copied to the output. The following operations are carried out by the filter:

1) standardisation of the budget data tagging: an empty element is added if necessary for items not containing any budget data; this is true both for "bud-data" and for "bud-data-extra";

2) the prefix (containing the alias) is removed from the item titles;

3) This style sheet also includes the other style sheets listed below:

4) "wrx2prs-btx.xslt" converts highlighting elements and lists

5) "wrx2prs-table.xslt" converts "hpos" cell attributes into "colspec" elements

6) "wrx2prs-grseq.xslt" converts section elements to "gr-seq"

7) "wrx2prs-data.xslt" converts items associated with budget data into schedule or human resources data.

5.7.6. "Presentation" to HTML conversion

This conversion is implemented by the class "`Abb2htm`", which derives from the class "`XSLTFilter`", which is part of the Xef framework.

This filter is one-way, i.e. there is no reverse conversion; it uses the "repo" to presentation conversion "`abb2prs`".

This filter requires a mandatory parameter: the document language.

This conversion filter cascades the following filters:

1) "abb2prs", which is described in an earlier section;

2) the style sheet "`prs2htm.xslt`".

The style sheet "`prs2htm.xslt`" includes the following XSLT modules:

1) "`prs2htm-cals.xslt`" converts the CALS tables into HTML tables;

2) "`prs2wrx-quote.xslt`" contains the definitions for quote marks in the different languages supported;

3) "`stdtxt-templates.xslt`" contains various character strings added during presentation (table headers, etc.), generally depending on the document language; this style sheet is generated automatically by the module "`stdtxt`".

The HTML code produced by this style sheet uses CSS ("Cascading Style Sheet") definitions. These definitions are produced by a template attached to the input document root.

A special process is used to convert "ft" highlight elements, for which the type of highlighting is specified by an attribute. Notes are also processed in a special way, and are contained in a dedicated division (HTML element "div").

The tagging for the presentation of the differential ("diff" tags) is processed in a special way so that insertions/deletions/copying are presented differently.

5.7.7. Processing "standard character strings" ("stdtxt")

Standard character strings such as table headers for budget data are held centrally in an XML file ("stdtxt.xml"); this document is a dictionary, each entry of which contains all the language versions. An entry can also contain a number of parameters.

For example, this is a document containing two entries in two different language versions; the parameter in this case is the year in question:

```
<?xml version="1.0" encoding="utf-8"?>
<stx:library xmlns:stx="" version="1.0">
<stx:entry key="appropriations">
<stx:text lang="en">Appropriations <stx:param name="p1"/></stx:text>
<stx:text lang="fr">Crédits <stx:param name="p1"/></stx:text>
</stx:entry>
<stx:entry key="outturn">
<stx:text lang="en">Outturn <stx:param name="p1"/></stx:text>
<stx:text lang="fr">Exécution <stx:param name="p1"/></stx:text>
</stx:entry>
</stx:library>
```

This document file "stdtxt.xml" is processed using the style sheet "st2xsl.xsl" to produce the style sheet "stdtxt-templates.xslt", which is used to convert configured references to a character string.

For the above example, we will have the following style sheet portion:

```
<xsl:template name="stx-appropriations">
<xsl:param name="p1"/>
<xsl:choose>
<xsl:when test="$lang = 'en'">
<xsl:text>Appropriations </xsl:text>
<xsl:value-of select="$p1"/>
</xsl:when>
<xsl:when test="$lang = 'fr'">
<xsl:text>Crédits </xsl:text>
<xsl:value-of select="$p1"/>
</xsl:when>
<xsl:otherwise>
<xsl:text>?????</xsl:text>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
```

5.7.8. The Full text Translation Format (FTF)

5.7.9. The Tabular Translation Format (TTF)

This translation format was used in production for the first time in March 2004 (PDB 2005). The purpose of this translation format is to allow concurrent authoring and translation. This translation format provides also other advantages, such as:

— The reduction of the translation effort concerning “already translated” translation segments;

— The reduction of the volume (Mb) of information that is exchanged with the translation services;

— The reduction of the production support (multilingual alignment corrections).

Refer to the "Generating translation proposals for The Translation Table Format" subsection of the "Translation Memory" section for more information about this format and how it is produced/

5.7.10. Extracting the XML text from the repository

5.7.10.1.Origine des données

Un amendement SEI-BUD+Amendments se compose de données éparpillées dans différents lieus de stockage :

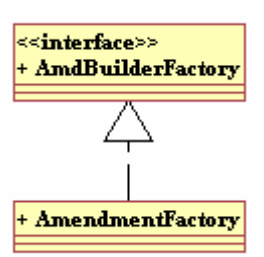
1) Les**données textuelles**sont stockées dans le repository XML. On y trouve les commentaires, les bases légales et les références associées aux transactions de type`Modify`,`Contents`,`Add`,`AddAmd`,`Merge`,`Split`et`SplitAdd`. On n’y trouve pas les transactions`Figure`set`Delete`. Les justificatifs sont également stockés à la fin de ces documents.

2) Les**chiffres**, les**échéanciers,**les**transactions**et toutes les autres informations de gestion concernant les amendements sont stockés dans une base de données Oracle classique.

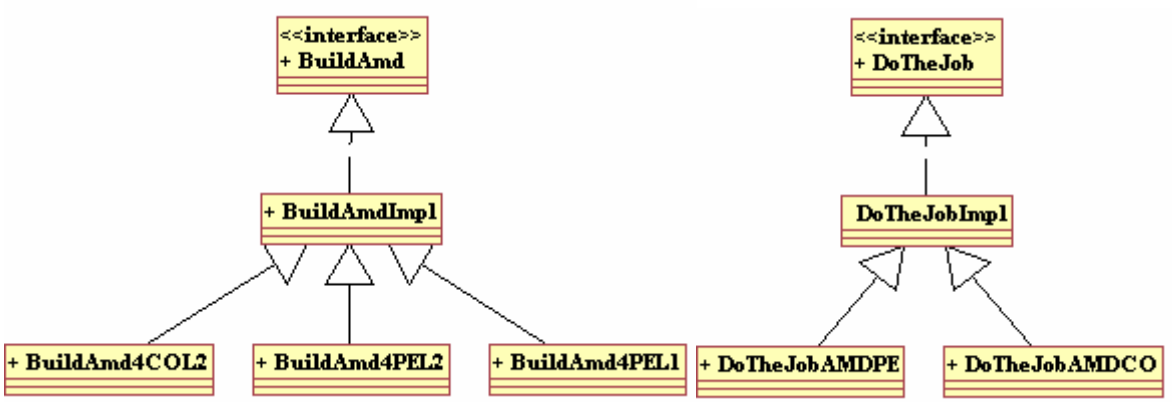
Un package séparé appelé`com.softwareag.belgium.seibud.amdbuilders`'occupe d’extraire les amendements du repository XML et de compléter leur contenu avec les données de gestion issues de la base de données de gestion Oracle.

5.7.10.2. Vue d'ensemble

Le principe de fonctionnement général est que le Repository Manager demande à une fabrique de créer un objet générateur (builder). L'objet se comporte aussi comme un décorateur puisqu'il enrichit les données venant du repository XML à l'aide de ce qui se trouve dans la base de données de gestion. Le repository manager ne connaît que la fabrique ainsi que toute une série d'interfaces. C'est seulement la fabrique qui connaît l'implémentation réelle :



La fabrique est capable d'instancier deux différents types de générateurs :

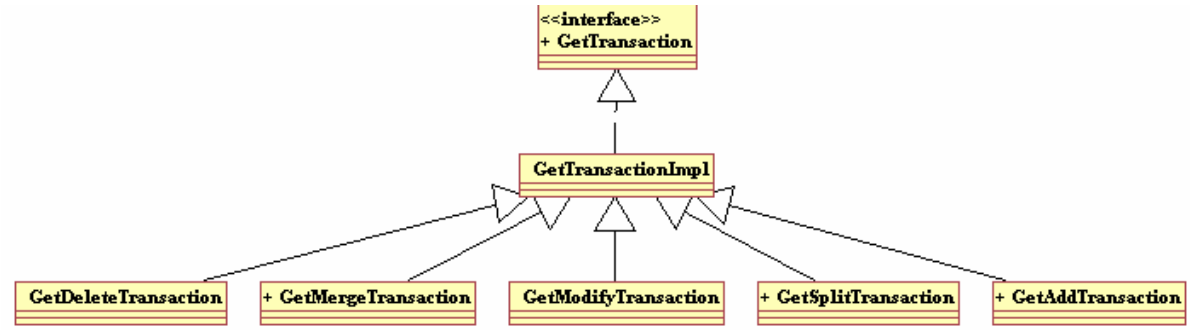


Les générateurs issus de l'interface`BuildAmd`composent la version initiale d'un amendement à partir d'une publication de référence (avant projet de budget ou projet de budget) en vue de le stocker dans le repository XML. Si par exemple les bases légales n'existent pas, on stocke quand même dans l'amendement un élément`bud-legal`vide afin que l'amendement puisse éventuellement le créer. Il en va de même pour tous les fragments optionnels à l'exception des fragments`bud-info`(objectifs généraux) qui ne sont créés que pour les lignes budgétaires de type`nmc-title`(titre). Cette façon de procéder implique qu'au moment où l'on intègre l'amendement dans une publication budgétaire, il faut de nouveau supprimer tous les éléments vides.

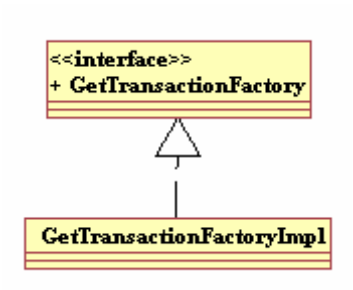
Les générateurs issus de l'interface`DoTheJob`composent une instance XML à partir d'un ou plusieurs amendements déjà stockés dans le repository XML et à partir des données de gestion.

5.7.10.3. Détail des classes

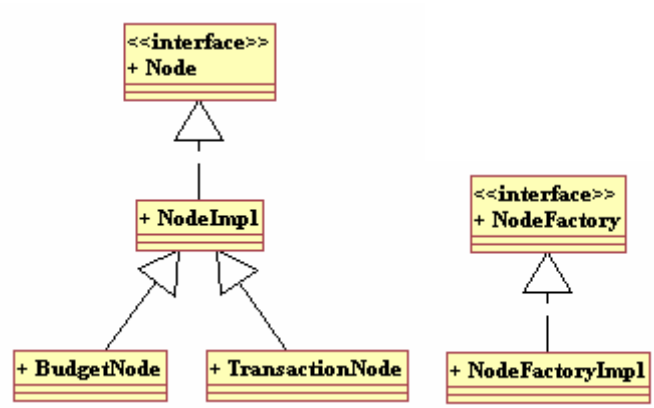
Toutes les extractions spécifiques aux transactions sont gérées dans des classes dédiées. On trouve une classe par type de transaction:



La création de ces objets est déléguée à une fabrique :

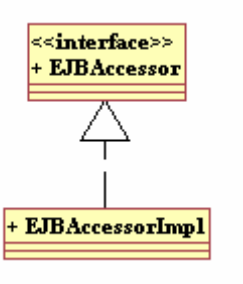


Toutes les transactions portent sur des lignes budgétaires. Dans`amdbuilder`, on fait la distinction entre des lignes budgétaires issues de la publication de référence (`BudgetNode`) et les lignes budgétaires faisant l'objet d'un amendement(`TransactionNode`). Suivant les cas, les chemins d'accès sont différents, mais les opérations disponibles sont identiques. Les deux classes héritent donc d'un même ancêtre et possèdent une fabrique qui leur est propre :

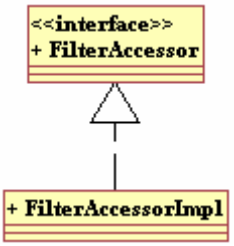


Il est important de noter que amdbuilder effectue lui-même les extractions dans le repository XML, mais que toutes les données de gestion en provenance de la base de données Oracle doivent être fournies par le Repository Manager. Voilà pourquoi, avant d'appeler amdbuilder, le Repository Manager doit rassembler toutes les données de gestion concernant un amendement dans une instance de la classe`SAmendment`. Pour que amdbuilder puisse travailler, l'objet`SAmendment`doit être complet. Il doit contenir les transactions, les chiffres et toutes les autres informations pertinentes.

Pour toutes les données de la base de données de gestion qui ne se trouvent pas dans un objet`SAmendment`, le Repository Manager doit fournir un adaptateur qui permet d'y accéder:



Afin que amdbuilder puisse accéder aux transformations XSLT du système Xef, le Repository Manager fournit un deuxième adaptateur:



Dans ces cas`EJBAccessor`et`FilterAccessor`sont définis par amdbuilder, mais les implémentations,`EJBAccessorImpl`et`FilterAccessorImpl`, sont fournies par le Repository Manager.

Finalement, toute une série de classes assure le gros du travail d'extraction et de composition:

Interface	Implémentation	Description
ExtractServices	ExtractServicesImpl	Le travail principal de cette interface est de préparer l'extraction des transactions dans le repository XML. Les unités d'extraction sont les suivantes : 1) le titre, élément`bud-heading`, 2) les objectifs généraux, élément`bud-intro`, 3) les remarques, éléments`bud-remarks`ou`bud-text`(ce dernier élément est spécifique à une ligne budgétaire de type`nmc-grseq`), 4) les bases légales, élément`bud-legal`, 5) les actes de référence, élément`bud-reference`, 6) le justificatif d'un amendement, élément`justification`(spécifique à SEI-BUD+Amendments et donc, ne se retrouve que dans les amendements et pas dans les publications de référence), 7) l'information en texte libre, élément`info`(spécifique à SEI-BUD+Amendments et donc, ne se retrouve que dans les amendements et pas dans les publications de référence). Le plus souvent, une méthode de cette interface retourne un fragment de document. Le fragment peut être stocké dans une chaîne de caractères, dans un flot d'entrée Java ou dans un fichier. L'interface offre donc chaque méthode en trois exemplaires selon le format de sortie désiré. Par exemple, les méthodes`extractAmendmentDataInStream`,`extractAmendmentDataInString`et`extractAmendmentDataInFile`ont toutes les trois comme mission d'extraire les données présentes en un seul exemplaire dans l'amendement : le justificatif et l'information en texte libre. La première dépose le texte résultant dans un flot d'entrée, la deuxième appelle la première et dépose le flot d'entrée dans une chaîne de caractères et la troisième appelle la première et déverse ensuite le résultat dans un fichier.
MlRepoExtract	MlRepoExtractImpl	Cette interface forme l'unique point d'accès au repository XML. Son implémentation est la seule à utiliser les opérations de Xef. La famille de méthodes`DoMlExtraction`est dédiée à l'extraction de fragments dans le repository XML. Les méthodes`DoMlExtraction`se distinguent par l'origine du fragment à extraire et le format de sortie.

		Les méthodes acceptant un paramètre de type `SAmendment` vont rechercher le texte dans les amendements. Les méthodes acceptant un paramètre du type `SBudgetFragmentReference` vont rechercher le texte dans les publications de référence. Concernant le format de sortie, on peut soit spécifier un nom de fichier ou un flot de sortie Java. Le dernier paramètre de cette famille de méthodes est aussi très important. Il est appelé `defaultNotExisting`. Il règle en effet le comportement de la méthode au cas où le fragment n'existe pas. Si la longueur de la chaîne de caractères est nulle, une exception est levée lorsqu'on ne trouve pas le fragment recherché. Si par contre la chaîne contient au moins un caractère, les méthodes `DoMlExtraction` supposent que lorsque le fragment n'existe pas, on veut retourner un élément vide dont le nom est justement la valeur du paramètre. L'autre méthode importante de `MlRepoExtraction` est `getDifference`. C'est elle qui produit les fragments FT, RT ou TT. Le type de texte produit dépend en fait du paramètre `differenceType`. Ce paramètre reprend un des noms de différentiels définis dans le fichier `document-repository-config.xml` du `Repository Manager` : 1) `xml-diff-text-amd` : génère le format RT, 2) `xml-diff-full-amd` : génère le format FT avec marques de révision ; 3) `xml-diff-translators-amd` : génère la liste des modifications pour le format TT devant être appliquée à une instance de la langue cible. L'implémentation `MlRepoExtractImpl` réalise également une optimisation très importante : elle garde un cache des fragments déjà extraits. En effet, une extraction d'une publication de référence est très coûteuse en temps et prend au moins quelques secondes. Dès lors, un cache avec les fragments déjà extraits permet de faire passer la durée de l'extraction de quelques secondes à une fraction de seconde. Ce gain est considérable surtout si l'on pense qu'un amendement peut se composer d'un nombre de fragments proportionnel au nombre de transactions !
Services	ServicesImpl	La classe implémentant cette interface assure le principal travail de composition. C'est elle qui par exemple demande la génération des données de gestion au début de l'amendement et qui ordonnance les différentes transactions. Concernant les transactions, elle n'effectue que le travail commun à tous les types de transactions. Le travail spécifique est délégué à une sous-classe de `GetTransaction`. La méthode principale est `getTransactionsOrMapaAndFigsummary`. C'est cette méthode qui parcourt toutes les transactions et qui génère le texte XML correspondant.
TextCalculation	TextCalculationImpl	Cette interface permet de produire le texte XML en format FT, RT ou TT au niveau de l'unité d'extraction (v. ExtractServices). Les méthodes servent de relais pour la méthode `MlRepoExtract.getDifference` qui exécute réellement le travail. Les méthodes de TextCalculation ne font rien d'autre que d'exposer une interface plus conviviale à MlRepoExtract. La méthode `processTTFilter` est la seule qui va un peu plus loin. Au résultat de la méthode `getDifference`, elle applique encore le filtre de conversion donné par `FilterAccessor.processTTFilter` pour générer un fragment au bon format TT.
XMLGenerator	XMLGeneratorImpl	Cette interface assure le formatage des données de gestion au format XML.

5.7.10.4. Sur la manière d'appeler AmendmentFactory

Pour un amendement donné, amdbuilder peut générer des instances XML totalement différentes en fonction de l'effet recherché. Ce qui suit énumère les cas les plus importants.

Création du contenu initial d'un amendement

En gros, la création du contenu XML initial d'un amendement se fait en trois étapes.

1) Il faut d'abord appeler la méthode`setNewAmendmentParameters`pour fixer les bons paramètres ;

2) ensuite, il faut obtenir un objet`BuildAmd`en appelant la méthode`getInitialAmendmentBuilder` ;

3) finalement, il faut exécuter la méthode`initialDataAmendement`de l'objet`BuildAmd`.

En détail, les paramètres de`setNewAmendmentParameters`sont :

1) `SAmendment amendment`: données de gestion concernant le nouvel amendement; cet amendement doit être complet et contenir toutes les informations concernant les transactions, les chiffres, les déposants, les amendements dont il est issu etc. ; il doit obligatoirement être fourni ;

2) `SAmendment amdToCopyJustification`: amendement existant duquel doit être copié le justificatif ; peut être nul si le justificatif ne doit pas être copié

3) `SAmendment amdPEL1`: seulement utilisé pour les amendements des phases Conseil deuxième lecture et Parlement deuxième lecture; indique l'amendement de première lecture qui correspond au nouvel amendement ; il faut qu'il contienne les informations concernant les transactions, mais il n'y a pas lieu d'y inclure les informations concernant les déposants et les éventuels amendements dont il est issu;

4) `String baseLanguageId`: la langue de base de l'amendement; doit toujours être fourni ;

5) `String phaseId`: identifiant de la phase dans laquelle l'amendement doit être crée ; doit toujours être fourni ;

6) `String originalLanguage`: dans le cas d'une copie d'amendement, fournir la langue qui doit être générée en priorité ;

7) `boolean useTextFromFile`: si vaut`true`, l'amendement possède des textes d'édition immédiate qui viennent remplacer le texte de la publication de référence ;

8) `boolean restoreThePDB` : si vrai, on veut rétablir les chiffres de l'avant-projet de budget ; cela provoque le placement d'un texte standard dans le justificatif ;

9) `File tmpDir` : répertoire qui devra accueillir les fichiers temporaires ; il faut toujours fournir une valeur ;

10) `XMLDispatcherContext xdCtx` : contexte utilisé par le Repository Manager ; il faut toujours fournir un contexte valide.

La méthode`getInitialAmendmentBuilder`crée le bon objet`BuildAmd`en fonction de l'institution et de la lecture indiquées dans le paramètre`amendment`.

Extraction en format FT, RT ou TT

L'extraction d'un amendement existant a également lieu en trois étapes:

— Il faut d'abord appeler la méthode`setExistingAmendmentParameters`pour fixer les bons paramètres ;

— ensuite, il faut obtenir un objet`DoTheJob`en appelant la méthode`getAmendmentBuilder` ;

— finalement, il faut exécuter la méthode`startTheJob`de l'objet DoTheJob.

Les paramètres de `setExistingAmendmentParameters` sont :

1) `SAmendment amendment`: données de gestion concernant l'amendement; cet amendement doit être complet et contenir toutes les informations concernant les transactions, les chiffres, les déposants, les amendements dont il est issu etc. ; il doit obligatoirement être fourni ;

2) `SAmendment amdPEL1`: seulement utilisé pour les amendements des phases Conseil deuxième lecture et Parlement deuxième lecture; indique l'amendement de première lecture qui correspond à l'amendement ; il faut qu'il contienne les informations concernant les transactions, mais il n'y a pas lieu d'y inclure les informations concernant les déposants et les éventuels amendements dont il est issu;

3) `String languageId`: la langue à extraire; la langue vaut "`OR`" pour obtenir l'original; en cas de demande de TT, ce paramètre contient la langue source;

4) `String targetLanguageId`: seulement utilisé pour obtenir des instances XML pour les traducteurs au format TT ; contient alors la langue cible ;

5) `String pointOfView`: `Parliament` ou `Council` suivant que l'institution demandant l'instance XML est le Parlement européen ou le Conseil;

6) `File tmpDir`: répertoire qui devra accueillir les fichiers temporaires ; il faut toujours fournir une valeur ;

7) `XMLDispatcherContext xdCtx` : contexte utilisé par le Repository Manager ; il faut toujours fournir un contexte valide ;

8) `FilterAccessor filterAccessor` : implémentation de l'interface `FilterAccessor` capable de terminer la génération d'une instance XML au format TT.

La méthode `getAmendmentBuilder` crée le bon objet `DoTheJob` en fonction de l'institution courante associée à l'amendement.

La méthode `DoTheJob.startTheJob` a également besoin de quelques paramètres pour pouvoir générer l'instance XML attendue:

1) `String tpContent`: vaut `AmendmentFactory.CONTENT_AMD` ou `AmendmentFactory.CONTENT_BUDGET` ; la première valeur est la valeur normale; elle indique que les textes de la valeur courante de l'amendement doivent être recherchés dans l'amendement même; la deuxième valeur ne peut être utilisée que pour compléter la valeur initiale d'un amendement de compromis ou d'un amendement contenant des transactions `Add` ; le paramètre tpFormat doit dans ce dernier cas absolument valoir `AmendmentFactory.FORMAT_FT` ; la valeur spéciale indique que le contenu de l'amendement doit être récupéré en priorité des lignes budgétaires indiquées dans les objets `ContentTransaction` énumérés dans l'amendement ; pour les transactions ADD, il s'agit de lignes de la publication de référence qui forment la valeur initiale de la nouvelle ligne budgétaire créée par amendement ; pour les amendements de compromis, les lignes sont issus d'autres amendements (puisqu'on on fait un compromis entre des plusieurs amendements existant déjà).

2) `String tpFormat`: contient une des valeurs `AmendmentFactory.FORMAT_FT`, `AmendmentFactory.FORMAT_RT` ou `AmendmentFactory.FORMAT_TT` ;

3) `String targetLanguageId`: langue du document devant être généré ;

4) `int mode` : n'est plus utilisé

5) `boolean XFT` : si la valeur vaut `true`, il faut absolument que tpFormat vaille `AmendmentFactory.FORMAT_RT` ; c'est la valeur à utiliser pour obtenir une instance FT avec marques de révision ;

6) `String[] headerLanguages` : est nul en général ; ce paramètre n'est utilisé que pour les rapports "cahier chiffres" et "liste de votes" ; si on donne une liste d'identifiants de langue (FR, DE, IT etc.), l'instance XML contiendra des titres supplémentaires des lignes budgétaires dans les langues spécifiées.

Amendements de compromis ou transactions ADD

Les amendements de compromis et les amendements contenant des transactions ADD ne peuvent pas être crées en utilisant simplement un objet `BuildAmd`. La raison est que les implémentations de cette interface ne sont pas assez intelligentes pour récupérer lors de la création le contenu à d'autres endroits que l'endroit habituel.

En effet, la valeur initiale d'un amendement de compromis est formée par les amendements dont on fait le compromis. De même, si on crée une transaction ADD, on peut spécifier que le contenu initial est issu d'une autre ligne budgétaire.

Les implémentations de `DoTheJob` sont par contre capables de trouver les bons textes.

Voilà pourquoi, la création d'amendements de compromis et d'amendements contenant des transactions ADD se fait en trois étapes :

1) création normale d'un amendement en utilisant un objet BuildAmd ;

2) extraction d'une première version de l'amendement en utilisant DoTheJob.startTheJob avec les paramètres `tpContent` valant `AmendmentFactory.CONTENT_BUDGET` et `tpFormat` valant `AmendmentFactory.FORMAT_FT`;

3) mise à jour de l'amendement avec le texte obtenu à l'étape précédente.

5.7.11. Producing the XML files for Full Text (FT)

Il existe deux sortes de présentations qui portent l'appellation *Full Text*.

Dans son sens le plus strict, le *Full Text* est le document original dans lequel l'auteur intègre les données de l'amendement. Il présente le texte dans sa version la plus récente, sans indiquer quelles modifications y ont été faites.

Il existe également une autre vue *Full Text*, basée sur un différentiel. C'est une vue qui présente à la fois le texte initial et les modifications. Elle est à distinguer du *Reduced Text*, qui ne présente que les modifications.

5.7.11.1.Le Full Text auteur

Le*Full Text*présenté ici est le document d'édition destiné aux auteurs. Il ne contient aucune information sur les modifications qui ont été faites dans le texte. C'est une vue de la version courante de l'amendement qui se trouve dans le système SEI-BUB+Amendments.

Exemple de document Full Text :

Volume 4 (Section 3) - Commission

Poste 05 02 08 04 Mesures spéciales pour les fruits à coque

Modifier les chiffres comme suit:

05 02 08 04	PB 2004		APB 2005		PB 2005		AMENDEMENT		PB+AMENDEMENT	
	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements
Crédits	20 000 000	20 000 000	20 000 000	20 000 000	20 000 000	20 000 000	0	0	20 000 000	20 000 000
Réserves							1 000 000	1 000 000	1 000 000	1 000 000

Intitulé:

Mesures spéciales pour les fruits à coque

Commentaires:

Ce crédit est destiné à couvrir :

— le coût des mesures spécifiques pour le financement, notamment, des aides aux producteurs de noisettes conformément à l'article 55 du règlement (CE) nº 2200/96,

— des aides spécifiques aux organisations de producteurs qui constituent un fonds de roulement et l'aide communautaire aux plans d'amélioration de la qualité des fruits à coque et des caroubes.

Bases légales:

Règlement (CEE) nº 789/89 du Conseil du 20 mars 1989 instaurant des mesures spécifiques pour les fruits à coque et les caroubes, et modifiant le règlement (CEE) nº 1035/72 portant organisation commune des marchés dans le secteur des fruits et légumes (JO L 85 du 30.3.1989, p. 3).

Actes de référence:

Justification:

5.7.11.2.Le différentiel Full Text

Le différentiel*Full Text*présente la totalité du texte initial et les modifications. Il est utilisé dans deux cas de figure:

1) Dans un document*Reduced Text*, l'intitulé, s'il a été modifié, est toujours présenté en*Full Text*. La présentation*Reduced Text*n'est pas adaptée car un intitulé est généralement constitué d'un paragraphe. Une vue*Full Text*apportera une meilleure lisibilité.

2) Durant le collationnement, les traducteurs passent en revue tous les amendements dans toutes les langues et vérifient que toutes les traductions soient synoptiques. Pour faciliter leur travail, un rapport*Full Text*contenant tous les amendements et un document différentiel*Full Text*par amendement leur sont fournis. Ils effectuent le collationnement sur le rapport mais saisissent les modifications dans le document reprenant un seul amendement.

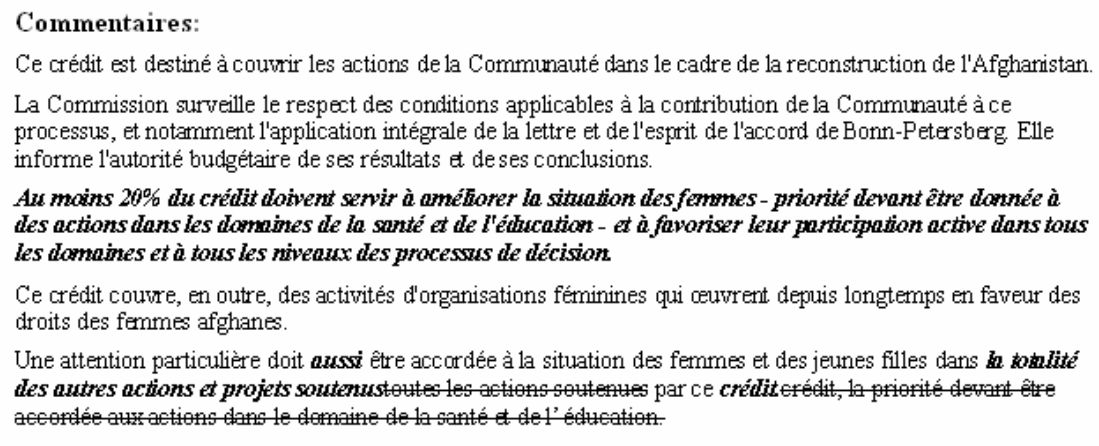
La DTD du Full Text

Le*Full Text*est basée sur la DTD suivante :

Namespace = fulltxt (http://www.softwareag.com/fulltxt)

```
<!ELEMENT fulltxt (#PCDATA) +(delete|insert|update|update-atts)>
<!ELEMENT delete (input)>
<!ELEMENT insert (input)>
<!ELEMENT update (input)>
<!ELEMENT update-atts (att)>
<!ELEMENT input (#PCDATA) +(version)>
<!ELEMENT version (#PCDATA) -(version)>
<!ATTLIST version
v1 CDATA #REQUIRED
v2 CDATA #REQUIRED>
<!ELEMENT att EMPTY>
<!ATTLIST att
name CDATA #REQUIRED
old-value CDATA #REQUIRED
new-value CDATA #REQUIRED>
```

Pour faciliter la compréhension de ces différents éléments, voici la représentation finale d'un différentiel*Full Text*:



Commentaires:

Ce crédit est destiné à couvrir les actions de la Communauté dans le cadre de la reconstruction de l'Afghanistan.

La Commission surveille le respect des conditions applicables à la contribution de la Communauté à ce processus, et notamment l'application intégrale de la lettre et de l'esprit de l'accord de Bonn-Petersberg. Elle informe l'autorité budgétaire de ses résultats et de ses conclusions.

Au moins 20% du crédit doivent servir à améliorer la situation des femmes - priorité devant être donnée à des actions dans les domaines de la santé et de l'éducation - et à favoriser leur participation active dans tous les domaines et à tous les niveaux des processus de décision.

Ce crédit couvre, en outre, des activités d'organisations féminines qui œuvrent depuis longtemps en faveur des droits des femmes afghanes.

Une attention particulière doit ***aussi*** être accordée à la situation des femmes et des jeunes filles dans ***la totalité des autres actions et projets soutenus*** ~~toutes les actions soutenues~~ par ce ***crédit*** ~~crédit, la priorité devant être accordée aux actions dans le domaine de la santé et de l' éducation.~~

Les modifications portent toujours sur des tableaux, des listes ou des paragraphes. On parle de**point de comparaison**. C'est un élément`table`,`list`ou`p`qui n'est pas contenu dans un autre élément`table`,`list`ou`p`, avec pour ordre de priorité`table`, puis`list`puis`p`. Ainsi, un élément`p`qui se trouve dans une liste ou un tableau n'est pas un point de comparaison.

Un**point de comparaison modifié**est un point de comparaison qui a été entièrement ajouté, entièrement supprimé ou qui contient une différence.

Dans l'exemple ci-dessus, les cinq paragraphes sont des points de comparaison. Seuls les troisième et cinquième paragraphes sont des points de comparaison modifiés.

La DTD*Full Text*permet de représenter les trois types de modifications possibles :

1) L'ajout d'un point de comparaison au moyen de l'élément`insert`:

```
<fulltxt:insert>
<fulltxt:input>
<table, list ou p ajouté>…</table, list ou p>
</fulltxt:input>
</fulltxt:insert>
```

1) La suppression par l'élément`delete`:

```
<fulltxt:delete>
<fulltxt:input>
<table, list ou p supprimé>…</table, list ou p>
</fulltxt:input>
```

```
</fulltxt:delete>
```

— La mise à jour par les élémentsupdateetversion. L'élémentupdatecontient le point de comparaison modifié et l'élémentversionsert à englober les parties de texte modifiées. Il a un attributv1, représentant le texte supprimé etv2pour le texte ajouté. Siv1vaut "1", le texte contenu dansversionest supprimé par l'amendement (dans ce cas,v2vaut obligatoirement "0"). Inversement, siv2vaut "1", le texte contenu dans l'élémentversionest ajouté etv1vaut obligatoirement "0" :

```
<fulltxt:update>
<fulltxt:input>
<table, list ou p modifié>
      …
<fulltxt:version v1="1" v2="0">ancien texte</fulltxt:version>
      …
<fulltxt:version v1="0" v2="1">nouveau texte</fulltxt:version>
      …
</table, list ou p>
</fulltxt:input>
</fulltxt:update>
```

Remarque : la modification des attributs est prévue par la DTD (élémentupdate-attsetatt) mais n'est pas utilisée pour l'instant.

Les règles de conversion XDiff vers la DTD Full Text

Le différentiel "Word Marking" de XDiff se caractérise par l'ajout d'un attributdiffaux éléments modifiés. Cet attribut peut avoir les valeurs "att", "ins", "del" ou "copy". Les portions de texte affectées sont également englobées par une balisediffavec le même attributdiff. (cf 4.3 The difference calculation tool).

Afin de passer de ce format à la représentation Full Text détaillée plus haut, des règles de conversion ont été établies :

— Undiff="copy" est équivalent à undiff="ins"

— Traitement d'un point de comparaison modifié ayant un attributdiff:

  — si l'attributdiffvaut "ins" ou "copy" :

    — avant l'élément, insertion des balises<fulltxt:insert><fulltxt:input>

    — après l'élément, insertion des balises</fulltxt:input></fulltxt:insert>

  — si l'attributdiffvaut "del" :

    — avant l'élément, insertion des balises<fulltxt:delete><fulltxt:input>

    — après l'élément, insertion des balises:</fulltxt:input></fulltxt:delete>

  — si l'attributdiffvaut "atts" :

    — on ignore cet attribut

    — suppression de l'attribut dans la sortie (l'élément en lui-même est préservé)

— Traitement d'un point de comparaison modifié n'ayant pas d'attributdiff:

  — avant l'élément, insertion des balises<fulltxt:update><fulltxt:input>

  — après l'élément, insertion des balises:</fulltxt:input></fulltxt:update>

— Traitement d'un élémentdiff:

  — s'il est contenu dans un point de comparaisondeleteouinsert, suppression des balisesdiff

- s'il est contenu dans un point de comparaison`update`et qu'il a dans ses ancêtres un élément dont l'attribut`diff`vaut la même valeur que lui, suppression des balises`diff`
- s'il est contenu dans un point de comparaison`update`et qu'il n'a pas, dans ses ancêtres, un élément dont l'attribut`diff`vaut la même valeur que lui :
  - si l'attribut`diff`vaut "ins" ou "copy" :
    - avant l'élément, insertion des balises`<fulltxt:version v1="0" v2="1">`
    - après l'élément, insertion des balises`</fulltxt:version>`
    - suppression des balises`diff`
  - si l'attribut`diff`vaut "del" :
    - avant l'élément, insertion des balises`<fulltxt:version v1="1" v2="0">`
    - après l'élément, insertion des balises`</fulltxt:version>`
    - suppression des balises`diff`
- Traitement d'un élément avec attribut`diff`qui n'est pas l'élément de plus haut niveau d'un point de comparaison :
  - s'il est contenu dans un point de comparaison`delete`ou`insert`, suppression de l'attribut`diff`
  - s'il est contenu dans un point de comparaison`update`et qu'il a dans ses ancêtres un élément dont l'attribut`diff`vaut la même valeur que lui, suppression de l'attribut`diff`
  - s'il est contenu dans un point de comparaison`update`et qu'il n'a pas, dans ses ancêtres, un élément dont l'attribut`diff`vaut la même valeur que lui :
    - si l'attribut`diff`vaut "ins" ou "copy" :
      - avant l'élément, insertion des balises`<fulltxt:version v1="0" v2="1">`
      - après l'élément, insertion des balises`</fulltxt:version>`
      - suppression de l'attribut`diff`
    - si l'attribut`diff`vaut "del" :
      - avant l'élément, insertion des balises`<fulltxt:version v1="1" v2="0">`
      - après l'élément, insertion des balises`</fulltxt:version>`
      - suppression de l'attribut`diff`
    - si l'attribut`diff`vaut "atts":
      - on ignore cet attribut
      - suppression de l'attribut`diff`

Le processus de production du différentiel

Pour produire le différentiel*Full Text*, on suit les étapes suivantes :

— On détermine les numéros de version à comparer. Ici, il faut comparer la version courante du budget de référence et la version courante du repository XML[8].

— Extraction (en lecture uniquement) des deux fragments de document à comparer.

— Comparaison des deux fragments de documents dans le mode "Word Marking" de XDiff.

— Formatage du différentiel suivant le "formateur de différentiel" spécifié dans le fichier de configuration Xef "document-repository-config.xml". Le but est de passer vers la DTD*Full Text*.

— Le XML*Full Text*est retourné au client.

La définition du "formateur de différentiel" dans le fichier de configuration est la suivante :

```
<diff-format>
<name>xml-diff-full-amd</name>
<descr>XML FT format for single basic text fragments< / descr>
<filename-extension>xml</filename-extension>
<supported-schema>
<schema-ref>amd</schema-ref>
<supported-views global-view-supported="y">
<supported-view>global</supported-view>
</supported-views>
</supported-schema>
<diff-formatter>com.softwareag.belgium.seibud.amd.AmdDiffFTFormatter</ diff-formatter>
</ diff-format>
```

Le formateur de différentiel s'intitule "xml-diff-full-amd". Il est représenté par la classe`AmdDiffFTFormatter`, qui est une extension de la classe abstraite`AbstractDiffFormatter`du framework Xef. Xef permet de définir un filtre de présentation (qui est appliqué sur les fragments de documents à comparer) et un filtre de formatage (qui est appliqué sur le résultat de la comparaison). Dans le cas présent, aucun filtre de présentation n'est nécessaire, et le filtre de formatage effectue la conversion XDiff vers la DTD*Full Text*.

Le filtre de formatage est constitué de deux feuilles de styles :

— findFTUnits.xsl (FindFTUnits.java, hérite de la classe`XSLTFilter`appartenant au framework Xef)

— generateVersion.xsl (GenerateFTVersion.java, hérite aussi de`XSLTFilter`).

findFTUnits.xsl est la première des deux feuilles de style à être appliquée sur le différentiel. Son but est d'encapsuler les points de comparaison modifiés dans les éléments*Full Text*appropriés. Pour cela, tous les éléments`table`,`list`et`p`de l'instance sont examinés. Sur chacun, on détermine s'il s'agit d'un point de comparaison, et le cas échéant, s'il est modifié. Si ces deux critères se vérifient, on encapsule le point de comparaison dans un élément :

— `fulltxt:inserts`'il a un attribut`diff`de valeur "ins" ou "copy" (règle de conversion 2a)

— `fulltxt:deletes`'il a un attribut`diff`de valeur "del" (règle de conversion 2b)

— `fulltxt:updates`sinon.

Prenons par exemple ce différentiel XDiff :

```
<bud-remarks>
```

[8]Dans le cas d'une transaction qui ajoute une ligne budgétaire, on compare la version courante avec un document XML ne contenant qu'un élément racine, de telle sorte que la version courante apparaisse comme entièrement ajoutée.

```
<p>Ce crédit est destiné à couvrir les actions de la Communauté dans le cadre de la
reconstruction de l'Afghanistan.</p>
<p>La Commission surveille le respect des conditions applicables à la contribution de la
Communauté à ce processus, et notamment l'application intégrale de la lettre et de l'esprit de
l'accord de Bonn-Petersberg. Elle informe l'autorité budgétaire de ses résultats et de ses
conclusions.</p>
<p diff="ins">
<diff diff="ins">Au moins 20% du crédit doivent servir à améliorer la situation des femmes -
priorité devant être donnée à des actions dans les domaines de la santé et de l'éducation - et
à favoriser leur participation active dans tous les domaines et à tous les niveaux des
processus de décision.< / diff>
</p>
<p>Ce crédit couvre, en outre, des activités d'organisations féminines qui œuvrent depuis
longtemps en faveur des droits des femmes afghanes.</p>
<p>Une attention particulière doit <diff diff="ins">aussi< / diff> être accordée à la
situation des femmes et des jeunes filles dans <diff diff="ins">la totalité des autres actions
et projets soutenus< / diff><diff diff="del">toutes les actions soutenues< / diff> par ce
<diff diff="ins">crédit.< / diff><diff diff="del">crédit, la priorité devant être accordée aux
actions dans le domaine de la santé et de l’éducation.< / diff></p>
</bud-remarks>
```

Après findFTUnits.xsl :

```
<bud-remarks xmlns:fulltxt="http://www.softwareag.com/fulltxt">
<p>Ce crédit est destiné à couvrir les actions de la Communauté dans le cadre de la
reconstruction de l'Afghanistan.</p>
<p>La Commission surveille le respect des conditions applicables à la contribution de la
Communauté à ce processus, et notamment l'application intégrale de la lettre et de l'esprit de
l'accord de Bonn-Petersberg. Elle informe l'autorité budgétaire de ses résultats et de ses
conclusions.</p>
<fulltxt:insert>
<fulltxt:input>
<p diff="ins">
<diff diff="ins">Au moins 20% du crédit doivent servir à améliorer la situation des femmes -
priorité devant être donnée à des actions dans les domaines de la santé et de l'éducation - et
à favoriser leur participation active dans tous les domaines et à tous les niveaux des
processus de décision.< / diff>
</p>
</fulltxt:input>
</fulltxt:insert>
<p>Ce crédit couvre, en outre, des activités d'organisations féminines qui œuvrent depuis
longtemps en faveur des droits des femmes afghanes.</p>
<fulltxt:update>
<fulltxt:input>
<p>Une attention particulière doit <diff diff="ins">aussi< / diff> être accordée à la
situation des femmes et des jeunes filles dans <diff diff="ins">la totalité des autres actions
et projets soutenus< / diff><diff diff="del">toutes les actions soutenues< / diff> par ce
<diff diff="ins">crédit.< / diff><diff diff="del">crédit, la priorité devant être accordée aux
actions dans le domaine de la santé et de l’éducation.< / diff></p>
</fulltxt:input>
</fulltxt:update>
</bud-remarks>
```

Seuls les troisième et cinquième paragraphes sont des points de comparaison modifiés. Le troisième paragraphe porte un attribut`diff`de valeur "ins", il est donc encapsulé dans un`fulltxt:insert`. Le dernier paragraphe n'a pas d'attribut`diff`, il est encapsulé dans un`fulltxt:update`.

Après findFTUnits.xsl, generateVersion.xsl est appliquée. Son but est de créer les éléments`fulltxt:version`. Les principes suivis sont les suivants :

— pas d'élément`version`dans un élément`fulltxt:insert`ou`fulltxt:delete`. La DTD l'interdit (règles de conversion 4a et 5a)

— dans un élément`fulltxt:update`:

— les éléments ayant un attribut`diff`de valeur "ins" ou "copy" deviennent des`<fulltxt:version v1="0" v2="1">`(règles de conversion 4c.i. et 5c.i.)

— les éléments`<diff diff="del">`deviennent des`<fulltxt:version v1="1" v2="0">`(règles de conversion 4c.ii. et 5c.ii.)

Remarque : l'élément`version`existe également dans la DTD du*Reduced Text*. La feuille de style generateVersion.xsl est donc commune à la génération du*Full Text*et du*Reduced Text*. Pour décider si les éléments version doivent porter le namespace`fulltxt`ou`redtxt`, on passe à la feuille de style un paramètre format, de valeur "FT" ou "RT".

Après generateVersion.xsl, l'exemple précédent devient conforme à la DTD*Full Text*:

```
<bud-remarks xmlns:fulltxt="http://www.softwareag.com/fulltxt">
<p>Ce crédit est destiné à couvrir les actions de la Communauté dans le cadre de la
reconstruction de l'Afghanistan.</p>
<p>La Commission surveille le respect des conditions applicables à la contribution de la
Communauté à ce processus, et notamment l'application intégrale de la lettre et de l'esprit de
l'accord de Bonn-Petersberg. Elle informe l'autorité budgétaire de ses résultats et de ses
conclusions.</p>
<fulltxt:insert>
<fulltxt:input>
<p>Au moins 20% du crédit doivent servir à améliorer la situation des femmes - priorité devant
être donnée à des actions dans les domaines de la santé et de l'éducation - et à favoriser
leur participation active dans tous les domaines et à tous les niveaux des processus de
décision.</p>
</fulltxt:input>
</fulltxt:insert>
<p>Ce crédit couvre, en outre, des activités d'organisations féminines qui œuvrent depuis
longtemps en faveur des droits des femmes afghanes.</p>
<fulltxt:update>
<fulltxt:input>
<p>Une attention particulière doit <fulltxt:version v1="0" v2="1">aussi </fulltxt:version>être
accordée à la situation des femmes et des jeunes filles dans <fulltxt:version v1="0" v2="1">la
totalité des autres actions et projets soutenus</fulltxt:version><fulltxt:version v1="1"
v2="0">toutes les actions soutenues</fulltxt:version> par ce <fulltxt:version v1="0"
v2="1">crédit.</fulltxt:version><fulltxt:version v1="1" v2="0">crédit, la priorité devant être
accordée aux actions dans le domaine de la santé et de l’éducation.</fulltxt:version></p>
</fulltxt:input>
</fulltxt:update>
</bud-remarks>
```

5.7.12. Producing the XML files for Reduced Text (RT)

Le*Reduced Text*(RT) est un document non modifiable (accessible en lecture seule) destiné au contrôle de l'amendement. Contrairement au*Full Text*, il n'inclut pas le texte intégral des lignes budgétaires modifiées. Il signale uniquement les modifications apportées par l'auteur.

5.7.12.1.La DTD du Reduced Text

Le*Reduced Text*est basée sur la DTD suivante :

Namespace = redtxt (http://www.softwareag.com/redtxt)

```
<!ELEMENT trn (elem-loc,blk+)>
<!ELEMENT elem-loc EMPTY>
<!ELEMENT blk (loc, (delete|insert|update)+)>
<!ELEMENT loc (anchor)? >
<!ATTLIST loc
pos (BEFORE|AFTER|WITHIN) #REQUIRED
>
<!ELEMENT anchor (#PCDATA)>
<!ELEMENT delete (input) -(version)>
```

```
<!ELEMENT insert (input) -(version)>
<!ELEMENT update (input)>
<!-- Only allow VERSION tag inside an UPDATE -->
<!ELEMENT input (#PCDATA) +(version)>
<!ELEMENT version (#PCDATA) -(version)>
<!ATTLIST version v1 (0|1) #REQUIRED
v2 (0|1) #REQUIRED>
```

Pour faciliter la compréhension de ces différents éléments, voici la représentation finale d'un*Reduced Text*:

Remarks:

Before paragraph

The main objective of the program is to help shift .....the anticipated growth of international road haulage to other modes.

Amend text as follows:

This appropriation is intended to cover expenditure on ~~implementation~~ ***implementation*** of a program to promote alternatives to international road haulage, designated Marco Polo.

After paragraph

The main objective of the program is to help shift .....the anticipated growth of international road haulage to other modes.

Amend text as follows:

Three types of action which complement each other are proposed:

— start-up support for new non-road freight transport services, which should be viable in the mid-term,

— support for launching freight services ***or facilities*** of strategic European interest,

— stimulating cooperative behavior in the freight logistics market.

Add following text:

The Marco Polo program will also be able to fund actions involving countries which are candidates for accession to the European Union.

Les notions de**points de comparaison**et de**points de comparaison modifiés**introduites lors de la présentation du*Full Text*sont également applicables pour le*Reduced Text*. Mais pour le*Reduced Text*, on fonctionne par**groupe de points de comparaison modifiés**, c'est-à-dire par succession de points de comparaison modifiés, consécutifs, fils du même élément XML.

Comme on peut le voir dans l'exemple ci-dessus, les modifications sont présentées par groupe. Afin de faciliter la lecture, les groupes sont localisés dans le document par rapport à un "point d'ancrage". Le point d'ancrage est toujours un point de comparaison non modifié. Prioritairement, on localise un groupe de modifications par rapport au paragraphe/liste/tableau qui le précède. S'il n'existe pas un tel élément, on localise le groupe par rapport au paragraphe/liste/tableau qui le suit. S'il n'existe toujours pas un tel élément, cela signifie que tous les points de comparaison ont été modifiés. Une localisation n'est donc pas nécessaire.

Dans l'exemple ci-dessus, il y a deux groupes de points de comparaison modifiés :

— Le premier est constitué d'un seul paragraphe : "This appropriation is intended… designated Marco Polo". Il est localisé par rapport au paragraphe qui le suit dans le document : "Before paragraph...".

— Le deuxième est constitué d'une liste et d'un paragraphe ("The Marco Polo program…European Union"). Il est localisé par rapport au paragraphe qui le précède : "After paragraph…".

L'élément`blk`représente un groupe de points de comparaison modifiés. Il contient un élément`loc`, pour décrire le point d'ancrage et sa position, et une suite d'éléments`sudpate`,`delete`et/ou`insert`, un par point de comparaison modifié.

L'élément`loc`a un attribut`pos`, pour indiquer si le groupe est localisé par rapport à un élément`table/list/p`qui le précède (valeur "after"), qui le suit (valeur "before") ou s'il n'est pas localisé (valeur "within"). Le point d'ancrage est englobé dans un élément`anchor`.

Les éléments`supdate`,`delete`,`insert`et`version`ont le même sens que celui expliqué dans la partie sur le*Full Text*.

Si, dans l'exemple précédent, on s'intéresse au deuxième groupe de points de comparaison modifiés, on peut en déduire que sa représentation "XML Reduced Text" aura la structure :

```
<redtxt:blk>
<redtxt:loc pos="after">
<redtxt:anchor>
<p>…</p>
</redtxt:anchor>
</redtxt:loc>
<redtxt:update>
<redtxt:input>
<list>…</list>
</redtxt:input>
</redtxt:update>
<redtxt:insert>
<redtxt:input>
<p>…</p>
</redtxt:input>
</redtxt:insert>
</redtxt:blk>
```

5.7.12.2.Les règles de conversion XDiff vers la DTD Reduced Text

Afin de passer du format "Word Marking" de XDiff à la représentation*Reduced Text*détaillée ci-dessus, des règles de conversion ont été établies :

— Le fichier RT de sortie débute par :`<redtxt:trn><redtxt:elem-loc/>`

— Le fichier RT de sortie se termine par :`</redtxt:trn>`

— Un`diff`="copy" est équivalent à un`diff`="ins"

— Un groupe de points de comparaison modifiés débute par :

  Si le groupe est précédé par un point de comparaison non modifié`An`ayant le même parent que le groupe :`<redtxt:blk><redtxt:loc pos="after"><redtxt:anchor>point de comparaison non modifié An</redtxt:anchor></redtxt:loc>`

  Sinon, si le groupe est suivi par un point de comparaison non modifié`Zn`ayant le même parent que le groupe :`<redtxt:blk><redtxt:loc pos="before"><redtxt:anchor>point de comparaison non modifié Zn</redtxt:anchor></redtxt:loc>`

  Sinon:`<redtxt:blk><redtxt:loc pos="within"></redtxt:loc>`

— Un groupe de points de comparaison modifiés se termine par :`</redtxt:blk>`

— Traitement d'un point de comparaison modifié D ayant un attribut diff :

  si l'attribut`diff`vaut "ins" ou "copy":

  avant l'élément, insertion des balises`<redtxt:insert><redtxt:input>`

après l'élément, insertion des balises`</redtxt:input></redtxt:insert>`

si l'attribut`diff`vaut "del":

avant l'élément, insertion des balises `<redtxt:delete><redtxt:input>`

après l'élément, insertion des balises`</redtxt:input></redtxt:delete>`

si l'attribut`diff`vaut "atts" :

on ignore cet attribut

suppression de l'attribut dans la sortie

— Traitement d'un point de comparaison modifié D n'ayant pas d'attribut`diff`:

avant l'élément, insertion des balises`<redtxt:update><redtxt:input>`

après l'élément, insertion des balises`</redtxt:input></redtxt:update>`

— Traitement d'un élément`diff`:

s'il est contenu dans un point de comparaison`delete`ou`insert`, on supprime les balises`diff`

s'il est contenu dans un point de comparaison`update`et qu'il a dans ses ancêtres un élément dont l'attribut`diff`vaut la même valeur que lui, on supprime les balises`diff`

s'il est contenu dans un point de comparaison`update`et qu'il n'a pas, dans ses ancêtres, un élément dont l'attribut`diff`vaut la même valeur que lui :

si l'attribut`diff`vaut "ins" ou "copy" :

avant l'élément, insertion des balises`<redtxt:version v1="0" v2= "1">`

après l'élément, insertion des balises`</redtxt:version>`

suppression des balises`diff`

si l'attribut`diff`vaut "del" :

avant l'élément, insertion des balises`<redtxt:version v1="1" v2="0">`

après l'élément, insertion des balises`</redtxt:version>`

suppression des balises`diff`

— Traitement d'un élément avec un attribut`diff`qui n'est pas l'élément de plus haut niveau d'un point de comparaison :

s'il est contenu dans un point de comparaison`delete`ou`insert`, on supprime l'attribut`diff`

s'il est contenu dans un point de comparaison`update`et qu'il a dans ses ancêtres un élément dont l'attribut`diff`vaut la même valeur que lui, on supprime l'attribut`diff`

s'il est contenu dans un point de comparaison`update`et qu'il n'a pas, dans ses ancêtres, un élément dont l'attribut`diff`vaut la même valeur que lui :

si l'attribut`diff`vaut "ins" ou "copy" :

avant l'élément, insertion des balises`<redtxt:version v1="0" v2="1">`

après l'élément, insertion des balises`</redtxt:version>`

suppression de l'attribut`diff`

si l'attribut`diff`vaut "del" :

avant l'élément, insertion des balises`<redtxt:version v1="1" v2="0">`

après l'élément, insertion des balises`</redtxt:version>`

suppression de l'attribut`diff`

si l'attribut`diff`vaut "atts" :

on ignore cet attribut

suppression de l'attribut`diff`

5.7.12.3.Le processus de production du différentiel

Pour produire le différentiel*Reduced Text*, on suit les étapes suivantes :

— Détermination des numéros de versions à comparer. Dans le cas du*Reduced Text*, il faut comparer la version courante du repository XML avec le budget de référence[9].

— Extraction (en lecture uniquement) des deux fragments de documents à comparer.

— Comparaison des deux fragments de documents dans le mode "Word Marking" de XDiff.

— Formatage du différentiel suivant le "formateur de différentiel" spécifié dans le fichier de configuration Xef "document-repository-config.xml". Le but est de passer vers la DTD*Reduced Text*.

— Le XML*Reduced Text*est renvoyé au client.

La définition du "formateur de différentiel" dans le fichier de configuration est la suivante :

```
<diff-format>
<name>xml-diff-text-amd</name>
<descr>XML RT format for single basic text fragments< / descr>
<filename-extension>xml</filename-extension>
<supported-schema>
<schema-ref>amd</schema-ref>
<supported-views global-view-supported="y">
<supported-view>global</supported-view>
</supported-views>
</supported-schema>
<diff-formatter>com.softwareag.belgium.seibud.amd.AmdDiffRTFormatter</ diff-formatter>
</ diff-format>
```

Le formateur de différentiel s'intitule "xml-diff-text-amd". Il est représenté par la classe`AmdDiffRTFormatter`, qui est une extension de la classe abstraite`AbstractDiffFormatter`du framework Xef. Le filtre de formatage défini dans`AmdDiffRTFormatter`permet d'effectuer la conversion XDiff vers la DTD Reduced Text.

Le filtre de formatage est constitué de trois feuilles de styles qui s'enchaînent dans cet ordre :

— findRTUnits.xsl (FindRTUnits.java)

— generateRT.xsl (GenerateRT.java)

— generateVersion.xsl (GenerateRTVersion.java)

---

[9]Dans le cas d'une transaction qui ajoute une ligne budgétaire, on compare la version courante avec un document XML ne contenant qu'un élément racine, de telle sorte que la version courante apparaisse comme entièrement ajoutée.

La feuille de style findRTUnits.xsl

Le but de la feuille de style findRTUnits.xsl est de rechercher les points de comparaison de l'instance XML et de déterminer leur rôle. Pour cela, sur chaque élément`table`,`list`ou`p`rencontré, on se pose les questions suivantes :

— est-ce un point de comparaison ?

— ce point de comparaison est-il modifié ?

— s'il est modifié :

— fait-il partie d'un groupe ?

— marque-t-il le début de ce groupe ?

— marque-t-il la fin de ce groupe ?

— peut-il être rattaché à un point de comparaison non modifié ? (détermination du point d'ancrage)

— s'il n'est pas modifié :

— est-il directement suivi d'un groupe de points de comparaison modifiés ?

— est-il directement précédé d'un groupe de points de comparaison modifiés ?

Une fois toutes ces hypothèses évaluées pour chacun des éléments`table`,`list`et`p`, le résultat est ajouté dans l'instance XML sous la forme d'attributs à ces éléments.

Les attributs ajoutés par findRTUnits.xsl sont :

— sur les points de comparaison modifiés :

— `blk-start`: attribut ajouté sur l'élément`table`,`list`ou`p`qui est le premier élément d'un groupe de points de comparaison modifiés. Sa valeur est de 1.

— `blk-end`: attribut ajouté sur l'élément`table`,`list`ou`p`qui est le dernier élément d'un groupe de points de comparaison modifiés. Sa valeur est de 1.

— `in-blk`: attribut ajouté sur un élément`table`,`list`ou`p`qui appartient à un groupe de points de comparaison modifiés, et qui n'est ni le premier, ni le dernier élément du groupe. Sa valeur est de 1.

— `transac`: Il détermine la type de la modification et prépare à l'ajout des éléments`redtxt:delete`,`redtxt:insert`et`redtxt:update`dans l'étape suivante. Les valeurs possibles pour l'attribut`transac`sont :

— "insert" : si XDiff a déterminé que cet élément`table`,`list`ou`p`avait été entièrement ajouté

— "delete" : si XDiff a déterminé que cet élément`table`,`list`ou`p`avait été entièrement supprimé

— "update" : si l'élément`table`,`list`ou`p`n'a été ni entièrement ajouté, ni entièrement supprimé.

— `loc-within`: attribut ajouté sur le premier élément`table`,`list`ou`p`d'un groupe de points de comparaison modifié, qui n'est ni directement précédé, ni directement suivi par un point de comparaison non modifié.

— sur les points de comparaison non modifiés :

— `loc-after`: attribut ajouté sur un élément`table`,`list`ou`p`qui est un point de comparaison non modifié et qui est directement suivi par un groupe de points de comparaison modifiés (c'est l'élément`An`de la règle 4a)

— `loc-before`: attribut ajouté sur un élément`table`,`list`ou`p`qui est un point de comparaison non modifié et qui est directement précédé par un groupe de points de comparaison modifiés, à la condition que ce groupe de points de comparaison modifiés ne soient pas déjà précédé par un point de comparaison non modifié (règle 4 : les loc "after" sont prioritaires par rapport aux loc "before")

Remarques :

— les attributs`blk-start`et`blk-end`ne sont pas exclusifs : un élément peut marquer à la fois le début et la fin d'un groupe de points de comparaison modifiés

— les attributs`loc-after`et`loc-before`ne sont pas exclusifs : un élément peut servir à la fois de point d'ancrage pour le groupe qui le précède et pour le groupe qui le suit

Exemple :

Si l'on reprend l'exemple de document*Reduced Text*présenté plus haut, le différentiel produit par XDiff était le suivant :

```
<bud-remarks>
<p>This appropriation is intended to cover expenditure on <diff diff="ins">implementation< /
diff><diff diff="del">implementation< / diff> of a programme to promote alternatives to
international road haulage, designated Marco Polo.</p>
<p>The main objective of the programme is to help shift an amount of cargo corresponding to
the anticipated growth of international road haulage to other modes.</p>
<list type="dash">
<int.li>
<p>Three types of action which complement each other are proposed:</p>
</int.li>
<item>
<p>start-up support for new non-road freight transport services,</p>
</item>
<item>
<p>support for launching freight services<diff diff="ins"> or facilities< / diff> of strategic
European interest,</p>
</item>
<item>
<p>stimulating cooperative behaviour in the freight logistics market.</p>
</item>
</list>
<p diff="ins">
<diff diff="ins">The Marco Polo programme will also be able to fund actions involving
countries which are candidates for accession to the European Union.< / diff>
</p>
<p>This appropriation also covers dissemination activities and accompanying measures.</p>
</bud-remarks>
```

Après la feuille de style findRTUnits.xsl

```
<bud-remarks>
<p blk-start="1" blk-end="1" transac="update">This appropriation is intended to cover
expenditure on <diff diff="ins">implementation< / diff><diff diff="del">implementation< /
diff> of a programme to promote alternatives to international road haulage, designated Marco
Polo.</p>
<p loc-after="1" loc-before="1">The main objective of the programme is to help shift an amount
of cargo corresponding to the anticipated growth of international road haulage to other
modes.</p>
<list blk-start="1" transac="update" type="dash">
<int.li>
<p>Three types of action which complement each other are proposed:</p>
</int.li>
<item>
<p>start-up support for new non-road freight transport services,</p>
</item>
<item>
```

```
<p>support for launching freight services<diff diff="ins"> or facilities< / diff> of strategic
European interest,</p>
</item>
<item>
<p>stimulating cooperative behaviour in the freight logistics market.</p>
</item>
</list>
<p blk-end="1" transac="insert" diff="ins">
<diff diff="ins">The Marco Polo programme will also be able to fund actions involving
countries which are candidates for accession to the European Union.< / diff>
</p>
<p>This appropriation also covers dissemination activities and accompanying measures.</p>
</bud-remarks>
```

Explications des attributs ajoutés sur les points de comparaison :

— Premier élément`p`

C’est un point de comparaison modifié.

Il n’est pas précédé d’un autre point de comparaison modifié, il marque donc le début d’un groupe => ajout de l’attribut`blk-start`et pas d’attribut`in-blk`.

Il n’est pas suivi d’un point de comparaison modifié, il marque donc la fin du groupe => ajout de l’attribut`blk-end`.

Il est suivi d’un point de comparaison non modifié, qui lui servira de point d’ancrage => pas d’attribut`loc-within`.

Il n’est ni entièrement ajouté, ni entièrement supprimé, l’attribut`transac`qui lui est ajouté vaut donc "update".

— Deuxième élément`p`

C’est un point de comparaison non modifié.

Il est précédé d’un point de comparaison modifié, qui lui-même n’est précédé d’aucun point de comparaison non modifié. Il sert donc de point d’ancrage pour le pour le groupe qui le précède => ajout de l’attribut`loc-before`

Il est suivi d’un point de comparaison modifié, il sert donc également de point d’ancrage pour ce groupe => ajout de l’attribut`loc-after`

— La liste

C’est un point de comparaison modifié.

Elle n’est pas précédée d’un autre point de comparaison modifié, elle marque donc le début d’un groupe => ajout de l’attribut`blk-start`et pas d’attribut`in-blk`.

Elle est suivie d’un point de comparaison modifié => pas d’attribut`blk-end`.

Elle est précédée d’un point de comparaison non modifié, qui lui servira de point d’ancrage => pas d’attribut`loc-within`.

Elle n’est ni entièrement ajoutée, ni entièrement supprimée, l’attribut`transac`qui lui est ajouté vaut donc "update".

— Troisième élément`p`

Il est précédé d'un point de comparaison modifié, il ne marque donc pas le début d'un groupe => pas d'attribut`blk-start`et pas d'attribut`loc-within`.

Il n'est pas suivi d'un point de comparaison modifié, il marque donc la fin du groupe => ajout d'un attribut`blk-end`et pas d'attribut`in-blk`.

Il est entièrement ajouté, l'attribut`transac`qui lui est ajouté vaut donc "insert".

— Quatrième élément`p`

C'est un point de comparaison non modifié.

Il est précédé d'un groupe de points de comparaison modifiés, mais il existe déjà un élément qui sert de point d'ancrage à ce groupe => pas d'attribut`loc-before`

Il n'est pas suivi d'un point de comparaison modifié => pas d'attribut`loc-after`

La feuille de style generateRT.xsl

Le but de la feuille de style generateRT.xsl est de créer les éléments`strn`,`elem-loc`,`blk`,`delete`,`insert`,`update`,`loc`,`anchor`et`input`de la DTD du*Reduced Text*sur base des attributs ajoutés par findRTUnits.xsl (le seul élément restant,`version`, est créé dans la dernière étape).

L'élément`trn`est toujourrs l'élément racine du fichier de sortie. Son premier fils est l'élément vide`elem-loc`.

Certains des attributs ajoutés lors de l'étape précédente permettent de créer directement des éléments du*Reduced Text*:

— Un élément`table`,`list`ou`p`qui porte un attribut`transac`de valeur "update" sera englobé dans les éléments`redtxt:update/redtxt:input`:

```
<redtxt:update>
<redtxt:input>
<table, list ou p></table, list ou p>
</redtxt:input>
</redtxt:update>
```

— Un élément`table`,`list`ou`p`qui porte un attribut`transac`de valeur "insert" sera englobé dans les éléments`redtxt:insert/redtxt:input`:

```
<redtxt:insert>
<redtxt:input>
<table, list ou p></table, list ou p>
</redtxt:input>
</redtxt:insert>
```

— Un élément`table`,`list`ou`p`qui porte un attribut`transac`de valeur "delete" sera englobé dans les éléments`redtxt:delete/redtxt:input`:

```
<redtxt:delete>
<redtxt:input>
<table, list ou p></table, list ou p>
</redtxt:input>
</redtxt:delete>
```

Les attributs`loc-before`,`loc-after`et`loc-within`de findRTUnits.xsl servent à la création du point d'ancrage (éléments`redtxt:loc`et`redtxt:anchor`).

Les attributs`loc-xxx`déterminent la valeur de l'attribut`pos`de`redtxt:loc`:

— `pos`vaut "BEFORE" en cas d'attribut`loc-before`

— `pos`vaut "AFTER" en cas d'attribut`loc-after`

— `pos`vaut "WITHIN" en cas d'attribut`loc-within`.

Le contenu de l'élément`redtxt:anchor`est l'élément`table`,`list`oupqui porte l'attribut`loc-xxx`.

Les points de comparaison ne portant aucun des attributs ajoutés lors de la première feuille de style seront ici supprimés de la sortie. A l'issue de cette étape, on ne conservera que les paragraphes, listes et/ou tableaux modifiés et leur point d'ancrage, ce qui est la caractéristique principale d'un document*Reduced Text*.

Si on reprend l'exemple précédent, voici le résultat obtenu en sortie de generateRT.xsl :

```
<redtxt:trn xmlns:redtxt="http://www.softwareag.com/redtxt">
<redtxt:elem-loc/>
<redtxt:blk>
<redtxt:loc pos="before">
<redtxt:anchor>
<p>The main objective of the programme is to help shift an amount of cargo corresponding to
the anticipated growth of international road haulage to other modes.</p>
</redtxt:anchor>
</redtxt:loc>
<redtxt:update>
<redtxt:input>
<p>This appropriation is intended to cover expenditure on <diff diff="ins">implementation< /
diff><diff diff="del">implementation< / diff> of a programme to promote alternatives to
international road haulage, designated Marco Polo.</p>
</redtxt:input>
</redtxt:update>
</redtxt:blk>
<redtxt:blk>
<redtxt:loc pos="after">
<redtxt:anchor>
<p>The main objective of the programme is to help shift an amount of cargo corresponding to
the anticipated growth of international road haulage to other modes.</p>
</redtxt:anchor>
</redtxt:loc>
<redtxt:update>
<redtxt:input>
<list type="dash">
<int.li>
<p>Three types of action which complement each other are proposed:</p>
</int.li>
<item>
<p>start-up support for new non-road freight transport services,</p>
</item>
<item>
<p>support for launching freight services<diff diff="ins"> or facilities< / diff> of strategic
European interest,</p>
</item>
<item>
<p>stimulating cooperative behaviour in the freight logistics market.</p>
</item>
</list>
</redtxt:input>
</redtxt:update>
<redtxt:insert>
<redtxt:input>
<p diff="ins">
<diff diff="ins">The Marco Polo programme will also be able to fund actions involving
countries which are candidates for accession to the European Union.< / diff>
```

```
</p>
</redtxt:input>
</redtxt:insert>
</redtxt:blk>
</redtxt:trn>
```

La feuille de style generateVersion.xsl

Le but de la feuille de style generateVersion.xsl est de créer les éléments`version`. Cette tâche est déléguée à une feuille de style particulière car`version`appartient aux deux DTDs du*Full Text*et du*Reduced Text*. Pour les explications, voir la partie sur le Full Text.

Le résultat final est :

```
<redtxt:trn xmlns:redtxt="http://www.softwareag.com/redtxt">
<redtxt:elem-loc/>
<redtxt:blk>
<redtxt:loc pos="before">
<redtxt:anchor>
<p>The main objective of the programme is to help shift an amount of cargo corresponding to
the anticipated growth of international road haulage to other modes.</p>
</redtxt:anchor>
</redtxt:loc>
<redtxt:update>
<redtxt:input>
<p>This appropriation is intended to cover expenditure on <redtxt:version v1="0"
v2="1">implementation</redtxt:version><redtxt:version v1="1"
v2="0">implemenattion</redtxt:version> of a programme to promote alternatives to international
road haulage, designated Marco Polo.</p>
</redtxt:input>
</redtxt:update>
</redtxt:blk>
<redtxt:blk>
<redtxt:loc pos="after">
<redtxt:anchor>
<p>The main objective of the programme is to help shift an amount of cargo corresponding to
the anticipated growth of international road haulage to other modes.</p>
</redtxt:anchor>
</redtxt:loc>
<redtxt:update>
<redtxt:input>
<list type="dash">
<int.li>
<p>Three types of action which complement each other are proposed:</p>
</int.li>
<item>
<p>start-up support for new non-road freight transport services,</p>
</item>
<item>
<p>support for launching freight services<redtxt:version v1="0" v2="1"> or
facilities</redtxt:version> of strategic European interest,</p>
</item>
<item>
<p>stimulating cooperative behaviour in the freight logistics market.</p>
</item>
</list>
</redtxt:input>
</redtxt:update>
<redtxt:insert>
<redtxt:input>
<p>The Marco Polo programme will also be able to fund actions involving countries which are
candidates for accession to the European Union.</p>
</redtxt:input>
</redtxt:insert>
</redtxt:blk>
</redtxt:trn>
```

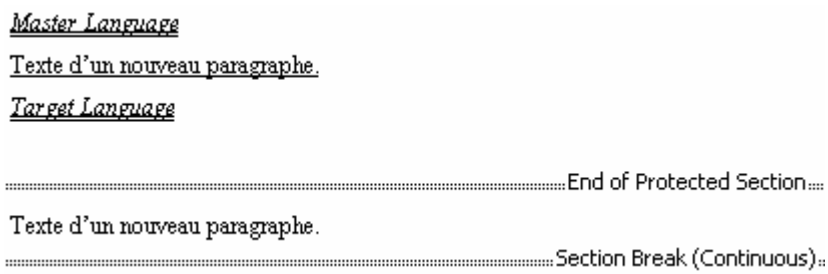
5.7.13. Producing the XML files for Translation Text (TT)

Le*Translation Text*est le document de travail des traducteurs. Il est partiellement modifiable:

— des sections protégées présentent les parties de texte qui ne doivent pas être traduites: les données linguistiquement neutres (chiffres, etc.), les parties de texte non modifiées, et une version non modifiable du texte à traduire inséré dans la langue source (Master language) à l'endroit de la modification;

— des sections non protégées incluent le texte à traduire inséré dans la langue source (Master language) à l'endroit de la modification.

Le texte à traduire est affiché deux fois (seule la seconde version est modifiable, la première sert de mémoire du texte à traduire) :



5.7.13.1.La DTD du Translation Text

Le*Translation Text*est basé sur la DTD suivante :

Namespace = transltxt (http://www.softwareag.com/transltxt)

```
<!ELEMENT update (masterl,targetl)>
<!ELEMENT masterl (input)>
<!ELEMENT targetl (input)>
<!ELEMENT input (#PCDATA) +(version)>
<!ELEMENT version   (#PCDATA) -(version)>
<!ATTLIST version v1 CDATA #REQUIRED
v2 CDATA #REQUIRED>
```

L'**unité de traduction**est l'unité élémentaire devant être traduite, c'est-à-dire qui contient une modification. Dans le cas de "SEIB-UD+Amendment", il peut s'agir :

— D'un paragraphe (p)

— D'une liste (list) ou d'un item de liste (item)

— D'un tableau (table), d'un titre (title), sous-titre (subtitle), ou d'une ligne (row) de tableau.

C'est toujours l'unité la plus "fine" qui est présentée à la traduction. Ainsi, dans une liste, les unités de traduction sont prioritairement les items. La liste n'est une unité de traduction que si elle a été entièrement ajoutée.

L'élément`update`contient une unité de traduction. Il a deux éléments fils car, comme expliqué plus haut, le texte à traduire est affiché deux fois. Son premier fils est un élément`masterl`, qui contient l'unité de traduction dans la langue source avec le texte supprimé barré et le texte ajouté en gras-italique (ce sera la version non modifiable). Son deuxième fils est l'élément`targetl`, qui contient également l'unité de traduction dans la langue source mais sans formatage (ce sera la version modifiable).

L'élément`version`a le même rôle que dans les DTDs du*Full Text*et du*Reduced Text*, c'est-à-dire celui d'englober les parties de texte modifiées, en indiquant lesquelles sont ajoutées et lesquelles sont supprimées au moyen des attributs`v1`et`v2`.

5.7.13.2.Le processus de production du XML*Translation Text*

Le processus de production du XML*Translation Text*se fait en deux étapes. Il est basé sur le synoptisme entre toutes les versions courantes de toutes les langues. En effet, dans le repository XML, si un auteur effectue une modification de structure, celle-ci est automatiquement reportée sur toutes les autres langues lors de la mise à jour. Cette caractéristique est grandement utilisée dans la génération du TT.

Dans une première étape, on travaille uniquement sur les versions courantes et initiales de la langue source. On utilise les fonctionnalités de calcul de différentiel et de filtre de formatage fournies par le framework Xef pour dresser une liste de toutes les unités de traduction du document.

Pour chaque unité trouvée, la liste doit contenir :

— L'élément`update`complet (c'est-à-dire avec`input`,`masterl`,`target`et`version`) correspondant à l'unité de traduction,

— Le chemin ("xpath") de l'unité de traduction dans l'arbre XML, qui servira d'identifiant unique dans la deuxième étape.

Dans la deuxième étape, on travaille sur la version courante de la langue cible. On parcourt les éléments susceptibles d'être des unités de traductions (`p`,`table`,`title`,`subtitle`,`row`,`list`et`item`) et pour chacun, on regarde s'il existe dans la liste dressée à l'étape précédente, un élément`update`qui porte le même "xpath". Le cas échéant, on remplace cet élément par l'élément`update`de la liste. Ceci n'est possible qu'à cause du synoptisme qui existe entre toutes les versions courantes de toutes les langues.

Génération de la liste des unités de traduction

Pour produire le différentiel, on suit les étapes suivantes :

— Détermination des numéros de versions à comparer. Ici il faut comparer les versions initiales et courantes de la langue source.

— Extraction (en lecture uniquement) des deux fragments de documents à comparer.

— Comparaison des deux fragments de documents dans le mode "Word Marking" de XDiff.

— Formatage du différentiel suivant le "formateur de différentiel" spécifié dans le fichier de configuration Xef "document-repository-config.xml". Le but est de générer la liste des unités de traduction.

— La liste est renvoyée au client.

La définition du "formateur de différentiel" dans le fichier de configuration est la suivante :

```
<diff-format>
<name>xml-diff-translators-amd</name>
<descr>XML TT format for single basic text fragments< / descr>
<filename-extension>xml</filename-extension>
<supported-schema>
<schema-ref>amd</schema-ref>
<supported-views global-view-supported="y">
<supported-view>global</supported-view>
</supported-views>
</supported-schema>
```

```
<diff-formatter>com.softwareag.belgium.seibud.amd.AmdDiffTTFormatter</ diff-formatter>
</ diff-format>
```

Il s'intitule "xml-diff-translators-amd" et il est représenté par la classe`AmdDiffTTFormatter`, qui est une extension de la classe abstraite`AbstractDiffFormatter`du framework Xef.

Le filtre de formatage est constitué de deux feuilles de style :

— prepareTT_applyDelete.xsl (PrepareTT_applyDelete.java)

— prepareTT_findAllChanges.xsl (PrepareTT_findAllChanges.java)

Le but de prepareTT_applyDelete.xsl est de supprimer du différentiel XDiff toutes les modifications de type "suppression d'un élément de structure". Ceci permet de retrouver une instance XML synoptique par rapport à la version courante de l'amendement. Sans cette opération, les "xpath" de la liste ne seraient pas corrects.

Seules les suppressions d'éléments de structure sont appliquées. Quand la suppression porte sur du texte, elle est conservée, car cette information va aider le traducteur dans son travail.

Par exemple, dans les commentaires suivants, seul le premier paragraphe doit être supprimé :

```
<bud-remarks>
<p diff="del">
<diff diff="del">Ce paragraphe a été supprimé< / diff>
</p>
<p>Ce paragrahe n'a pas été modifié</p>
<p>Ce paragraphe a été modifié.<diff diff="ins">Texte inséré< / diff><diff diff="del">Texte
supprimé< / diff> bbb.</p>
</bud-remarks>
```

En sortie de prepareTT_applyDelete.xsl :

```
<bud-remarks>
<p>Ce paragrahe n'a pas été modifié</p>
<p>Ce paragraphe a été modifié.<diff diff="ins">Texte inséré< / diff><diff diff="del">Texte
supprimé< / diff> bbb.</p>
</bud-remarks>
```

Une fois que l'on est assuré du synoptisme, on peut créer la liste des unités de traduction, ce qui est fait par la feuille de style prepareTT_findAllChanges.xsl.

prepareTT_findAllChanges.xsl fonctionne sur les principes suivants :

— Un tableau ne peut être une unité de traduction que s'il est entièrement ajouté. Si ce n'est pas le cas, alors on regarde si son titre, ses sous-titres et ses lignes sont des unités de traduction.

— Une liste ne peut être une unité de traduction que si elle est entièrement ajoutée, ou si son introductif (`int.li`) a été ajouté ou modifié. Si ce n'est pas le cas, alors on regarde si ses items sont des unités de traduction.

— Un paragraphe ne peut être une unité de traduction que s'il est modifié et qu'il n'appartient ni à une liste, ni à un tableau.

La liste générée a la structure suivante : l'élément racine s'intitule`changes`et il contient une suite d'éléments`change`, un par unité de traduction. Chaque élément`change`contient un fils`where`où l'on met le xpath et un fils`insert`où l'on met l'élément`update`créé.

Par exemple, si l'on prend le différentiel suivant :

```
<bud-remarks>
<list type="dash">
```

```
<int.li>
<p>Ce crédit est destiné à couvrir les dépenses relatives à des actions menées par la
Commission pour mettre en œuvre la législation en vigueur, les mesures de sensibilisation et
les autres mesures générales basées sur le programme d’action de la Communauté en faveur de
l’environnement, ces actions étant axées sur:</p>
</int.li>
<item>
<p>la mise en œuvre effective de la législation environnementale en vigueur,<diff diff="ins">
en tenant plus particulièrement compte des besoins des nouveaux États membres,< / diff></p>
</item>
<item>
<p>l’intégration des préoccupations environnementales dans les autres politiques
communautaires,</p>
</item>
<item>
<p>la collaboration avec le marché à travers les entreprises et les consommateurs, en vue de
favoriser la mise en place de modes de production et de consommation plus durables,</p>
</item>
<item>
<p>le souci de mettre à la disposition des Européens des informations fiables et accessibles
sur l’environnement,</p>
</item>
<item>
<p>le développement d’une mentalité plus respectueuse de l’environnement en matière
d’utilisation des sols.</p>
</item>
</list>
<p diff="del">
<diff diff="del">Les actions comprendront des subventions et des contrats de services
concernant des projets, des ateliers et des séminaires, la couverture des frais de préparation
et de production de documents audiovisuels, de manifestations et d’expositions, de missions de
presse, de publications et autres activités de diffusion, notamment sur l’internet.< / diff>
</p>
<p>Une approche en matière de stratégies thématiques sera élaborée dans le but de concourir
d’une manière efficace et économique à la réalisation des objectifs environnementaux. Cette
approche s’appliquera à toute la gamme des problèmes d’environnement.</p>
<p diff="ins">
<diff diff="ins">Ce crédit est également destiné à financer un service d'assistance Natura
2000.< / diff>
</p>
</bud-remarks>
```

L'auteur de l'amendement a ajouté du texte dans le premier item de la liste, supprimé le paragraphe qui suit la liste et ajouté un dernier paragraphe.

Après prepareTT_applyDelete.xsl, le paragraphe qui suit la liste a été supprimé :

```
<bud-remarks>
<list type="dash">
<int.li>
<p>Ce crédit est destiné à couvrir les dépenses relatives à des actions menées par la
Commission pour mettre en œuvre la législation en vigueur, les mesures de sensibilisation et
les autres mesures générales basées sur le programme d’action de la Communauté en faveur de
l’environnement, ces actions étant axées sur:</p>
</int.li>
<item>
<p>la mise en œuvre effective de la législation environnementale en vigueur,<diff diff="ins">
en tenant plus particulièrement compte des besoins des nouveaux États membres,< / diff></p>
</item>
<item>
<p>l’intégration des préoccupations environnementales dans les autres politiques
communautaires,</p>
</item>
<item>
<p>la collaboration avec le marché à travers les entreprises et les consommateurs, en vue de
favoriser la mise en place de modes de production et de consommation plus durables,</p>
</item>
<item>
```

```
<p>le souci de mettre à la disposition des Européens des informations fiables et accessibles
sur l’environnement,</p>
</item>
<item>
<p>le développement d’une mentalité plus respectueuse de l’environnement en matière
d’utilisation des sols.</p>
</item>
</list>
<p>Une approche en matière de stratégies thématiques sera élaborée dans le but de concourir
d’une manière efficace et économique à la réalisation des objectifs environnementaux. Cette
approche s’appliquera à toute la gamme des problèmes d’environnement.</p>
<p diff="ins">
<diff diff="ins">Ce crédit est également destiné à financer un service d'assistance Natura
2000.< / diff>
</p>
</bud-remarks>
```

La liste produite par prepareTT_findAllChanges.xsl contient les deux unités de traduction de ce document, c’est-à-dire le premier item de la liste et le dernier paragraphe :

```
<changes>
<change>
<where>/bud-remarks[1]/list[1]/item[1]</where>
<insert>
<transltxt:update>
<transltxt:masterl>
<transltxt:input>
<item>
<p>la mise en œuvre effective de la législation environnementale en vigueur,<transltxt:version
v1="0" v2="1"> en tenant plus particulièrement compte des besoins des nouveaux États
membres,</transltxt:version></p>
</item>
</transltxt:input>
</transltxt:masterl>
<transltxt:targetl>
<transltxt:input>
<item>
<p>la mise en œuvre effective de la législation environnementale en vigueur, en tenant plus
particulièrement compte des besoins des nouveaux États membres,</p>
</item>
</transltxt:input>
</transltxt:targetl>
</transltxt:update>
</insert>
</change>
<change>
<where>/bud-remarks[1]/p[2]</where>
<insert>
<transltxt:update>
<transltxt:masterl>
<transltxt:input>
<transltxt:version v1="0" v2="1">
<p>Ce crédit est également destiné à financer un service d'assistance Natura 2000.</p>
</transltxt:version>
</transltxt:input>
</transltxt:masterl>
<transltxt:targetl>
<transltxt:input>
<p>Ce crédit est également destiné à financer un service d'assistance Natura 2000.</p>
</transltxt:input>
</transltxt:targetl>
</transltxt:update>
</insert>
</change>
</changes>
```

Remplacement des unités de traduction dans la langue cible

Une fois la liste des unités de traduction créée, il ne reste qu'à effectuer la substitution dans la langue cible. C'est la feuille de style generateTT.xsl (GenerateTT.java) qui réalise ce traitement. Elle prend un paramètre "changesFile", qui est le nom du fichier produit lors de la première étape.

Le fonctionnement de generateTT.xsl est simple : tous les éléments`p`,`table`,`title`,`subtitle`,`row`,`list`et`item`du document sont analysés. On calcule leur "xpath" et s'il correspond à un de ceux se trouvant dans la liste des unités de traduction, on remplace l'élément par le`transltxt:update`de la liste.

Dans le cas de l'exemple précédent, la version courante de la langue cible est :

```
<bud-remarks>
<list type="dash">
<int.li>
<p>This appropriation is intended to cover expenditure on actions undertaken by the Commission
to implement existing legislation, awareness-raising and other general actions based on the
Community Environment action programme, the actions taken will target:</p>
</int.li>
<item>
<p>la mise en œuvre effective de la législation environnementale en vigueur, en tenant plus
particulièrement compte des besoins des nouveaux États membres,</p>
</item>
<item>
<p>the integration of environmental concerns into other Community policies,</p>
</item>
<item>
<p>working with the market through business and consumers towards more sustainable production
and consumption patterns,</p>
</item>
<item>
<p>ensuring reliable and accessible information on the environment for the European
citizen,</p>
</item>
<item>
<p>the development of a more environmentally conscious attitude towards land use.</p>
</item>
</list>
<p>A thematic strategy approach will be developed aimed to make an efficient and cost-
effective contribution to the achievement of environmental goals. The approach will apply
across the spectrum of environmental issues.</p>
<p>Ce crédit est également destiné à financer un service d'assistance Natura 2000.</p>
</bud-remarks>
```

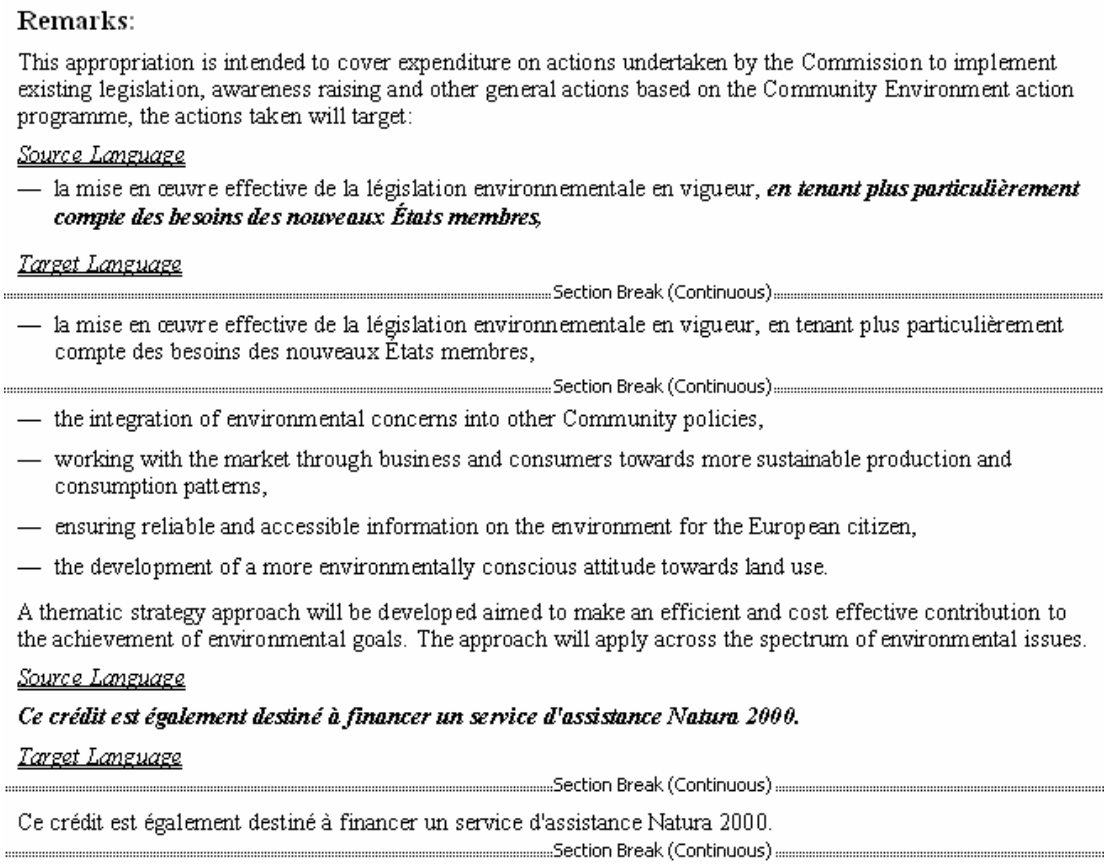
On voit que toutes les modifications ont bien été apportées lors de la mise à jour de l'amendement : le premier élément de liste est en français parce que le texte modifié par l'auteur est reporté dans toutes les langues. Le paragraphe qui se trouvait après la liste n'existe plus et le dernier paragraphe a été ajouté en français.

La liste passée en paramètre indique deux unités de traduction, se trouvant aux positions `/bud-remarks[1]/list[1]/item[1]`et`/bud-remarks[1]/p[2]`. L'item et le paragraphe en question sont substitués pour produire le XML*Translation Text*:

```
<bud remarks>
<list type="dash">
<int.li>
<p>This appropriation is intended to cover expenditure on actions undertaken by the Commission
to implement existing legislation, awareness raising and other general actions based on the
Community Environment action programme, the actions taken will target:</p>
</int.li>
<transltxt:update xmlns:transltxt="http://www.softwareag.com/transltxt">
<transltxt:masterl>
<transltxt:input>
<item>
```

```
<p>la mise en œuvre effective de la législation environnementale en vigueur,
<transltxt:version v1="0" v2="1">en tenant plus particulièrement compte des besoins des
nouveaux États membres,</transltxt:version></p>
</item>
</transltxt:input>
</transltxt:masterl>
<transltxt:targetl>
<transltxt:input>
<item>
<p>la mise en œuvre effective de la législation environnementale en vigueur, en tenant plus
particulièrement compte des besoins des nouveaux États membres,</p>
</item>
</transltxt:input>
</transltxt:targetl>
</transltxt:update>
<item>
<p>the integration of environmental concerns into other Community policies,</p>
</item>
<item>
<p>working with the market through business and consumers towards more sustainable production
and consumption patterns,</p>
</item>
<item>
<p>ensuring reliable and accessible information on the environment for the European
citizen,</p>
</item>
<item>
<p>the development of a more environmentally conscious attitude towards land use.</p>
</item>
</list>
<p>A thematic strategy approach will be developed aimed to make an efficient and cost
effective contribution to the achievement of environmental goals. The approach will apply
across the spectrum of environmental issues.</p>
<transltxt:update xmlns:transltxt="http://www.softwareag.com/transltxt">
<transltxt:masterl>
<transltxt:input>
<transltxt:version v1="0" v2="1">
<p>Ce crédit est également destiné à financer un service d'assistance Natura 2000.</p>
</transltxt:version>
</transltxt:input>
</transltxt:masterl>
<transltxt:targetl>
<transltxt:input>
<p>Ce crédit est également destiné à financer un service d'assistance Natura 2000.</p>
</transltxt:input>
</transltxt:targetl>
</transltxt:update>
</bud remarks>
```

Une fois converti en RTF, le traducteur recevra le document :



5.7.14. Transforming the FT, RT and TT XML files to RTF

Les documents*Full Text*,*Reduced Text*et*Translation Text*sont mis en forme par le filtre de présentation`AmdAbb2Rtf`. Ce filtre hérite de la classe`XSLTFilter`qui fait partie du framework Xef. Il est constitué de quatre filtres qui s'enchaînent :

addEmptyP.xsl (AddEmptyP.java)

preProcessVersioning.xsl (PreProcessVersionning.java)

convAll2Tbl.xsl (ConvAll2Tbl.java)

amdabb2rtf.xsl (AmdAbb2Rtf.java)

5.7.14.1.La feuille de style addEmptyP.xsl

Le but de la feuille de style addEmptyP.xsl est d'ajouter des paragraphes vides entre les tableaux. Il s'agit de solutionner un problème lié à la présentation Word du document : dans MsWord, le seul séparateur accepté entre deux tableaux est un paragraphe. Comme les paragraphes n'existent pas dans le balisage initial, on les ajoute ici.

Typiquement, le but est de transformer ceci :

```
<table frame="none"><tgroup>…<tgroup></table>
<table frame="none"><tgroup>…</tgroup></table>
```

en ceci

```
<table frame="none"><tgroup>…</tgroup></table>
<p/>
```

`<table frame="none"><tgroup>…</tgroup></table>`

Ce traitement est volontairement effectué au début de la transformation et non pas lors de la génération du RTF. En effet, au moment de la génération du RTF, plusieurs éléments peuvent avoir été ajoutés entre les tableaux, ce qui complexifie grandement l'ajout du paragraphe vide.

5.7.14.2. La feuille de style preProcessVersioning.xsl

Le but de la feuille de style preProcessVersioning.xsl est de ramener les différents positionnements possibles des éléments `version` dans un tableau à un cas commun, simple, sur lequel l'application de styles RTF pourra être facilement effectuée (l'élément `version` est commun aux DTDs du *Full Text*, du *Reduced Text* et *Translation Text*, les 3 espaces de nom sont donc pris en compte).

Le problème vient du fait qu'un élément `version` peut aussi bien englober du texte, qu'une ou plusieurs cellules (ou lignes) entières. Comme la conversion vers RTF d'un tableau est une opération délicate, seul le cas simple d'un élément `version` englobant le texte d'une cellule est permis. Il faut donc convertir tous les autres cas envisageables vers ce cas simple.

Voici quelques exemples de transformations effectuées par preProcessVersioning.xsl (l'élément `<version v1="1" v2="0">` est symbolisé par `<old>` et l'élément `<version v1="0" v2="1">` est symbolisé par `<new>`)

AVANT	APRES	Remarques
`<entry><old><p>P1</p></old></entry>`	`<entry><p><old>P1</old></p></entry>`	Suppression du contenu d'une cellule. Le principe est identique pour un élément new (ajout de texte dans une cellule vide).
`<entry><old><p>P1</p></old><new><p>P2</p></new></entry>`	`<entry><p><old>P1</old><new>P2</new></p></entry>`	Modification du contenu d'une cellule.
`<old><entry colname=A><p>P1</p></entry><entry colname=B><p>P2</p></entry></old><new><entry colname=A><p>P3</p></entry><entry colname=B><p>P4</p></entry></new>`	`<entry colname=A><p><old>P1</old></p><p><new>P3</new></p></entry><entry colname=B><p><old>P2</old></p><p><new>P4</new></p></entry>`	Modification de cellules. Si les modifications n'affectent pas les merges entre les cellules, on fusionne les entry de même colname ou de même couple namest/nameend.
`<old><entry colname=A><p>P1</p></entry><entry colname=B><p>P2</p></entry></old>`	`<entry colname=A><p><old>P1</old></p></entry><entry colname=B><p><old>P2</old></p></entry>`	Suppression de cellules. Le principe est identique pour un élément new (ajout de cellules).
`<old><row><entry><p>P1</p></entry><entry><p>P2</p></entry></row></old>`	`<row><entry><p><old>P1</old></p></entry><entry><p><old>P2</old></p></entry></row>`	Suppression d'une ligne. Le principe est identique pour un élément new (ajout d'une ligne).

PreProcessVersioning.xsl simplifie également un autre cas problématique: celui du *Translation Text* dans les tableaux. On a vu dans la présentation du *Translation Text* que l'unité de traduction dans un tableau pouvait être la ligne. Dans ce cas, le tableau est donc divisé en éléments `update`[10] alors que lors de la conversion vers RTF, on s'attend à traiter une division en `row`.

En conséquence, preProcessVersioning transforme chaque section du tableau en un tableau indépendant, en recopiant sur chacun les informations de taille et de nom de colonnes (élément colspec).

Par exemple, le tableau suivant contient quatre sections :

[10] Update est l'élément contenant les unités de traduction, cf. documentation du Translation Text

La première ligne du tableau (`row id="1"`)

La ligne "langue source" (`row id="2"`)

La ligne "langue cible" (`row id="3"`)

Les deux dernières lignes du tableau (`row id="4"`et`row id="5"`)

```
<table>
<tgroup>
<colspec.../>
<colspec.../>
<tbody>
<row id="1">...</row>
<transltxt:update>
<transltxt:masterl>
<transltxt:input>
<row id="2">…</row>
</transltxt:input>
</transltxt:masterl>
<transltxt:targetl>
<transltxt:input>
<row id="3">…</row>
</transltxt:input>
</transltxt:targetl>
</transltxt:update>
<row id="4">…</row>
<row id="5">…</row>
</tbody>
</tgroup>
</table>
```

Il est donc transformé en 4 tableaux :

```
<table>
<tgroup>
<colspec.../>
<colspec.../>
<tbody>
<row id="1">...</row>
</tbody>
</tgroup>
</table>
<transltxt:update>
<transltxt:masterl>
<transltxt:input>
<table>
<tgroup>
<colspec.../>
<colspec.../>
<tbody>
<row id="2">…</row>
</tbody>
</tgroup>
</table>
</transltxt:input>
</transltxt:masterl>
<transltxt:targetl>
<transltxt:input>
<table>
<tgroup>
<colspec.../>
<colspec.../>
<tbody>
<row id="3">…</row>
</tbody>
</tgroup>
```

```
</table>
</transltxt:input>
</transltxt:targetl>
</transltxt:update>
<table>
<tgroup>
<colspec.../>
<colspec.../>
<tbody>
<row id="4">…</row>
<row id="5">…</row>
</tbody>
</tgroup>
</table>
```

5.7.14.3.La feuille de style convAll2Tbl.xsl

La feuille de style convAll2Tbl.xsl remplit les tâches suivantes :

création du tableau des chiffres sous la forme d'un tableau CALS,

création de l'élément`lb-title`. Il contient le texte qui, sur chaque transaction, liste toutes les parties de la ligne budgétaire qui ont été modifiées (chiffres, intitulé, commentaires, actes de références, bases légales…),

ajout de textes spécifiques aux*Full Text*,*Reduced Text*et*Translation Text.*

ConvAll2Tbl.xsl a pour paramètres :

`CstTag`: le tag CST

`Role`: à qui est destiné le document :`auteur`(auteurs),`trad`(traducteurs),`cor`(collationnement),`cons`(consultation),`print`(cahier texte) (par défaut :`auteur`)

`Vl`: la version linguistique

`LbParts`: les parties de documents à ne pas afficher

`PFormat`:`FT`(Full Text),`RT`(Reduced Text),`TT`(Translation Text),`AFTER_VOTE_RT`(rapport de la plénière)

`AmdYear`: l'année budgétaire

`DisplayFig15`: pour afficher ou ne pas afficher les chiffres à 15 (par défaut 0, c'est-à-dire. on n'affiche pas les chiffres à 15)

`CstBase`: la langue de base de l'amendement (par défaut,`CstBase`prend la valeur du paramètre`Vl`)

Création du tableau des chiffres

Le tableau des chiffres est créé à partir de l'élément`figures`de l'instance XML qui contient toutes les données chiffrées de la ligne budgétaire et de l'amendement.

L'élément`figures`a pour fils :

`node_figure`:

`pd/n`: les chiffres de l'avant-projet de budget

`pd/n_1`: les chiffres du budget pour l'année N-1

`pd/n_2`: les chiffres de l'exécution du budget pour l'année N-2

`d/n`: les chiffres du projet de budget

`d/n_1`: les chiffres du budget pour l'année N-1

`d/n_2`: les chiffres de l'exécution du budget pour l'année N-2

`amd_figure`: les chiffres de l'amendement (nouveaux montants et différences)

`h_figure[cd_version = "PE L1"]`: les chiffres PEL1

`h_figure[cd_version = "CO L2"]`: les chiffres COL2

Il va de soit que tous ces chiffres ne sont pas systématiquement disponibles. Par exemple, les chiffres PEL1 n'existent qu'à partir de COL2.

Suivant l'institution (Conseil ou Parlement) et la lecture (L1 ou L2), on décide du tableau des chiffres qui sera construit.

En COL1, le tableau des chiffres se compose des données suivantes :

le budget de l'année N-1 (`node_figure/pd/n_1`)

l'avant-projet de budget de l'année courante (`node_figure/pd/n`)

l'amendement (`amd_figure/df_xxx`)

les nouveaux montants (`amd_figure/mt_xxx`)

Exemple de tableau des chiffres COL1 :

Volume 4 (section 3) — Commission

Item 06 02 08 01 European Railway Agency for Safety and Interoperability – Subsidy under Titles 1 and 2

Amend figures as follows:

06 02 08 01	2004		PDB 2005		CHANGE		PDB+CHANGE	
NCE	Commitments	Payments	Commitments	Payments	Commitments	Payments	Commitments	Payments
Appropriations	p.m.	p.m.	10 770 000	10 770 000	-4 605 000	-4 605 000	6 165 000	6 165 000
Reserves	4 490 000	4 490 000						

En PEL1 :

le budget de l'année N-1 (`node_figure/d/n_1`)

l'avant-projet de budget (`node_figure/pd/n`)

le projet de budget (`node_figure/d/n`)

l'amendement (`amd_figure/df_xxx`)

les nouveaux montants (`amd_figure/mt_xxx`)

Exemple de tableau des chiffres PEL1 :

Volume 4 (section 3) — Commission

Item 06 02 02 01 European Maritime Safety Agency — Subsidy under Titles 1 and 2

Amend figures as follows:

06 02 02 01	DB 2004		PDB 2005		DB 2005		AMENDMENT		DB+AMENDMENT	
	Commitments	Payments	Commitments	Payments	Commitments	Payments	Commitments	Payments	Commitments	Payments
Appropriations	9 800 000	9 800 000	14 000 000	14 000 000	14 000 000	14 000 000	1 000 000	1 000 000	15 000 000	15 000 000
Reserves										

En COL2 :

l'avant-projet de budget (`node_figure/pd/n`)

le projet de budget (`node_figure/d/n`)

les chiffres PEL1 (`h_figure[cd_version="PE L1"]`)

l'amendement (`amd_figure/df_xxx`)

les nouveaux montants (`amd_figure/mt_xxx`)

Exemple de tableau des chiffres COL2 :

SECTION IV - COUR DE JUSTICE

Chapitre 10 0 CRÉDITS PROVISIONNELS

Modifier les chiffres comme suit:

10 0	APE2004		PE2004		PE1		AMENDEMENT		CSL2	
DNO	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements
Crédits	p.m.	p.m.	p.m.	p.m.	p.m.	p.m.	1 000 000	1 000 000	1 000 000	1 000 000
Réserves										

En PEL2 :

le projet du budget (`node_figure/d/n`)

les chiffres PEL1 (`h_figure[cd_version="PE L1"]`)

les chiffres COL2 (`h_figure[cd_version="CO L2"]`)

l'amendement (`amd_figure/df_xxx`)

les nouveaux montants (`amd_figure/mt_xxx`)

Exemple de tableau des chiffres PEL2 :

SECTION IV - COUR DE JUSTICE

Poste 3 7 1 0 Frais judiciaires

Modifier les chiffres comme suit:

3 7 1 0	PB2004		PE1		CSL2		AMENDEMENT		PE2	
	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements	Engagements	Paiements
Crédits	30 000	30 000	30 000	30 000	30 000	30 000	5 000	5 000	35 000	35 000
Réserves										

Création de l'élément lb_title

L'élément`lb-title`est ajouté à chaque transaction. Il apporte des précisions sur la nature de la transaction : est-ce une suppression, un ajout, une modification, etc. … ?

S'il s'agit d'une modification, on énumère toutes les parties de document modifiées par l'amendement. Il peut s'agir des chiffres, des objectifs généraux, de l'intitulé, des commentaires, des bases légales et/ou des actes de référence. Le cas des échéanciers et du tableau des effectifs a également été prévu, même s'il n'est pas applicable pour le moment.

Le texte de`lb_title`est dépendant de la version linguistique. Il est construit à partir de données de l'amendement (alias) et de portions de textes extraites des fichiers de titres.

Pour construire le contenu de`lb_title`, on se base tout d'abord sur l'élément de la "transaction" :`modify`,`delete`,`add`,`merge`,`mergeupd`,`split`,`splitupd`ou`splitadd`. S'il s'agit de`modify`, on s'intéresse alors au statut de toutes les parties de document. Un statut se trouve dans un élément de nom`partname-status`, avec un attribut`change`de valeur "change" si la partie a été modifiée ou " unchanged" si la partie n'a pas été modifiée.

Par exemple, pour les chiffres, le statut est renseigné dans un élément`figures-status`.Une valeur "change" indique que les chiffres ont été amendés. De même il existe les éléments`bud-heading-status`(statut de l'intitulé),`bud-intro-status`(les objectifs généraux),`bud-remarks-status`(les commentaires),`bud-legal-status`(les bases légales) et`bud-reference-status`(les actes de référence).

Exemple d'une suppression de ligne :

```
<delete id="">
<lb_alias>16 01 02 02</lb_alias>
<node.ti.bud>…</node.ti.bud>
<amd-data exprev="exp">...</amd-data>
</ delete>
```

Ajout du lb_title en français :

```
<delete id="">
<lb_alias>16 01 02 02</lb_alias>
<node.ti.bud>…</node.ti.bud>
<lb_title>Supprimer: 16 01 02 02</lb_title>
<amd-data exprev="exp">...</amd-data>
</ delete>
```

Exemple d'une modification des chiffres et des commentaires :

```
<modify id="TRA5326">
<lb_alias>08 02 01 02</lb_alias>
<node.ti.bud>…</node.ti.bud>
<amd-data exprev="exp">
<figures-status change="changed">...</figures-status>
<figures>...</figures>
</amd-data>
```

```
<bud-heading-status change="unchanged"/>
<bud-heading>...</bud-heading>
<bud-remarks-status change="changed"/>
<bud-remarks>...</bud-remarks>
<bud-legal-status change="unchanged"/>
<bud-legal></bud-legal>
<bud-reference-status change="unchanged"/
<bud-reference></bud-reference>
</modify>
```

Ajout du lb_title en français :

```
<modify id="TRA5326">
<lb_alias>08 02 01 02</lb_alias>
<node.ti.bud>…</node.ti.bud>
<lb_title>Modifier les commentaires et les chiffres comme suit:</lb_title>
<amd-data exprev="exp">
<figures-status change="changed">...</figures-status>
<figures>...</figures>
</amd-data>
<bud-heading-status change="unchanged"/>
<bud-heading>...</bud-heading>
<bud-remarks-status change="changed"/>
<bud-remarks>...</bud-remarks>
<bud-legal-status change="unchanged"/>
<bud-legal></bud-legal>
<bud-reference-status change="unchanged"/
<bud-reference></bud-reference>
</modify>
```

Traitements propres aux Full Text, Reduced Text et Translation Text

Ces traitements sont localisés dans des feuilles de style séparées, incluses dans la feuille de style principale convAll2Tbl.xsl :

convAll2Tbl-fulltxt.xsl transforme les éléments`fulltxt:update-attsen``fulltxt:update.`

convAll2Tbl-redtxt.xsl ajoute du texte spécifique au*Reduced Text*. Il s'agit des libellés sur la localisation de la modification (avant/après l'alinéa/le tableau/la liste commençant par … et se terminant par…) et sur son type (supprimer le texte suivant, ajouter le texte suivant, modifier le texte comme suit). C'est également dans cette feuille de style que sont "réduits" les paragraphes servant de point d'ancrage (seuls les 10 premiers et 10 derniers mots en sont conservés).

convAll2Tbl-transltxt.xsl ajoute le texte "Langue source" (dans la version linguistique appropriée) à chaque élément`transltxt:masterl`et le texte "Langue cible" à chaque élément`transltxt:targetl`.

5.7.14.4.La feuille de style amdabb2rtf.xsl

La feuille de style amdabb2rtf.xsl est la dernière étape dans le processus de production du document Word, c'est donc elle qui génère le code RTF.

Elle inclut plusieurs feuilles de style, chacune dédiée à la conversion vers RTF d'une type d'élément particulier :

amdabb2rtf-utils.xsl : contient une série de templates "utilitaires" (pour le formatage d'une date, le padding d'une chaîne de caractères, pour créer un bookmark, démarrer ou fermer une section protégée…)

amdabb2rtf-header.xsl : contient le "header" et le "trailer" RTF

amdabb2rtf-styles.xsl : contient une série de styles RTF (Normal, Table title, Table subtitle…)

amdabb2rtf-list.xsl : effectue la conversion vers RTF des listes

amdabb2rtf-tables.xsl : effectue la conversion vers RTF des tableaux

amdabb2rtf-redtxt.xsl : gère les éléments propres à la DTD du*Reduced Text*(formatage du contenu des élémentsversionen gras-italique, en "strike" ou en marques de révision)

amdabb2rtf-fulltxt.xsl : gère les éléments propres à la DTD du*Full Text*(formatage du contenu des élémentsversionen gras-italique, en "strike" ou en marques de révision)

amdabb2rtf-transltxt.xsl : gère les éléments propres à la DTD du*Translation Text*. La langue source (masterl) est mise dans une section protégée, le texte ajouté y est formaté en gras-italique et le texte supprimé en "strike". La langue cible (targetl) est déprotégée.

Elle inclut également trois feuilles de style en charge de la génération de la page de garde des cahiers texte de la séance plénière :

amdabb2rtf-coverPage.xsl : contient le "squelette" d'une page de garde, c'est-à-dire la structure commune à toutes les versions linguistiques

amdabb2rtf-coverPage_VLDepdtTxt.xsl : contient les parties dépendantes de la version linguistique

amdabb2rtf-coverPage_pictures.xsl : contient les images.

5.7.15. Producing the "Cahier Chiffres" report

Le Cahier Chiffres est un rapport basé sur une liste d'amendements sélectionnés par l'utilisateur. Les amendements y sont organisés en catégorie/politique (c'est la ligne budgétaire principale de l'amendement qui détermine à quelle catégorie/politique il appartient).

Pour chaque amendement on présente la liste de lignes amendées et les chiffres de la transaction.

5.7.15.1.Génération de l'XML

L'instance XML, comme le rapport, est organisée en catégories (élémentCATEGORY) et politiques (élémentPOLITICS). Leurs libellés sont contenus dans un élémentLANG_LABELSavec un ou plusieursLANG_LABEL(un par libellé et version linguistique) Les politiques contiennent des amendements. Aucune structure particulière n'a été créée pour représenter un amendement destiné au cahier chiffres, c'est donc la structure habituelle qui est utilisée ici (i.e. celle qui sert également à la génération des cahiers textes). Plus précisément, on effectue une extraction au format FT (*Full Text*) car elle n'est pas basée sur un différentiel et elle est donc plus rapide que celle du RT (*Reduced Text*).

L'élément racine de l'instance XML produite estREPORT. L'instance XML est donc organisée de la sorte :

```
<report>
<lb_figures>FIGURES_BEFORE_COUNCIL</lb_figures>
<category id_category="1">
<lang_labels>
<lang_label id_language="FR">...</lang_label>
</lang_labels>
<politics id_politics="1.1">
<lang_labels>
<lang_label id_language="FR">...</lang_label>
```

```
</lang_labels>
<amd>
      ...
<transactions>
<modify id="">...</modify>
</transactions>
      ...
</amd>
<amd>
       ...
<transactions>
<delete>...lete>
<add>...</add>
</transactions>
      ...
</amd>
</politics>
<politics id_politics="1.2">
<lang_labels>
<lang_label id_language="FR">...</lang_label>
</lang_labels>
<amd>...</amd>
<amd>...</amd>
</politics>
</category>
</report>
```

La génération du Cahier Chiffres s'effectue au moyen de la classe `CahierChiffres`(package com.softwareag.belgium.seibud.rmg.reports). Elle hérite de la classe`Cahier`, qui contient notamment une méthode pour classer une liste d'amendements par catégories et politiques (`mergeAllAmendmentsByCatPol`, utilisée également pour la liste de votes).

5.7.15.2.Génération du rapport RTF

Lors de la demande de génération d'un Cahier Chiffres, l'utilisateur a la possibilité d'ajouter des informations supplémentaires dans le rapport :

Le résultat du vote ("Vote results")

Les notes du Parlement ("EP notes")

Les notes du Conseil et du Parlement ("Council and EP notes")

Il existe donc deux types de feuilles de style : celles générant un Cahier Chiffres "simple", sans d'autres informations que les chiffres, et celles générant un Cahier Chiffres "extra", contenant en plus des chiffres le résultat du vote ou les notes.

Les chiffres présentés varient selon l'institution et la lecture courantes (COL1, PEL1, COL2 ou PEL2). Au Conseil 1ière lecture, le Cahier Chiffres contient :

Le budget de l'année N-1 (`figures/node_figure/pd/n_1`)

L'avant-projet de budget (`figures/node_figure/pd/n`)

Les différences (nouveaux montants – avant-projet de budget)

Les nouveaux montants (`figures/amd_figure`)

Header d'un Cahier chiffres COL1 "simple" en français :

No.Am.	Intitulé	2004	APB 2005	AMENDEMENT	APB+AMENDEMENT

Dans le cas d'un Cahier Chiffres "extra", une colonne est ajoutée après les nouveaux montants. Elle porte l'intitulé "Vote" s'il faut ajouter les résultats du vote ou "Notes" s'il faut ajouter des notes.

Au Parlement 1ière lecture :

Le budget de l'année N-1 (`figures/node_figure/pd/n_1`)

L'avant-projet de budget (`figures/node_figure/pd/n`)

Le projet de budget (`figures/node_figure/d/n`)

Les différences (nouveaux montants – projet de budget)

Les nouveaux montants (`figures/amd_figure`)

Au Conseil 2ième lecture :

L'avant-projet de budget (`figures/node_figure/pd/n`)

Le projet de budget (`figures/node_figure/d/n`)

Le chiffres PEL1 (`figures/h_figure[cd_version="PE L1"]`)

Les différences (nouveaux montants – PEL1)

Les nouveaux montants (`figures/amd_figure`)

Au Parlement 2ième lecture :

Le projet de budget (`figures/node_figure/d/n`)

Le chiffres PEL1 (`figures/h_figure[cd_version="PE L1"]`)

Le chiffres COL2 (`figures/h_figure[cd_version="CO L2"]`)

Les différences (nouveaux montants – COL2)

Les nouveaux montants (`figures/amd_figure`)

Afin de gérer tous les cas possibles (COL1, PEL1, COL2 ou PEL2 et Cahier Chiffres "simple" ou "extra"), on dispose de huit filtres :

Reportcdc2rtf_col1 : Cahier Chiffres "simple" au Conseil 1ière lecture (feuille de style reportcdc2rtf-co-l1.xsl)

Reportcdc2rtf_col1_extra : Cahier Chiffres "extra" au Conseil 1ière lecture (reportcdc2rtf-co-l1-extra.xsl)

Reportcdc2rtf_pel1 : Cahier Chiffres "simple" au Parlement 1ière lecture (reportcdc2rtf-pe-l1.xsl)

Reportcdc2rtf_pel1_extra : Cahier Chiffres "extra" au Parlement 1ière lecture (reportcdc2rtf-pe-l1-extra.xsl)

Reportcdc2rtf_col2 : Cahier Chiffres "simple" au Conseil 2ième lecture (reportcdc2rtf-co-l2.xsl)

Reportcdc2rtf_col2_extra : Cahier Chiffres "extra" au Conseil 2ième lecture (reportcdc2rtf-co-l2-extra.xsl)

Reportcdc2rtf_pel2 : Cahier Chiffres "simple" au Parlement 2ière lecture (reportcdc2rtf-pe-l2.xsl)

Reportcdc2rtf_pel2_extra : Cahier Chiffres "extra" au Parlement 2ième lecture (reportcdc2rtf-pe-l2-extra.xsl)

Ces feuilles de style prennent toutes les mêmes paramètres :

`TitleLanguage`: la version linguistique choisie par l'utilisateur

`ReportLanguage`: la version linguistique choisie par l'utilisateur ou la langue par défaut si l'utilisateur a choisi "OR"

`TitlesDir`: le chemin vers le répertoire qui contient les fichiers de titres

`Year`: l'année budgétaire en cours

`Extra`: quelles informations supplémentaires doivent être affichées dans le rapport ?

Valeur "`ADDVOTE`" : le résultat du vote

Valeur "`ADDEPNOTES`" : les notes du Parlement

Valeur "`ADDCLEPNOTES`" : les notes du Conseil et du Parlement

Valeur "`NONE`" : aucune

`TpPhase`: le budget de référence. Cette valeur permet d'indiquer, pendant la 1ière lecture du Parlement, si l'on travaille par rapport à l'avant-projet de budget (avant le 15 août) ou par rapport au projet de budget (après le 15 août)

Valeur "`PD`" pour l'avant-projet de budget

Valeur "`D`" pour le projet de budget

`ShowComp`: "`YES`" si l'information Dépense obligatoire/Dépense non obligatoire doit être affichée, "NO" sinon.

`ShowPolicy`: "`YES`" si l'identifiant de la politique doit être affichée sur chaque transaction, "NO" sinon.

`SortByCatPolArea`:

Valeur "1" si l'utilisateur a choisi, dans ses critères de recherche, de trier les amendements par "Category and policy area". Dans ce cas les libellés des politiques ne sont pas affichés dans le rapport.

Valeur "0" sinon (tri par CAT/POL). Les libellés des politiques sont affichés dans le rapport.

Les huit feuilles de style regroupent les traitements propres à leur phase. Elles contiennent chacune leur implémentation des fonctions suivantes :

`createCDCHeader`: sortie du header du tableau du Cahier Chiffres

`outputTransactionRow_FirstAndLast`: sortie de la ligne d'un amendement ne contenant qu'une transaction

`outputTransactionRow_First`: sortie de la première ligne d'un amendement contenant plusieurs transactions

`outputTransactionRow_Last`: sortie de la dernière ligne d'un amendement contenant plusieurs transactions

`outputTransactionRow`: sortie d'une ligne qui n'est ni la première ni la dernière d'un amendement contenant plusieurs transactions

`outputTotalRow`: sortie de la ligne du total d'un amendement contenant plusieurs transactions

Ainsi, dans l'exemple suivant (COL1 "simple"), la première ligne a été générée par`outputTransactionRow_First`, la deuxième par`outputTransactionRow`, la troisième par`outputTransactionRow_Last`et la dernière par`outputTotalRow`:

0012	A6 01 01	Expenditure related to staff in active employment	37 674 000	37 674 000	31 797 000	31 797 000	**-400 000**	**-400 000**	31 397 000	31 397 000
	A6 01 02 01	External staff	10 587 000	10 587 000	12 884 000	12 884 000	**-420 000**	**-420 000**	12 464 000	12 464 000
0012	A6 01 03	Buildings and related expenditure	11 056 000	11 056 000	11 390 000	11 390 000	**-151 166**	**-151 166**	11 238 834	11 238 834
						Total	**-971 166**	**-971 166**		

Exemple de ligne produite par`outputTransactionRow_FirstAndLast`:

0213	2 3 4	Damages and interest	p.m.	p.m.	p.m.	p.m.			p.m.	p.m.

Les huit feuilles de style incluent deux autres feuilles de style qui effectuent des traitements communs à toutes les phases :

`reportcdc2rtf.xsl`: c'est dans cette feuille de style qu'est initiée la conversion.

`reportcdc2rtf-utils.xsl`: contient une série de templates "utilitaires" (sortie du header et du trailer RTF, sortie du titre d'une catégorie ou d'une politique,…).

5.7.16. Producing the Vote List report

La liste de votes est un rapport basé sur une liste d'amendements sélectionnés par l'utilisateur. Les amendements y sont organisés en catégorie/politique puis par ligne budgétaire principale (c'est la ligne budgétaire principale qui détermine à quelle catégorie/politique l'amendement appartient).

Pour chaque ligne, on donne son intitulé en allemand, anglais et français (car une liste de vote n'est pas liée à une langue particulière) puis la liste des amendements dont elle est la ligne budgétaire principale.

Pour chaque amendement on indique le total des engagements (ligne + réserve) et des paiements (ligne + réserve), toutes transactions confondues, ainsi que les types de vote, opinion et décision qui ont été pris.

La liste de votes est un rapport basé sur une liste d'amendements sélectionnés par l'utilisateur. Les amendements y sont organisés en catégorie/politique puis par ligne budgétaire principale (c'est la ligne budgétaire principale qui détermine à quelle catégorie/politique l'amendement appartient).

Pour chaque ligne, on donne son intitulé en allemand, anglais et français (car une liste de vote n'est pas liée à une langue particulière) puis la liste des amendements dont elle est la ligne budgétaire principale.

Pour chaque amendement on indique le total des engagements (ligne + réserve) et des paiements (ligne + réserve), toutes transactions confondues, ainsi que les types de vote, opinion et décision qui ont été pris.

5.7.16.1. Génération de l'XML

L'XML servant à générer la liste de votes est le même que celui du cahier chiffres. Se référer à cette partie.

5.7.16.2. Génération du rapport RTF

Le filtre de génération de la liste de votes est contenu dans la classe`Reportvote2rtf`. La transformation s'effectue au moyen d'une seule feuille de style : reportvote2rtf.xsl.

Elle prend les paramètres suivants :

`Year`: l'année budgétaire en cours

`Extra`: indique si l'utilisateur a demandé d'afficher le résultat des votes (valeur "ADDVOTE" s'il l'a demandé, vide sinon)

`TpPhase`: le budget de référence. Cette valeur permet d'indiquer, pendant la 1ière lecture du Parlement, si l'on travaille par rapport à l'avant-projet de budget (avant le 15 août) ou par rapport au projet de budget (après le 15 août).

Reportvote2rtf.xsl contient la logique de la transformation et elle inclut une autre feuille de style, reportvote2rtf-utils.xsl, qui détient tout le code RTF.

Les principales templates de reportvote2rtf-utils.xsl sont :

`createVoteHeader`: sort le header du document

`createVoteFooter`: sort le footer du document (contient la date d'ouverture du fichier et la pagination)

`outputCategoryTitle`: sort les libellés allemand, anglais et français d'une catégorie séparés par des "slash"

`outputPoliticsTitle`: sort les libellés allemand, anglais et français d'une politique séparés par des "slash"

`outputFirstRow`: sort la première ligne d'un tableau, avec l'alias de la ligne budgétaire principale et son intitulé allemand.

`outputSecondRow`: sort la deuxième ligne du tableau. La première colonne est vide, la deuxième contient l'intitulé anglais de la ligne budgétaire principale.

`outputOtherRow`: sort la troisième ligne du tableau. La première colonne est vide, la deuxième contient l'intitulé français de la ligne budgétaire principale.

`outputSummaryRow`: sort la dernière ligne du tableau. Elle contient le numéro de l'amendement, le type d'opinion (+,-,), le type de vote, le totaux en engagement et en paiement toutes transactions confondues, et le type de la décision (+,-,).

Le type d'opinion : la décision de la Commission Budgétaire

"+" si adopté,

"-" si rejeté,

rien si retiré

Le type de vote: la manière dont la décision a été prise et le nom du bloc (par exemple Bloc 1) en cas de vote en bloc lors de la séance plénière

"Roll call" si`tp_vote`vaut "`Nominal`"

"Separate vote" si`tp_vote`vaut "`Separated`"

"Split vote" si`tp_vote`vaut "`Division`"

"Roll call and split vote" si`tp_vote`vaut "`NomDiv`"

La valeur de`tp_vote`sinon

Le type de la décision : le résultat du vote de la séance plénière (affiché uniquement si l'utilisateur a coché "Add Vote results") :

"+" si adopté,

"-" si rejeté,

rien si retire.

Par exemple, dans cet extrait de liste de votes, le header a été généré par`createVoteHeader`, le texte "Verkehr/Transport/Transports" par`outputPoliticsTitle`, la première ligne du tableau par`outputFirstRow`, la deuxième par`outputSecondRow`, la troisième par`outputOtherRow`et chacune des deux suivantes par`outputSummaryRow`.

Line	PLENARY	Vote

Verkehr/Transport/Transports

06 02 01 01	Europäische Agentur für Flugsicherheit — Haushaltszuschüsse im Rahmen der Titel 1 und 2				
	European Aviation Safety Agency — Subsidy under Titles 1 and 2				
	Agence européenne pour la sécurité aérienne — Subvention aux titres 1 et 2				
	Amd. 640	+	Block 3	2 440 000 1 859 000	+
	Amd. 553	+	Separate vote		-

5.7.17. Producing the Vote Results report

This report is a new type of vote list that gives a brief overview of the vote results achieved during the plenary sessions of the European Parliament dedicated to the Budget.

It contains only standard mentions that can therefore be requested in any language of the European Union (except from the part after the voting list, containing the details of the split votes and the specific requests from groups or MEPs concerning the voting procedure).

No manual translation is needed because all the texts present in it are standard texts.

The columns of the vote list are:

Amendment number

Main budget line

Vote type: Bloc n roll-call vote (RCV) electronic vote (EV) split vote (split) separate vote (sep)

Vote results by parts: +: adopted -: rejected ↓: lapsed W: withdrawn

Vote results in case of RCV or EV vote: for, against, abstentions (max 15 characters)

When there is a split vote, there is more than one vote result. Each result takes up another line. The results are differentiated by sequence numbers (1..n).

Any budget line number that is the same in several amendments is visually grouped into one cell. It is the same for the vote results, but in the case of vote results, the result is repeated 3 times, for the first row, the middle row and the last row if there are more than 15 rows with the same result. Vote type values are grouped in the same way as vote results when they all refer to the same budget line.

The order of the amendments is the one defined by the vote list used during the plenary session of the European Parliament.

No header or footer is needed for this type of report because it is integrated in a larger document.

The table header is repeated on every page.

Bloc votes have a grey background in the 3 first columns of the table.

5.7.17.1.XML generation

The XML used to generate this report is the same as the one of the “Cahier Chiffres” report. Please refer to the corresponding section of this document for the specification of the format.

5.7.17.2.RTF report generation

The following picture shows a sample report as produced by the SEI-BUD system.

1. Draft general budget of the European Union – Financial year 2005

Am No	Budget line	Block, RCV, EV, sep, split		Vote	RCV/EV – remarks
COMMISSION					
1	21 99	Block 1		W	replaced by 1,2,3
7				↓	50,40,60
25	05 03 02 05	Block 99		+	50,40,60
4	06 02 02 03	Block 1		+	50,40,60
3	06 04 04	split	1	+	
			2	-	
			3	W	
			4	↓	
			5	W	
			6	-	100,50,20
		split/EV	7	+	
		split	8	-	
			9	W	
			10	↓	
5	18 03 08	Block 1		+	
2	18 07 01 02	Block 1			
6					1,2,999

The Java class `VoteResults2rtf.java` calls the vote results report generation filter. The transformation is done by the XSLT stylesheet `voteresults2rtf.xsl`.

The stylesheet accepts the following parameters:

`TitlesDir`: the directory where the fixed texts can be found; a translation is available for every language of the European Union

`Language`: the language that should be generated

`Year`: the Budget year (used in the title of the report)

`Extra`: a special flag for signalling if the enlargement of the European Union should be handled separately; currently unused.

`voteresults2rtf.xsl` handles the whole transformation process. It calls a subordinated stylesheet called `voteresults2rtf-utils.xsl` containing the actual RTF mark-up.

The most important templates of `voteresults2rtf-utils.xsl` are :

`createVoteHeader`: outputs the title of the report ;

`createTableHeader`: outputs the table header that is repeated on every page of the report;

`outputNewRow`: is the main template that outputs a row containing a single vote results; id defines the colour of the background and the borders of the row according to the following rules:

if the current amendment is the last one, also draw a border on the bottom of the row;

only draw a border on the top of the cell containing the amendment number (or alias) if the current amendment number (or alias) is different from the previous one;

if the vote type is `Block`, the background colour of the cells containing the amendment number, the alias or the vote type is set to grey with the RTF mark-up `\clcbpatraw18`;

only draw a border on the top of the cell containing the vote type if the current vote type is not the same as the previous one or if the current alias is different from the previous one or if the vote type is `Block` and the block number changes;

only draw a border on the top of the cell containing the vote result if the current decision is different from the previous one, if the current block number is different from the previous one or if the vote type is not `Block`;

do not repeat the amendment number (or alias) if they are the same as the previous one;

only output the vote type if it is not the same as the previous one or if the alias is not the same as the previous one or if the vote type is `Block` and the block number changes;

if the vote type is `split`, also insert the sequence number of the split amendment in the vote type column;

only output the vote result if the vote type is not `Block` or if the current block number is different from the previous one or if the current vote result is different from the previous one or if we have more than 14 decisions that are the same and if we are on the middle or last one.

5.7.18. Producing the Line by Line report

Le "Line by line" est un rapport présentant les chiffres de toutes les lignes budgétaires contenues dans un volume (ou dans une partie d'un volume).

Il existe trois présentations d'un rapport Line by line :

le Line by line "classique",

le Line by line "correction",

le Line by line "corrected".

Les rapports Line by line "correction" et "corrected" existent depuis l'introduction du chapitre XX 01 à l'ABB. Des problèmes ont été rencontrés lors du rapatriement des amendements vers le chapitre XX 01, si

bien qu'à l'issue de l'intégration, des divergences existent entre les chiffres obtenus par le système "SEI-BUD+Amendment" et ceux obtenus par "Badge-Bud". Pour remédier à ce problème, les montants "Badge-Bud" sont chargés dans le système (sans écraser ceux de SEI-BUD+Amendment) et les rapports "correction" et "corrected" permettent de les visualiser :

Le Line by line "correction" ne contient que les lignes pour lesquelles les montants "SEI-BUD+Amendment" et "Badge-Bud" diffèrent.

Le Line by line "corrected" contient la totalité des lignes, mais sur certaines les nouveaux montants indiqués sont ceux de Badge-Bud et non ceux calculés par "SEI-BUD+Amendment" (ils sont soulignés pour préciser que ce sont des montants corrigés).

5.7.18.1. Génération de l'XML

La structure de l'instance XML rappelle celle du rapport Line by line : une suite de lignes, ayant toutes un intitulé et éventuellement des chiffres.

Les lignes sont représentées par des éléments `NODE_INTEGRATION`. Chacune a un élément `TI.BUDS` contenant un ou plusieurs `TI.BUD` (l'intitulé de la ligne, un par version linguistique).

Les chiffres sont centralisés dans un élément `NODE_FIGURES`. On y trouve les éléments :

`PD_N_1`: contient les chiffres de l'avant-projet de budget de l'année N-1

`PD_N`: les chiffres de l'avant-projet de budget

`D_N_1`: les chiffres du projet de budget de l'année N-1

`D_N`: les chiffres du projet de budget

`PEL1`: les chiffres du Parlement, 1ère lecture

`COL2`: les chiffres du Conseil, 2ième lecture

`AMD`: les nouveaux montants calculés par SEI-BUD+Amendment

`CORRECTED`: les montants de Badge-Bud

Voici par exemple la représentation de la ligne 07 03 09 pendant la 1ère lecture du Conseil :

```
<NODE_INTEGRATION ID_NODE_INTEGRATION="76754">
<CD_CATEGORY>3</CD_CATEGORY>
<CD_POLITICS>3.0.13</CD_POLITICS>
<TP_COMPNCOMP>NCOMP</TP_COMPNCOMP>
<LB_ALIAS>07 03 09</LB_ALIAS>
<LB_ELEMENT>nmc-article</LB_ELEMENT>
<TP_EXPREV>exp</TP_EXPREV>
<TP_TOTAL />
<TP_HORIZONTAL />
<TI.BUDS>
<TI.BUD ID_LANGUAGE="EN">
<p>Community cooperation in the field of marine pollution</p>
</TI.BUD>
</TI.BUDS>
<NODE_FIGURES>
<AMD>...</AMD>
<PD_N>...</PD_N>
```

```
<PD_N_1>...</PD_N_1>
</NODE_FIGURES>
</NODE_INTEGRATION>
```

Bien évidemment la structure de l'instance XML n'est pas "plate", les lignes sont imbriquées :

```
<NODE_INTEGRATION ID_NODE_INTEGRATION="520687">
<LB_ALIAS>01</LB_ALIAS>
<NODE_INTEGRATION ID_NODE_INTEGRATION="520688">
<LB_ALIAS>01 01</LB_ALIAS>
<NODE_INTEGRATION ID_NODE_INTEGRATION="520689">
<LB_ALIAS>01 01 01</LB_ALIAS>
<NODE_FIGURES>...</NODE_FIGURES>
</NODE_INTEGRATION>
<NODE_INTEGRATION ID_NODE_INTEGRATION="520690">
<LB_ALIAS>01 01 02</LB_ALIAS>
<NODE_INTEGRATION ID_NODE_INTEGRATION="520691">
<LB_ALIAS>01 01 02 01</LB_ALIAS>
<NODE_FIGURES>...</NODE_FIGURES>
</NODE_INTEGRATION>
<NODE_INTEGRATION ID_NODE_INTEGRATION="520692">
<LB_ALIAS>01 01 02 11</LB_ALIAS>
<NODE_FIGURES>...</NODE_FIGURES>
</NODE_INTEGRATION>
</NODE_INTEGRATION>
    ....
```

Cette imbrication va permettre, lors de la génération du RTF, d'ajouter les lignes des totaux : à chaque fois qu'une branche de l'arbre XML se referme, on calcule son total et on insère une ligne supplémentaire "Total" dans le rapport.

L'élément racine de l'instance XML produite est`REPORT`. Il contient des informations de gestion sur la demande de rapport Line by line :

```
<REPORT ID_REPORTREQUEST="1538">
<DTT_BEGIN>12-DEC-03</DTT_BEGIN>
<DTT_END />
<DTT_DEADLINE />
<TP_REPORT>BUDGETLBL</TP_REPORT>
<STATE>S_INITIALIZED</STATE>
<LB_LANGUAGES>EN</LB_LANGUAGES>
<LB_RESULTS />
<LB_PARTS />
<LB_FIGURES>FIGURES_AFTER_COUNCIL</LB_FIGURES>
<LB_REPORT>VOL4 R+D</LB_REPORT>
<LB_INFO />
<ID_USER>59</ID_USER>
```

L'extraction est effectuée par la méthode statique`process`de la classe`LBLBuilder`(package com.softwareag.belgium.seibud.amdbuilder.reports.lbl.processing).

Trois "business objects" sont utilisés au cours de l'extraction (les classes se trouvent dans le package com.softwareag.belgium.seibud.amdbuilder.reports.lbl.bo):

`Report`: représentation d'une demande de rapport Line by line. Correspond à la table REPORTREQUESTS de la base de données. Dans l'instance XML du Line by line : élément`REPORT`.

`NodesIntegration`: représentation d'une ligne budgétaire. Correspond à la table NODES_INTEGRATION de la base de données. Dans l'instance XML du Line by line : élément`NODES_INTEGRATION`.

`FiguresIntegration`: représentation des chiffres d'une ligne budgétaire. Correspond à la table VERSIONS_FIGURES_INTEGRATION de la base de données. Dans l'instance XML du Line by line : élément`NODE_FIGURES`.

5.7.18.2.Génération du rapport RTF

Les demandes de rapport Line by line sont gérées par la méthode`generate`de la classe`LBLReport`, (package com.softwareag.belgium.seibud.rmg.reports).

Une fois l'XML produit par l'appel de la méthode`LBLBuilder.process`, le filtre XSLT adéquat est appliqué. Si l'utilisateur a choisi, sur l'écran de demande de Line by line, l'option "Show corrected amounts only", c'est un rapport "correction" qui est lancé. S'il a choisi l'option "Use corrected amounts", on lance un rapport "corrected". Sinon, c'est le Line by line classique qui est généré.

Le rapport ligne par ligne "classique"

Le filtre de génération d'un Line by line "classique" est`ReportLbl2RtfFilter`. Il hérite de la classe`XSLTFilter`faisant partie du framework Xef.

Le filtre est constitué d'une seule feuille de style :`reportlbl2rtf.xsl`. Elle prend pour paramètres les informations suivantes :

`Language`: la version linguistique du rapport

`Lecnum`: la lecture :

`FIGURES_BEFORE_PARLIAMENT`: Conseil 1ière lecture

`FIGURES_BEFORE_COUNCIL`: Parlement 1ière lecture

`FIGURES_AT_COUNCIL`: Conseil 2ième lecture

`FIGURES_AFTER_COUNCIL`: Parlement 2ième lecture

`TitlesDir`: le chemin vers le répertoire qui contient les fichiers de titres

`Year`: l'année budgétaire en cours

`ShowComp`: "`YES`" si l'information Dépense obligatoire/Dépense non obligatoire doit être affiché sur chaque ligne. "`NO`" sinon.

`ShowPolicy`: "`YES`" si la politique doit être affichée sur chaque ligne. "`NO`" sinon.

Suivant la valeur de`Lecnum`, les montants présentées ne seront pas les mêmes.

Au Conseil 1ière lecture, le Line by line contient :

Le budget de l'année N-1 (`node_figures/pd_n_1`)

L'avant-projet de budget de l'année courante (`node_figures/pd_n`)

Les nouveaux montants (`node_figures/amd`)

L'amendement (nouveaux montants – avant-projet de budget)

Le pourcentage "DB – Année N-1"

Header d'un Line by line COL1 en français :

Intitulé	2004	AFB 2005	PB 2005	PB-AFB	% PB-2004

Au Parlement 1ière lecture :

Le budget de l'année N-1 (`node_figures/d_n_1`)

L'avant-projet de budget (`node_figures/pd_n`)

Le projet de budget (`node_figures/d_n`)

Les nouveaux montants (`node_figures/amd`)

L'amendement (nouveaux montants – projet de budget)

Header d'un Line by line PEL1 en anglais :

Heading	2003	PDB 2004	DB 2004	PE1 2004	DIFFERENCE

Au Conseil 2ième lecture :

Le budget de l'année N-1 (`node_figures/d_n_1`)

Le projet de budget (`node_figures/d_n`)

Les chiffres PEL1 (`node_figures/pel1`)

Les nouveaux montants (`node_figures/amd`)

L'amendement (nouveaux montants – chiffres PEL1)

Header d'un Line by line COL2 en français :

Intitulé	2003	PB 2004	PE1 2004	CSL2 2004	Différence

Au Parlement 2ième lecture :

Le budget de l'année N-1 (`node_figures/d_n_1`)

Les chiffres PEL1 (`node_figures/pel1`)

Les chiffres COL2 (`node_figures/col2`)

Les nouveaux montants (`node_figures/amd`)

L'amendement (nouveaux montants – chiffres COL2)

Header d'un Line by line PEL2 en anglais :

Heading	2003	PE1 2004	CSL2 2004	PE2 2004	DIFFERENCE

Reportlbl2rtf.xsl inclut la feuille de style reportlbl2rtf-utils.xsl.

Reportlbl2rtf.xsl comprend toute la logique de la conversion. C'est dans cette feuille de style qu'on recherche les montants, que les totaux sont calculés et que les templates de reportlbl2rtf-utils sont appelées.

Reportlbl2rtf-utils.xsl contient huit templates capables de produire le code RTF des six types de lignes que l'on peut trouver dans un rapport Line by line, ainsi que celui du header et du footer du document :

`createLBLHeader`: sort un des header présentés ci-dessus (dépend de la valeur de`Lecnum`).

`createLBLFooter`: sort le footer (dépend de la valeur de`Lecnum`).

`outputROW_Exp_NoMt`: sort le code RTF d'une ligne budgétaire classifiée en tant que dépenses et qui n'a pas de chiffres (uniquement des totaux).

`outputROW_Rev_NoMt`: même chose que`outputROW_Exp_NoMt`mais pour des recettes.

`outputROW_Exp_SimpleMt`: sort le code RTF d'une ligne budgétaire classifiée en tant que dépenses et qui a des chiffres.

`outputROW_Rev_SimpleMt`: même chose que`outputROW_Exp_SimpleMt`mais pour des recettes.

`outputROW_Exp_Total`: sort le code RTF du total d'une ligne budgétaire classifiée en tant que dépenses mais qui n'a pas de chiffres.

`outputROW_Rev_Total`: même chose que`outputROW_Exp_Total`mais pour des recettes.

Par exemple, dans l'extrait de Line par line suivant, la première ligne a été générée par`outputROW_Rev_NoMt`, les deux suivantes par`outputROW_Rev_SimpleMt`et la dernière par`outputROW_Rev_Total`:

5 1 1	Proceeds from letting and subletting immovable property and reimbursement of charges connected with lettings					
5 1 1 0	Proceeds from letting and subletting immovable property	p.m.		p.m.	p.m.	0
5 1 1 1	Reimbursement of charges connected with lettings	p.m.		p.m.	p.m.	0
Total 5 1 1		0	0	0	0	0

Dans l'extrait suivant portant sur des dépenses, la première ligne a été générée par outputROW_Exp_NoMt, les deux suivantes par outputROW_Exp_SimpleMt et la dernière par outputROW_Exp_Total :

02 03 04	Completion of previous programmes									
02 03 04 01	Completion of programmes prior to 1999 (NCE)	–	8 540 000			–	1 000 000	–	1 000 000	0	0
02 03 04 02	Completion of the fifth EC framework programme (1998 to 2002) (NCE)	–	74 000 000			–	52 870 000	–	52 870 000	0	0
Total 02 03 04		0	84 540 000	0	0	0	53 870 000	0	53 870 000	0	0

Le rapport ligne par ligne "correction"

Comme expliqué dans la présentation du Line by line, le rapport "correction" ne contient que des lignes portant sur des frais administratifs, et dont les montants calculés par "SEI-BUD+Amendment" sont différents de ceux de Badge-Bud.

Le but du rapport "correction" est de mettre en avant ces divergences. Il contient donc une colonne avec les montants "SEI-BUD+Amendment", suivie d'une colonne avec les montants Badge-Bud, suivie d'une dernière colonne montrant les différences entre les deux.

En COL1, on a:

Le budget de l'année N-1 (`node_figures/pd_n_1`)

L'avant-projet de budget de l'année courante (`node_figures/pd_n`)

Les nouveaux montants "SEI-BUD+Amendment" (`node_figures/amd`)

Les montants Badge-Bud (`node_figures/corrected`)

Les différences entre "SEI-BUD+Amendment" et Badge-Bud

En PEL1 :

L'avant-projet de budget (`node_figures/pd_n`)

Le projet de budget (`node_figures/d_n`)

Les nouveaux montants "SEI-BUD+Amendment" (`node_figures/amd`)

Les montants Badge-Bud (`node_figures/corrected`)

Les différences entre "SEI-BUD+Amendment" et Badge-Bud

En COL2 :

Le projet de budget (`node_figures/d_n`)

Les chiffres PEL1 (`node_figures/pel1`)

Les nouveaux montants "SEI-BUD+Amendment" (`node_figures/amd`)

Les montants Badge-Bud (`node_figures/corrected`)

Les différences entre "SEI-BUD+Amendment" et Badge-Bud

En PEL2 :

Les chiffres PEL1 (`node_figures/pel1`)

Les chiffres COL2 (`node_figures/col2`)

Les nouveaux montants "SEI-BUD+Amendment" (`node_figures/amd`)

Les montants Badge-Bud (`node_figures/corrected`)

Les différences entre "SEI-BUD+Amendment" et Badge-Bud

Le filtre de génération du Line by line correction est`ReportLbl2RtfCorrectionFilter`. Il hérite de la classe`XSLTFilter`faisant partie du framework Xef.

Le filtre fonctionne sur le même principe que celui du Line by line "classique". Il n'est constitué que d'une feuille de style,`reportlbl2rtf-correction.xsl`, qui inclut la feuille de style`reportlbl2rtf-correction-`

`utils.xsl`.`Reportlbl2rtf-correction.xsl`contient la logique de la transformation et reportlbl2rtf-correction-utils.xsl contient les templates de génération de code RTF. Des six templates générant des lignes budgétaires, seules`outputROW_Exp_SimpleMt`et`outputROW_Rev_SimpleMt`sont conservées ici, puisqu'on ne présente que des lignes contenant des chiffres.

`Reportlbl2rtf-correction.xsl`prend les mêmes paramètres que reportlbl2rtf.xsl. Seules les lignes ayant des montants corrigés (`NODE_FIGURES/CORRECTED`) sont traitées et elles ne sont affichées qu'à la condition qu'au moins un des montants calculés par "SEI-BUD+Amendment" (`NODE_FIGURES/AMD`) diffère du montant de Badge-Bud.

Voici un extrait d'un Line by line "correction" du Parlement 2ième lecture :

Heading	PE1 2004		CSL2 2004		PE2 2004		PE2 2004 Corrected Amount		DIFFERENCE	
XX 01 02 01 01 Auxiliary staff	63 975 000	63 975 000	63 975 000	63 975 000	64 696 156	64 696 156	**64 599 035**	**64 599 035**	-97 121	-97 121
					238 925	*238 925*	*322 142*	*322 142*	*83 217*	*83 217*
XX 01 02 01 02 Agency staff and technical and administrative assistance in support of different activities	23 100 000	23 100 000	23 100 000	23 100 000	23 000 862	23 000 862	**22 967 154**	**22 967 154**	-33 708	-33 708
					84 945	*84 945*	*113 828*	*113 828*	*28 883*	*28 883*
XX 01 02 01 03 National and international civil servants and private-sector staff temporarily assigned to the institution	31 022 000	31 022 000	31 022 000	31 022 000	30 888 866	30 888 866	**30 843 596**	**30 843 596**	-45 270	-45 270
					114 075	*114 075*	*152 864*	*152 864*	*38 789*	*38 789*

Le rapport ligne par ligne "corrected"

Le rapport Line by line "corrected" est très semblable au Line by line "classique". La seule différence entre ces deux rapports est que, pour toute les lignes pour lesquelles il existe un montant corrigé, on affiche ce montant corrigé au lieu du montant calculé par "SEI-BUD+Amendment" (et on le souligne pour indiquer qu'il provient de Badge-Bud).

Le filtre de génération du Line by line "corrected" est`ReportLbl2RtfCorrectedFilter`. Il hérite de la classe`XSLTFilter`faisant partie du framework Xef.

Le principe est à nouveau le même que celui du Line by line "classique" : une seule feuille de style (`reportlbl2rtf-corrected.xsl`) génère le rapport. Elle spécifie toute la logique de calcul des montants et de l'affichage des lignes. Elle inclut reportlbl2rtf-corrected-utils.xsl, qui contient les templates de génération de code RTF.

Voici un extrait d'un Line by line "corrected" du Conseil 1ière lecture. Les montants de la ligne 03 01 02 01 proviennent de Badge-Bud et non de "SEI-BUD+Amendment" :

Intitulé	2004		APB 2005		PB 2005 New Corrected Situation		PB-APB		% PB-2004	
03 Concurrence										
03 01 Dépenses administratives du domaine politique Concurrence										
03 01 01 Dépenses liées au personnel en activité du domaine politique «Concurrence»	55 444 947	55 444 947	60 336 010	60 336 010	59 839 220	59 839 220	**-496 790**	**-496 790**	*7.9*	*7.9*
	268 135	*268 135*					*0*	*0*	*-100*	*-100*
03 01 02 Personnel externe et autres dépenses de gestion à l'appui du domaine politique «Concurrence»										
03 01 02 01 Personnel externe	9 217 960	9 217 960	8 918 561	8 918 561	8 587 472	8 587 472	**-331 089**	**-331 089**	*-6.8*	*-6.8*
03 01 02 11 Autres dépenses de gestion	4 386 954	4 386 954	4 926 847	4 926 847	4 523 903	4 523 903	**-402 944**	**-402 944**	3.1	3.1
	204 946	*204 946*					*0*	*0*	*-100*	*-100*
Total 03 01 02	13 604 914	13 604 914	13 845 408	13 845 408	13 111 375	13 111 375	**-734 033**	**-734 033**	*-5.6*	*-5.6*
	204 946	*204 946*	*0*	*0*	*0*	*0*	*0*	*0*	*-100*	*-100*

5.7.19. Producing the SEIAMD specific CatPol report

Le CatPol est un rapport de synthèse reprenant l'affectation des montants aux différentes politiques et catégories de l'Union européenne.

Au cours de la procédure budgétaire 2006, les institutions politiques ont changé le contenu du rapport CatPol. Pour pouvoir effectuer tous les calculs demandés dans le nouveau rapport, la méthode initiale de calcul ne suffisait plus. Voilà pourquoi, une nouvelle méthode de calcul a été introduite, mais l'ancienne représentation continue à être disponible dans le système SEIAMD.

5.7.19.1. Principe d'utilisation

La nouvelle méthode de calcul du rapport CATPOL se base sur la même méthodologie que le calcul du rapport CatPol de SEI-BUD.

Les administrateurs de l'application éditent avec l'outil de contrôle un patron en format VL0. Le patron porte le nom `AAAAVOL0-AMD_CATPOL_REPORTS` où `AAAA` est l'année budgétaire en cours d'élaboration. Il se présente en quatre sections, pour les deux lectures du Conseil et pour les deux lectures du Parlement européen.

L'administrateur insère les éléments `reuse` décrits au chapitre 5.4.5 pour faire apparaitre un montant spécifique.

Les utilisateurs demandent le calcul du rapport CatPol dans l'interface Webint (interface Web aux amendements).

Le système calcule d'abord une intégration pour tous les volumes de la publication budgétaire courante afin d'intégrer les chiffres des amendements dans les volumes de la publication budgétaire de référence.

Le système calcule ensuite tous les montants susceptibles d'être appelés dans un élément `reuse`.

Le patron préparé préalablement par l'administrateur est lu dans le repository XML. En fonction de la phase pour laquelle le rapport CatPol a été demandé, on récupère la bonne section (Parlement/Conseil, première ou deuxième lecture).

Les éléments `reuse` présents dans le patron sont remplacés par les valeurs calculées précédemment.

Le document VLO résultant est converti en RTF en suivant la filière de conversion VL0 vers RTF usuelle.

Si l'utilisateur l'a demandé, un deuxième fichier en format CSV est généré. Il indique pour chaque code CatPol possible les lignes budgétaires et les montants qui ont été sommés.

Les deux documents sont mis à disposition de l'utilisateur dans l'interface Webint.

5.7.19.2. Description générale des classes entrant dans le calcul des montants

Le calcul des montants se fait dans les classes des modules `com.softwareag.belgium.seibud.amdbuilder.reports.catpol.bo` et `com.softwareag.belgium.seibud.amdbuilder.reports.catpol.processing`.

Classe	Description
`AbbMetadatLoader`	Classe qui gère l'extraction et la mise à disposition des codes CatPol, des montants maximaux et du produit intérieur brut à partir du volume 4 d'une certaine publication.
`Category`	Dans le cadre de l'extraction de données pour l'ancien CatPol, cette classe gère les catégories de plus haut niveau. Elle génère notamment tout la représentation XML des catégories. De plus, elle permet de trouver tous les sous-groupes ou codes politiques qui sont associés à une catégorie.

Classe	Description
`Cluster`	Dans le cadre de l'extraction de données pour l'ancien CatPol, cette classe gère les sous-groupes. Elle génère notamment tout la représentation XML des sous-groupes. Elle permet aussi de trouver tous les codes politiques qui font partie d'un sous-groupe donné.
`Figure`	Cette classe regroupe toutes les données nécessaires au calcul du CatPol pour une ligne budgétaire donnée.
`Label`	Cette classe gère les libellés des sous-groupes (uniquement utilisée dans le cadre des anciens rapports CatPol).
`Politics`	Dans le cadre de l'extraction de données pour l'ancien CatPol, cette classe gère les codes politiques. Elle génère notamment tout la représentation XML groupes politiques (y compris les montants). Elle permet aussi de trouver toutes les lignes budgétaires qui ont un code politique donné.
`Report`	Cette classe s'occupe à elle seule de générer tous les montants nécessaires pour le calcul du CatPol. Une description plus détaillée est donné plus bas.
`CatPolBuilder`	Cette classe sert de point d'entrée à la génération des sommes et au fichier XML interne qui détaille les lignes budgétaires qui ont contribué au calcul d'une somme pour un certain code politique.

5.7.19.3.Calcul des montants pour le rapport CatPol actuel

Le gros des calculs s'effectue dans la classe `com.softwareag.belgium.seibud.amdbuilder.reports.catpol.bo.Report`.

Suivant la phase dans laquelle on se trouve, on va rechercher les chiffres dans certaines publications et pas dans d'autres. Par exemple, en première lecture du parlement, on récupère les chiffres intégrés issus des amendements, les chiffres du projet de budget et les chiffres de l'avant-projet de budget. Pour chaque publication, on effectue les étapes qui suivent.

On effectue une recherche dans la base de données pour retrouver toutes les lignes budgétaires ayant un code politique donné. Les lignes budgétaires qui ont l'attribut `TP_HORIZONTAL` qui vaut `true` sont exclues de la recherche. En effet, on considère que ces lignes sont un récapitulatif de se qui se trouve déjà ailleurs (lignes XX du volume 4).

Les sommes sont construites d'après le code politique. Un chiffre entre certainement dans la somme du code politique auquel il appartient. Une deuxième somme est faite pour tous les codes qui commencent par le code donné.

Si le code politique contient un point, on ôte un caractère à la fin du code. Si le nouveau caractère le plus à droite est un point, il est également ôté. On produira une somme pour le nouveau code obtenu. Le processus est alors continué récursivement.

Si le code politique ne contient pas de point, on ne le décompose pas plus finement.

Le code politique est aussi combiné avec l'alias de la ligne budgétaire. On effectue une somme pour le nouveau code obtenu et on décompose le code politique selon les points qu'il contient. On supprime du code le point le plus à droite ainsi que tout ce qui suit et on répète la procédure récursivement jusqu'à ce que le code ne contienne plus de points.

5.7.19.4.Calcul du détail des lignes budgétaires

Chaque fois qu'une ligne budgétaire participe à une somme, elle est insérée dans une liste qui permet de retrouver toutes les lignes budgétaires ayant participé à la somme pour une ligne.

Cette liste est en fait intégrée au fichier XML généré pour l'ancien rapport CatPol. C'est donc le fichier XML de l'ancien CatPol qui sert de base pour générer le détail des lignes budgétaires.

5.7.19.5. Enchaînement des actions

L'enchaînement des actions pour calculer un rapport CatPol est assuré par la classe `com.softwareag.belgium.seibud.rmg.reports.CatPolReport` dont la méthode principale est `generate`.

5.7.19.6. Ancienne représentation du rapport CatPol

Génération de l'XML

La structure de l'instance XML reflète celle du rapport CatPol : elle est organisée en une suite de catégories contenant des politiques. Dans chaque politique, on trouve la succession des lignes qui lui sont affectées, ainsi que leurs chiffres, qui permettront de calculer les totaux.

Les catégories sont représentées par l'élément `CATEGORY` et les politiques par `POLITICS`. Leurs libellés sont contenus dans un élément `LANG_LABELS` avec un ou plusieurs `LANG_LABEL` (un par libellé et version linguistique) :

```
<CATEGORY ID_CATEGORY="4">
<LANG_LABELS>
<LANG_LABEL ID_LANGUAGE="DE">Externe Politikbereiche</LANG_LABEL>
<LANG_LABEL ID_LANGUAGE="EN">External Actions</LANG_LABEL>
<LANG_LABEL ID_LANGUAGE="FR">Actions extérieures</LANG_LABEL>
        ...
</LANG_LABELS>
        ...
<POLITICS CD_POLITICS="403" ID_POLITICS="39" TP_PRESENTATION="">
<LANG_LABELS>
<LANG_LABEL ID_LANGUAGE="DE">Humanitäre Hilfe</LANG_LABEL>
<LANG_LABEL ID_LANGUAGE="EN">Humanitarian aid</LANG_LABEL>
<LANG_LABEL ID_LANGUAGE="FR">Aide humanitaire</LANG_LABEL>
        ...
</LANG_LABELS>
        ...
</POLITICS>
        ...
</CATEGORY>
```

En plus des catégories et des politiques, il existe un autre niveau de regroupement, le cluster. Il ne s'agit pas d'une classification "officielle" mais d'un regroupement décidé par l'utilisateur lors de la demande de CatPol. En effet, "SEIBUD+Amendement" offre la possible d'associer une présentation particulière à un rapport CatPol (option "Select a presentation" de l'écran de la demande de CatPol). La présentation a été préalablement créée et sauvegardée par l'utilisateur (bouton "CAT/POL Presentations" du menu principal de "SEIBUD+Amendement"). Les possibilités offertes sont :

Déplacement de politiques vers le haut ou vers le bas dans une catégorie

Regroupement de politiques dans une catégorie

Le regroupement est symbolisé par un élément `cluster`, qui possède également des libellés :

```
<CATEGORY ID_CATEGORY="4">
<LANG_LABELS>...</LANG_LABELS>
        ...
<CLUSTER ID_CLUSTER="1" CD_CLUSTER="12">
<LANG_LABELS>...</LANG_LABELS>
        ...
<POLITICS CD_POLITICS="403" ID_POLITICS="39" TP_PRESENTATION="">
<LANG_LABELS>...</LANG_LABELS>
</POLITICS>
        ...
```

```
</CLUSTER>
        ...
</CATEGORY>
```

Les chiffres sont centralisés dans un élément `FIGURES` fils de `POLITICS`. On y trouve les éléments :

`PD/N_1`: les chiffres du budget de l'année N-1

`PD/N`: les chiffres de l'avant-projet de budget

`D/N_1`: les chiffres du budget de l'année N-1 (devraient être identiques à`PD/N_1`)

`D/N`: les chiffres du projet de budget

`PEL1`: les chiffres du Parlement, 1ère lecture

`COL2`: les chiffres du Conseil, 2ième lecture

`AMD`: les nouveaux montants

Les lignes sont représentées par des éléments `FIG_UNIT`. On y retrouve le minimum d'informations utiles : l'alias (`LB_ALIAS`), les différents montants (`MT_ENGAGEMENT`,`MT_PAIEMENT`,`MT_RESERVE_ENGAGEMENT`,`MT_RESERVE_PAIEMENT`,`MT_RECETTE`), crédits dissociés ou non dissociés (`TP_DIFFNDIFF`), dépenses obligatoires ou non obligatoires (`TP_COMPNCOMP`) :

```
<FIGURES>
<PD>
<N>
<FIG_25>
<FIG_UNIT ID_NODE="5273" ID_NODE_INTEGRATION="534128" ID_TRANSACTION="" LB_ALIAS="05 01 04
01">
<TP_DIFFNDIFF>NDIFF</TP_DIFFNDIFF>
<TP_COMPNCOMP>COMP</TP_COMPNCOMP>
<MT_RECETTE>0</MT_RECETTE>
<MT_ENGAGEMENT>3860000</MT_ENGAGEMENT>
<MT_RESERVE_ENGAGEMENT>0</MT_RESERVE_ENGAGEMENT>
<MT_PAIEMENT>3860000</MT_PAIEMENT>
<MT_RESERVE_PAIEMENT>0</MT_RESERVE_PAIEMENT>
</FIG_UNIT>
<FIG_UNIT ID_NODE="5280" ID_NODE_INTEGRATION="534137" ID_TRANSACTION="" LB_ALIAS="05 02 01
01">
<TP_DIFFNDIFF>NDIFF</TP_DIFFNDIFF>
<TP_COMPNCOMP>COMP</TP_COMPNCOMP>
<MT_RECETTE>0</MT_RECETTE>
<MT_ENGAGEMENT>532000000</MT_ENGAGEMENT>
<MT_RESERVE_ENGAGEMENT>0</MT_RESERVE_ENGAGEMENT>
<MT_PAIEMENT>532000000</MT_PAIEMENT>
<MT_RESERVE_PAIEMENT>0</MT_RESERVE_PAIEMENT>
</FIG_UNIT>
        ...
```

L'élément racine de l'instance XML produite est `REPORT`. Il contient des informations de gestion sur la demande de rapport CatPol :

```
<REPORT ID_REPORTREQUEST="1576">
<DTT_BEGIN>2003-12-17</DTT_BEGIN>
<DTT_END />
<DTT_DEADLINE />
<TP_REPORT>CATPOL</TP_REPORT>
<STATE>S_INITIALIZED</STATE>
<LB_LANGUAGES>EN</LB_LANGUAGES>
<LB_RESULTS>ALL_AMENDMENTS</LB_RESULTS>
<LB_PARTS />
```

```
<LB_FIGURES>FIGURES_AFTER_COUNCIL</LB_FIGURES>
<LB_REPORT>CATPOL COBU au 17/12</LB_REPORT>
<LB_INFO>0</LB_INFO>
<ID_USER>59</ID_USER>
<ID_DEFAULTLANGUAGE>FR</ID_DEFAULTLANGUAGE>
```

L'extraction est effectuée par la méthode statique `process` de la classe `CatPolBuilder` (package com.softwareag.belgium.seibud.amdbuilder.reports.catpol.processing).

Six "business objects" sont utilisés au cours de l'extraction (les classes se trouvent dans le package`com.softwareag.belgium.seibud.amdbuilder.reports.catpol.bo`):

`Report`: représentation d'une demande de rapport CatPol. Correspond à la table REPORTREQUESTS de la base de données. Dans l'instance XML du CatPol : element `REPORT`.

`Category`: représentation d'une catégorie. Correspond à la table CATEGORIES de la base de données. Dans l'instance XML du CatPol : element `CATEGORY`.

`Cluster`: représentation d'un cluster. Correspond à la table CLUSTERS de la base de données. Dans l'instance XML : element `CLUSTER`.

`Politics`: représentation d'une politique. Correspond à la table POLITICS de la base de données. Dans l'instance XML : element `POLITICS`.

`Figure`: représentation des chiffres d'une ligne budgétaire. Correspond à la table VERSIONS_FIGURES_INTEGRATION de la base de données. Dans l'instance XML : element `FIG_UNIT`.

`Label`: représentation d'un libellé d'une catégorie, d'une politique ou d'un cluster. Correspond à la table CODES de la base de données. Dans l'instance XML : element `LANG_LABEL`.

Génération du rapport RTF

Les demandes de rapport CatPol sont gérées par la méthode `generate` de la classe `CatPolReport`, (package com.softwareag.belgium.seibud.rmg.reports).

Une fois l'XML produit par l'appel de la méthode `CatPolBuilder.process`, le filtre XSLT est appliqué. Il est contenu dans la classe`ReportCp2RtfFilter`, qui hérite de la classe`XSLTFilter` du framework Xef.

Le filtre est constitué d'une seule feuille de style:`reportcp2rtf.xsl`. Elle prend pour paramètres les informations suivantes :

`Language`: la version linguistique dans laquelle a été demandé le CatPol

`Lecnum`: la lecture :

`FIGURES_BEFORE_PARLIAMENT`: Conseil 1ière lecture

`FIGURES_BEFORE_COUNCIL`: Parlement 1ière lecture

`FIGURES_AT_COUNCIL`: Conseil 2ième lecture

`FIGURES_AFTER_COUNCIL`: Parlement 2ième lecture

`TitlesDir`: le chemin vers le répertoire qui contient les fichiers de titres

`Year`: l'année budgétaire en cours.

Suivant la valeur de`Lecnum`, les montants présentées ne seront pas les mêmes.

Au Conseil 1ière lecture, le CatPol contient :

Le budget de l'année N-1 (`FIGURES/PD/N_1`)

L'avant-projet de budget (`FIGURES/PD/N`)

Les nouveaux montants (`FIGURES/AMD`)

Les différences (nouveaux montants – avant-projet de budget)

Le pourcentage "DB – Année N-1"

Header d'un CatPol COL1 en français:

Intitulé	2003	APB 2004	PB 2004	PB-APB	% PB-2003

Au Parlement 1ière lecture:

Le budget de l'année N-1 (`FIGURES/D/N_1`)

L'avant-projet de budget (`FIGURES/PD/N`)

Le projet de budget (`FIGURES/D/N`)

Les nouveaux montants (`FIGURES/AMD`)

Les différences (nouveaux montants – projet de budget)

Header d'un CatPol PEL1 en français:

Intitulé	2003	APB 2004	PB 2004	PE1 2004	Différence

Au Conseil 2ième lecture:

Le budget de l'année N-1 (`FIGURES/D/N_1`)

Le projet de budget (`FIGURES/D/N`)

Les chiffres PEL1 (`FIGURES/PEL1`)

Les nouveaux montants (`FIGURES/AMD`)

Les différences (nouveaux montants – chiffres PEL1)

Header d'un CatPol COL2 en français :

Intitulé	2003	PB 2004	PE1 2004	CSL2 2004	Différence

Au Parlement 2ième lecture:

Le budget de l'année N-1 (`FIGURES/D/N_1`)

Les chiffres PEL1 (`FIGURES/PEL1`)

Les chiffres COL2 (`FIGURES/COL2`)

Les nouveaux montants (`FIGURES/AMD`)

Les différences (nouveaux montants – chiffres COL2)

Header d'un CatPol PEL2 en anglais :

Heading	2003	PE1 2004	CSL2 2004	PE2 2004	DIFFERENCE

`Reportcp2rtf.xsl`inclut deux feuilles de style :

`reportcp2rtf-var.xsl` contient les variables globales à la transformation. Ces variables servent à calculer les différents totaux (d'une catégorie, d'une politique, d'un cluster)

`reportcp2rtf-utils.xsl` contient des templates capables de produire le code RTF des quatre types de lignes que l'on peut trouver dans un rapport CatPol, ainsi que celui du header et du footer du document :

createCATPOLHeader: sort un des header présentés ci-dessus (dépend de la valeur de Lecnum).

`createCATPOLFooter`: sort le footer (dépend de la valeur de`Lecnum`).

`outputPoliticRow`: sort le code RTF d'une ligne d'une politique.

`outputClusterRow`: sort le code RTF d'une ligne d'un cluster.

`outputCategoryRow`: sort le code RTF des trois lignes d'une catégorie : une première ligne pour le total général de la catégorie, une deuxième pour les crédits dissociés et une troisième pour les crédits non dissociés.

`outputTotalGeneralRow`: sort le code RTF des trois lignes du total général du CatPol: une première ligne pour le total général de toutes les catégories, une deuxième pour les crédits dissociés et une troisième pour les crédits non dissociés.

Reportcp2rtf.xsl contient l'aspect logique de la conversion. C'est dans cette feuille de style qu'on recherche les montants, que les totaux sont calculés et que les templates de reportcp2rtf-utils sont appelées.

Par exemple, dans l'extrait de Catpol suivant, la première ligne a été générée par `outputPoliticRow`, la suivante également, et les trois dernières ont été générées par `outputCategoryRow`:

(11) Agricultural expenditure (excluding rural development)	40 082 450 000	40 082 450 000	41 330 599 000	41 330 599 000	40 090 285 000	40 090 285 000	**40 245 285 000**	**40 245 285 000**	155 000 000	155 000 000
(12) Rural development and supporting measures	4 698 000 000	4 698 000 000	6 536 000 000	5 448 000 000	6 536 000 000	5 448 000 000	**6 536 000 000**	**5 448 000 000**	0	0
Total Agricultural policy	44 780 450 000	44 780 450 000	47 866 599 000	46 778 599 000	46 626 285 000	45 538 285 000	**46 781 285 000**	**45 693 285 000**	155 000 000	155 000 000
CE	40 082 450 000	40 082 450 000	41 330 599 000	41 330 599 000	40 090 285 000	40 090 285 000	**40 245 285 000**	**40 245 285 000**	155 000 000	155 000 000
NCE	4 698 000 000	4 698 000 000	6 536 000 000	5 448 000 000	6 536 000 000	5 448 000 000	**6 536 000 000**	**5 448 000 000**	0	0

Exemple de lignes générées par `outputTotalGeneralRow`:

General Total	0	0	11 839 463 386	5 680 410 682	11 754 846 816	5 003 918 313	**11 771 242 284**	**5 105 513 728**	16 395 468	101 395 415
CE	0	0	1 782 524 301	1 778 586 501	1 704 420 056	1 704 420 056	**1 701 220 056**	**1 697 282 056**	-3 200 000	-7 138 000
NCE	0	0	10 056 939 085	3 901 824 181	10 050 426 760	3 299 498 257	**10 070 022 228**	**3 408 231 672**	19 595 468	108 733 415

5.8. The Translation Requests and Translation Followup System

5.8.1. Constraints

The Business Objects, related to the Translation Requests (TR) and Translation Follow-up (TF), have been created according to the following constraints:

Allow non-blocking translations to enable concurrent work of authors and translators

Have a different BO for the translations and the revisions (in order to clearly show the difference between the two)

Allow the detailed follow-up of translation/revision requests: how far is the request prepared, how many translations have already returned,…

Parallelize the preparation of the translation requests : A single demand may require the generation of up to 50 different documents (2 for each language to translate and 3 for the original language)

Plan, for a near future, the use of another translation service than Poetry

5.8.2. Guiding principles

We distinguish a user “Demand” from the “Requests” sent to the translation service.

Therefore, a single translation/revision user ‘ Demand , may generate **n** translation/revision “Requests” to the translation service (the only translation service is Poetry for the moment).

The user must be able to follow the life cycle of each “request” associated to his “demand”.

Note: A “Demand” is composed of a SEIBUD part, and a Translation Service part.

When a “demand” is created, it generates all the suitable requests, and waits.

When a “demand” is waiting, one can search for the states of the different requests that it is made of: for example, for a non-blocking translation demand, on 22 requests sent (for the 22 different languages), 10 can be waiting, 10 released, and 2 in error.

Once all the requests are released, the demand state is “released”.

A “request” is first created, than prepared, than sent, than released (possibly several times) or in error if the returned document is not correct.

The preparation and the dispatch of each request is handled in different BPSRV tasks, to promote parallelization of the preparation and management tasks of translation demands (one request for each target language).

Until a suitable framework is available within the BPSRV, the BO (the one specific to the translation service) will contain the synchronization logic of the different tasks of the "demand" (preparation, sending, return).

5.8.3. Business Objects description

A "translation" demand is composed of a part specific to SEIBUD (SeibudTranslqationUserDemand) and a part specific to the translation service (TranslationServicePoetryUserDemand).

5.8.3.1. SeibudTranslationUserDemand

A SeibudTranslationUserDemand is a demand created by a user, corresponding to a translation or revision of a fragment of a publication.

It contains all the information related to the publication/fragment to translate/revise.

All the data entered in the ControlTool to create the "demand" is stored in the DataBase.

Depending on the translation service associated to the demand (only Poetry for the moment), an appropriate TranslationServiceUserDemand BO (TranslationServicePoetryUserDemand for now) will be associated to the SeibudTranslationUserDemand .

The only state for a SeibudTranslationUserDemand is created. Since a SeibudTranslationUserDemand is always associated to a single TranslationServiceUserDemand, the other states corresponding to the evolution of the "demand" is stored in the TranslationServiceUserDemand.

Note: For the moment another mock translation service TranslationServiceAnotherUserDemand is available to show how things should be done.

The table SEIBUD_TRANSL_USER_DEMANDS is associated to the BO SeibudTranslationUserDemand: it contains all the data specific to SEIBUD for a translation/revision demand.

5.8.3.2. TranslationServiceUserDemand

A BO TranslationServiceUserDemand corresponds to the part of the demand specific to the translation service used.

It is an interface, so that the SeibudTranslationUserDemand does not explicitly address a specific translation service.

5.8.3.3. TranslationServicePoetryUserDemand

This is the BO specific to the translation service POETRY.

This BO contains all the methods to prepare the requests: some PoetryRevisionLVRequest or PoetryTranslationLVRequest, depending on the type of demand.

The table POETRY_TRANSL_USER_DEMANDS is associated to this BO.

5.8.3.4. TranslationServiceAnotherUserDemand

This is a mock BO to represent another translation service that might be used.

The purpose is to see how it cab be integrated in the rest of the architecture.

The table ANOTHER_TRANSL_USER_DEMANDS is associated to this BO.

From now on, only the TranslationServicePoetryUserDemand will be considered.

5.8.3.5. PoetryXxxLVRequest

A request is specific to a linguistic version and consists of all the documents sent to Poetry, and possibly all the documents returned.

It can either be a translation (translate a document from a source language to a target language) or a revision (revise a document in a given language) request.

The basic life cycle of a request is:

created -> prepared -> sent -> error | released

Note: the specific information for the translation service, common for all the requests, is stored in the TranslationServicePoetryUserDemand. The PoetryXxxLVRequest contains only the information specific to the request (target language, deadline…)

5.8.3.6. PoetryTranslationLVRequest

The PoetryTranslationLVRequest (LV=Linguistic Version) is a translation request. It can be blocking or non-blocking.

Non-blocking translation requests can be released several times by the translators.

Non-blocking translation requests can be pending if the publication is locked when the translation returns (The translation is imported when the lock is released).

The table POETRY_TRANS_LV_REQUESTS is associated to this BO.

5.8.3.7. PoetryRevisionLVRequest

The PoetryRevisionLVRequest (LV=Linguistic Version) is a revision request. It can only be blocking.

The table POETRY_REV_LV_REQUESTS is associated to this BO.

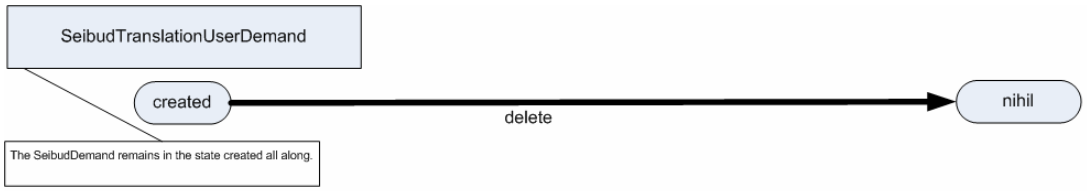
5.8.3.8. PoetryDocument

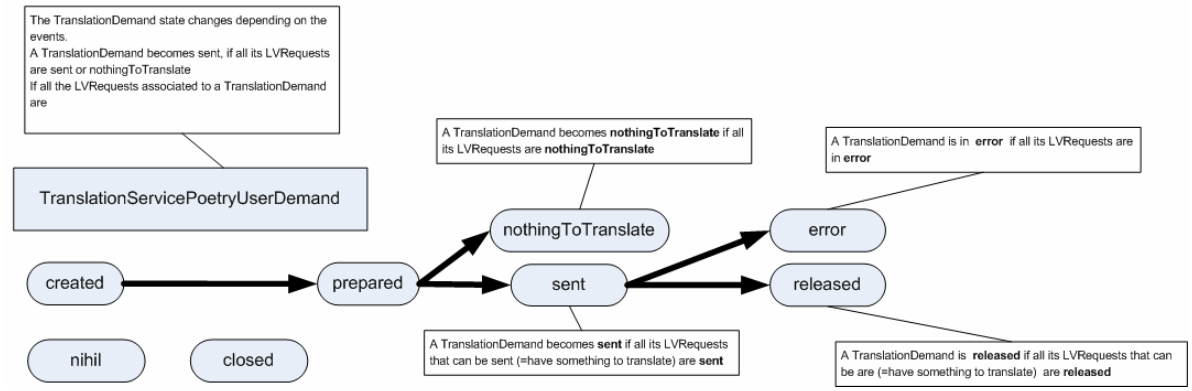
PoetryDocuments are associated to PoetryRequests. They are of different kind (“ORI”, “CMP”, “PRT”, “REF1”, “REF2”) and can be “sent” or “received”.

Once a PoetryDocument is created, its content is never changed, only its state will evolve.

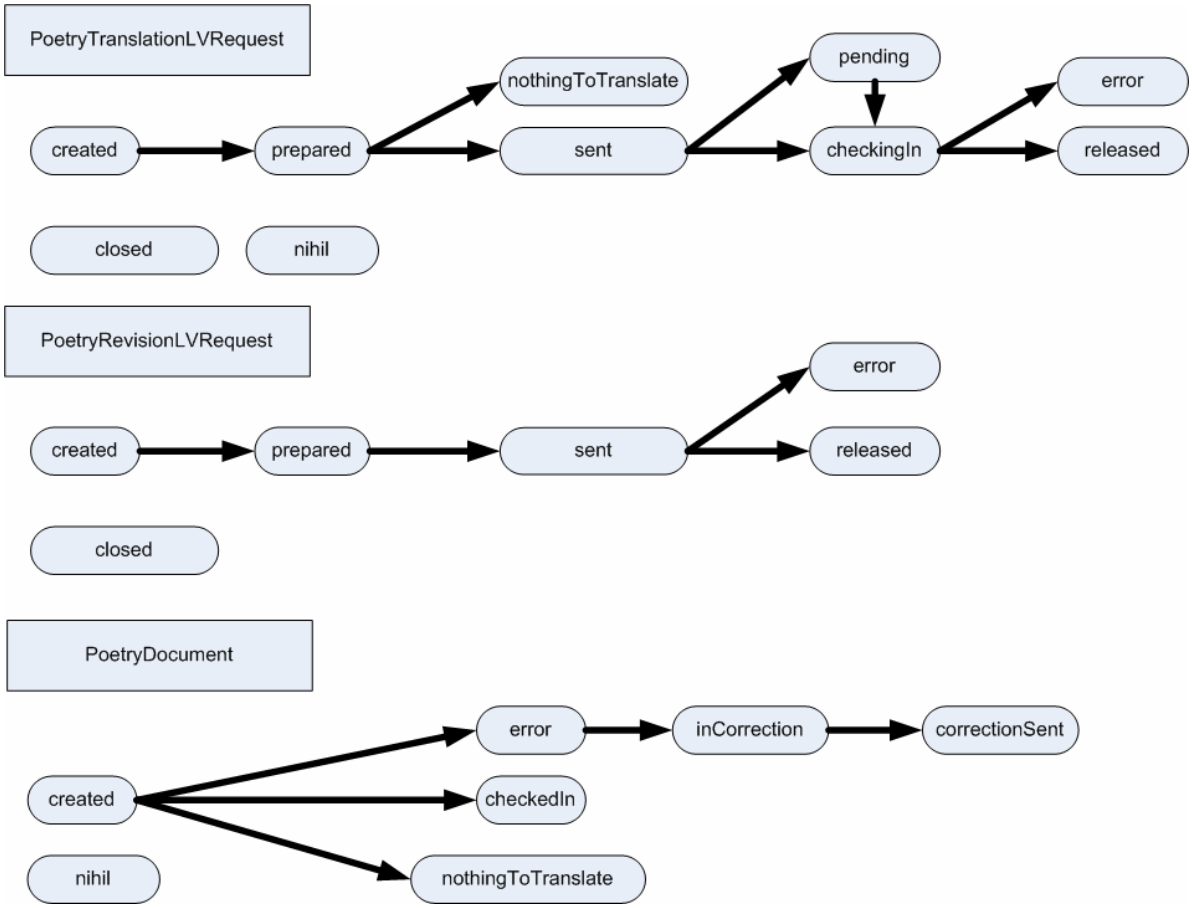
5.8.4. Life cycles

5.8.4.1. Demands

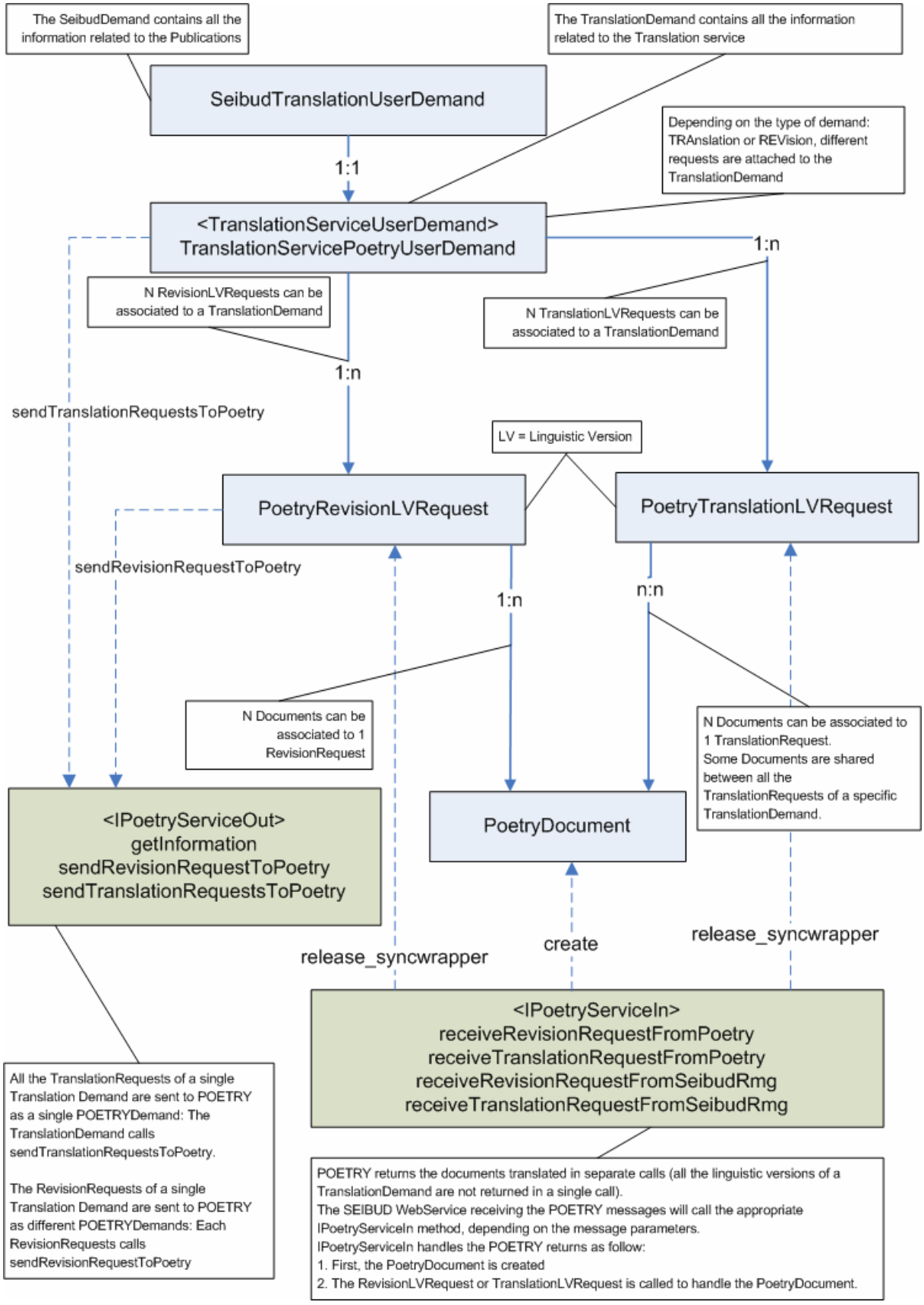




5.8.4.2. Requests and Documents



5.8.5. TR/TF Business Objects relations



5.9. Amendments integration process

By this process, the changes contained in a list of amendments are integrated in a target publication.

5.9.1. Principles

The Web-based interface to amendments offers the “Integration” button to perform the actions. It chains the following operations in a single process:

select a set of amendments,

solve the text conflicts inside them,

send the chosen set of amendments to the Repository Manager,

integrate the figures,

generate the set of changes to be applied using the meta-transactions formalism,

build a new target publication by copying an existing one,

apply the set of changes to the target publication.

The semantic validation of the integration process stays manual. In particular, it is useful to perform a figures comparison between the line by line report and an extraction viewed with Figure Tool.

If the target publication already exists and must not be deleted, the automated process cannot be used because it initialises the target publication with the content of another one. In that case, the process should stop when the set of changes is produced. The changes must be manually applied to the target publication using the Figures Tool.

In comparison to the SEI-BUD 4 versions prior to 4.6, the stages of sending the chosen set of amendments to the Repository Manager, the integration of the figures and the generation of the set of changes has been completely rewritten. The application of the set of changes has been improved to hide some implementation detail. The automated process in itself has been introduced in version 4.7. The only thing that has stayed stable is the input format to make mass updates to budget volume: the meta-transactions formalism.

5.9.2. Selecting the amendments and solving the conflicts

A dedicated applet allows memorising the selected amendments. When the user has finished choosing the amendments, he pushes the “Send to integration” button to start the real integration process.

5.9.2.1. Definition of a conflict

There is a conflict when two or more different amendments modify the comments of the same section inside the Budget.

The figures are never in conflict because whenever there is more than one amendment on the same budget line, the integration adds up the differences.

Inside the comments, it does not matter where the change happens. For instance, there is already a conflict when one amendment modifies the remarks and another the legal basis.

5.9.2.2. Conflict detection and resolution

The Web-based interface scans all the transactions of the selected amendments. It orders them by transactions modifying the figures and transactions modifying the comments. For all sections containing a text change, it counts how many transactions change it. There is a conflict when this number is larger than one.

The sections containing a conflict are ordered by volume and alias. For each section the Web based interface displays a list of radio buttons that allows the user to choose a text version among all the amendments. One more choice excludes the text modification from the integration process.

5.9.2.3. Choosing the target of the integration

On the same conflict resolution screen, the user specifies which kind of transaction he wants to perform:

only integrate the figures,

only integrate the changes to the comments,

only integrate the changes that modify the structure of the Budget (add a section, delete a section, merge 2 sections, split a section),

integrate on a combination of the previous choices.

If the user wants to automatically apply the changes to a publication, he must supply the source publication and the target publication. The source publication must exist. The target publication must not exist.

5.9.2.4. Sending the request to the Repository Manager

The last action of the Web-based interface is to collect all the transactions, whether they are part of a conflict or not, to pack each transaction in an`SAmdIntegrationTransactionInfo`Xooof structure and to call the`AmdIntegrationManager.create`service of the Repository Manager.

All this is done in the method`sendIntegrationSend`of the`services.FrameService`Java servlet.

5.9.3. Integrating the figures

The integration of the figures is by far the most complex stage. It is completely executed by the Repository Manager.

Before the actual integration of the figures starts, all the transactions that come from the Web-based interface are stored in the management database in the table`INTEGRATIONTRANSACTIONS`.

The result of this stage is a set of integrated figures stored in the management database in the table`VERSIONS_FIGURES_INTEGRATION`.

There are three different kinds of figures to integrate:

usual figures coming from the amendments,

figures coming from the compensation transactions (the compensation transactions are automatically produced by the system to balance out the administrative expenditures inside the volume 4 because

the XX budget lines are an aggregation of all the administrative expenditures and the Parliament may amend the XX lines as well as the administrative expenditures inside the financial perspectives),

corrected figures of the XX budget lines coming from the Commission and BadgeBud.

The different kinds of figures are handled in sequence. The usual figures are computed first. The compensation transactions are deduced from the usual figures and finally the corrected figures may supersede the computed ones.

5.9.3.1. Integration of figures coming from the amendments

This is the usual case. All the figures transactions are ordered by volume and budget line. Every transaction contains an increase or decrease of money. The increases or decreases are simply summed up. The total increase or decrease is added to the*base amount*.

For each reading and kind of amendment, the following table shows where base amounts come from:

Lecture		**Base amount**
Council, 1st reading		preliminary draft budget
Parliament, 1st reading	before August, 15th	preliminary draft budget
	after August, 15th	draft budget
Council, 2nd reading		draft budget
Parliament 2nd reading		Council position on 2nd reading

All these computations are handled by the`AmdIntegrationManager`Java bean.

5.9.3.2. Computation of the compensation transactions

In this stage, all the integrated figures found in the`VERSIONS_FIGURES_INTEGRATION`table are scanned again. If they belong to the volume 4 of the Budget, their budget lines are looked for in the`COEFFICIENTS_INTEGRATION`table of the management database. The`ID_NODE_INTEGRATION_FROM`and`TP_BUDGET`fields are used as keys.

If at least one entry is found, the associated`ID_NODE_INTEGRATION_TO`and`CF`fields are retrieved. They generate a new entry in the`VERSIONS_FIGURES_INTEGRATION`table. The newly created records all have “1” in the`BL_COMPENSATED`field.

The compensations are computed in the`AmdTransactionsCompensationManager`Java bean.

5.9.3.3. Generating the corrected figures as computed by the Commission and BadgeBud

A last time, all the integrated figures found in the`VERSIONS_FIGURES_INTEGRATION`table are scanned. If they belong to the volume 4 of the Budget, their alias is looked for in the`CORRECTIONS_XX`table of the management database. The`ID_PHASE`and`LB_ALIAS`fields are used as keys.

If at least one entry is found, the associated amounts are retrieved. They generate a new entry in the`VERSIONS_FIGURES_INTEGRATION`table. The newly created records all have “`CORRECTED`” in the`TP_VERSION`field. Such a row has a higher priority than the other rows.

This processing is executed in the`AmdIntegrationManager`and`AmdCorrectionsXX`Java beans.

5.9.4. Generating the set of changes

The aim of this stage is to produce one ore more instances of the MetaTrans DTD. As a matter of fact, the SEI-BUD system generates 1 instance per amended volume.

The actual generation takes place in the Java class called`com.softwareag.belgium.seibud.amdbuilder.MetaTrans.MetaTransBuilder`.

First, all the transactions found in the`INTEGRATIONTRANSACTIONS`table are grouped by volume and ordered by alias. If the transaction is a figures transaction, the actual figures are read from the`VERSIONS_FIGURES_INTEGRATION`table. Then, all the transactions are output to a temporary file according to their type.

The generation follows simple rules:

A delete transaction becomes a MetaTrans`delete`element.

An add transaction becomes a MetaTrans`insert`element. If some transactions depend on a previous add transaction to find its right position, it is integrated inside the`insert`element it refers to.

A figures transaction becomes a MetaTrans`update`element modifying the ABB element`bud-data`.

A content modification transaction is split into several MetaTrans`update`or`delete`elements. The splitting is essential in order to preserve the structure of the target document. The target document is an XML instance. It may contain elements that in turn contain other elements. The structure of a budget document is very rigid. Either an element is a leaf element containing only text and/or figures or it is an intermediate node containing figures, texts and other elements. If the system generated one`update`element per content modification transaction, it would have to handle the leaf and intermediate nodes differently because a content modification transaction only modifies the text and not the children elements. For an intermediate node, you would not only have to supply the modified text, but also all the contained elements. If you did not provide them, you would loose all the contained elements at the loading stage! The only solution to this problem is to generate one MetaTrans`update`element per ABB`bud-heading`,`bud-text`,`bud-remarks`,`bud-legal`or`bud-reference`element. The choice between a MetaTrans`update`or`delete`element is dictated by the content of the ABB element. If the element is empty, a MetaTrans`delete`element is generated; otherwise a MetaTrans`update`element is generated.

In any case, the set of meta-transactions about a specific volume are saved in a file. The name of the file starts with "`mtt_`" and is followed by the name of the volume to which the meta-transactions belong. The file is stored in the`$RUN/tmp`directory where`$RUN`stands for the directory path where all the files of the SEI-BUD system are located.

The reasons for generating a MetaTrans instance instead of an Xupd instance are as follows:

The MetaTrans instance is a lot more compact than the equivalent Xupd instance. This becomes important for the volumes 1 and 4 of the budget where the instances are several megabytes large!

The Figures Tool already accepts MetaTrans instances as input. This allows using the same file for an automatic update or for a manual update.

The MetaTrans instance postpones the solving of several details to a further stage. These details would be very hard to solve at this stage but they are resolved in a more general way later on when the set of meta-transactions is applied to a target publication. The details are:

Deletion of an non-existing element, this happens for instance quite often with the `bud-legal` or `bud-reference` elements;

Update of an element that does not exist yet

Localisation of an element when its siblings are optional

5.9.5. Building a new target publication

The Java bean AmdIntegrationManager also orchestrates the building of a new target publication.

The user having specified a name for the source publication and a name for the target publication, their syntax is controlled first.

The existence of the source and target publications is then checked. If a source publication does not exist, the integration process cannot proceed. If a target publication already exists, it is deleted. All the volumes belonging to a publication are deleted.

At this point, a new target publication is created by simply copying all the volumes of the source publication.

The set of meta-transactions are now ready to be applied. They are only applied if the volume to which they belong effectively exists.

5.9.6. Applying the set of meta-transactions

The final stage is the conversion of the meta-transactions to Xupd transactions and the update of the target publication. The XSLT stylesheet meta2xupd.xslt does most of the work. Some postponed implementation details are solved here in a very elegant way:

A MetaTrans delete transaction always generates a Xupd delete transaction with the option to ignore the transaction if the element localised by the XPath does not exist.

All the MetaTrans update transactions are converted to an Xupd delete transaction (with ignore error if element not found) followed by an Xupd insert transaction.

To localise correctly the insertion point of ABB `bud-reference`, `bud-legal` and `bud-remarks` and `bud-intro` elements, they are grouped by father element, sorted in the reverse order given by the ABB DTD and always inserted after the `bud-data` element. All this happens in a dedicated XSLT stylesheet called `sort-xupd.xslt`.

The resulting Xupd transactions are actually submitted to the XML repository to update target publication.

The XEF framework is responsible for carrying out all this work.

6. SEI-BUD SYSTEM CLIENT TOOLS

6.1. Control tool

6.1.1. Principles

The control tool is a Java application that allows communication with the Repository via an HTTP connection. All users have a control tool on their computer and can communicate with the SEI-BUD server at any time.

The control tool is thus an "individual" client system application. The control tool is the only interface the user has with the SEI-BUD server. It is therefore through the control tool that the user organises his request/response dialogue with the SEI-BUD Repository.

From a design point of view, the client/server architecture of the SEI-BUD system is of the following type:

data on the server,

sharing of logic between the server and the clients,

presentation of data assured on client.

More specifically, the client/server division is as follows:

on the server side there are:

the data (SEI-BUD XML Repository)

part of the system logic (Repository Manager seen by the XmlDispatcher façade, filters, delta, etc.)

on the client side there are:

another part of the system logic (modification of figures, structure, content, etc.)

the presentation of information (Author Word editing tool, figures tool, etc.)

The client/server dialogue, which is performed by means of the exchange of XML messages, is organised:

by the RMG (Repository Manager) on the server side

by the control tool on the client side

From an operational point of view, and disregarding the role of the user and the details of the production environment, the functions of the control tool are generic. These functions are as follows:

to download a nomenclature (current nomenclature) for a given volume and language. Nomenclatures are published in XML on the server and are accessible using a URL.

on the basis of a selection made from this "current" nomenclature, the user is able to send to the Repository:

a request to consult an object for editing (in response to this request, the server sends the object to be edited in the format requested, or an error message)

a request to reserve an object for editing (in response to this request, the server sends the object for editing in the format requested, or an error message), in order to make changes using a client application (Author Word editing tool, figures tool, etc.) and for subsequent updating of the Repository

a request to print the differences in an object for editing.

an end-of-phase order that moves this object for editing on to the next phase in its workflow. This request is the subject of a special privilege.

an order to create a translation request, which sends this object for editing to the Translation Service (Poetry): consultation in the author language, reservation of other languages and printing of differences. This request is the subject of a special privilege.

A query on the translation being performed (with Poetry), using the Translation Followup GUI. This request is the subject of a special privilege.

depending on the objects for editing found on the user tool, the user can send to the Repository:

a request to update an object for editing

a request to cancel a reservation of an object for editing

the creation of a new publication (special privilege)

the destruction of a publication (special privilege)

The profile (Author, Translator, etc.) of the user determines the type of object for editing (Structure, Headings, Remarks, Figures) and the format (RTF, XML, CSV, SGML, etc.) that the user is allowed to handle. The group (DG19, Council, etc.) to which the user belongs determines access to publications. For example, the group DG19 has access to Volume 4.

Groups and profiles are defined in the user database on the server. The control tool receives from the server the list of operations permitted to the user who launched the job session.

The control tool therefore only shows the types of object for editing and the formats to which the user has access.

For example, for consultations and reservations, according to a selection made from the current nomenclature, an Author (PDB, DB or B Authoring phase) is allowed to:

make a request to consult or reserve the Remarks in a section of the Budget (object for editing), in RTF format for modification using the Word Authoring tool, or HTML format (consultation only in HTML format).

make a request to consult or reserve the Figures in any section of the Budget (object for editing) in XML format, loading these into the figures tool for consultation or modification.

make a request to consult or reserve the Headings of a section of the Budget (object for editing) in CSV format for modification in the appropriate tool.

make a request to consult or reserve the Structure of a section of the Budget (object for editing) in XML format, loading this into the structure tool for consultation or modification;

Any updating attempt that is incompatible or prohibited is blocked within the server, which draws attention to the problem by sending an error message. For example:

any attempt to reserve an object for editing that is already reserved will be rejected by the system and this will be indicated by the dispatch of an error message to the user.

Certain functions of the control tool do not allow any choice regarding consultation/reservation, for example:

there is no point in specifying that you want to consult or reserve a printout of differences (the result of a request to print differences is a file in HTML format for printing);

consultation and reservation are meaningless when it comes to registering an end-of-phase for a particular alias.

6.1.1.1. SEI-BUD system views

In the SEI-BUD system, when you want to obtain data, you must specify a "view" in your request parameters. A view is defined by the data it allows you to modify. The list of available views varies from one publication to another; it also varies from one user to another, in line with the permissions he has been granted.

The table below presents the principal views available:

View	Principal characteristics
nomenclature	This view allows the structure of a document fragment to be modified: budget (or total) lines can be created, merged, deleted, split and renamed. It is accessible to the profile*AuthorPlus*. Changes made to the modified version are applied to all versions.
figures	This view allows the figures in a document fragment to be modified: budgetary figures, schedules, staff and relationships. It is accessible to the profile*AuthorPlus*. Changes made to the modified version are applied to all versions. Changes made to the text of schedule notes must be translated.
headings	This view allows the headings of a document fragment to be modified. It is accessible to the profile*AuthorPlus*. Changes made to the modified version are applied to all versions. They must be translated.
remarks	This view allows the objectives, remarks, legal basis and references of a document fragment to be modified. It is accessible to the profiles*Author, AuthorPlus* and *Master Corrector*. Changes made to the modified version are applied to all versions. They must be translated.
remarks (transl)	This view allows all the text in a document fragment to be modified: headings, objectives, remarks, legal basis, references and schedule notes. It is accessible to the profiles*Translator* and *Corrector*. Changes made to the modified version have no impact on the other versions.
remarks transl (-notes)	This view allows all the text (except the schedule notes) in a document fragment to be modified: headings, objectives, remarks, legal basis and references. It is accessible to the profiles*Translator* and *Corrector*. Changes made to the modified version have no impact on the other versions.
global	This view allows all the data in a document fragment to be modified. It is accessible to the SuperUsers and users responsible for editing Volume 0-EPRD. The changes made to the modified version depend on the role of the user.

6.1.1.2. SEI-BUD system formats

Each format proposed corresponds to a SEI-BUD editing tool:

The format	corresponds to the editing tool
Transactions (.xml)	**MrgTool**: SEI-BUD integration tool.
StrTool2 (xml)	**StrTool**: SEI-BUD Structure editing tool.
FigTool2 (xml)	**FigTool**: SEI-BUD Figures editing tool.
Word97 (.abb)	**AutTool**: SEI-BUD Author Word editing tool.
Word97 (.Vol0)	**AutTool**: SEI-BUD Author Word editing tool.
Word cxt (.rtf)	**AutTool**: SEI-BUD Author Word editing tool.
Word97 (.rtf)	**Word**: Standard Word.
Adept8.1 (.xml)	**CorTool**: SEI-BUD Adept Corrector editing tool.

6.1.2. Data exchange with the server

6.1.2.1. Principles of communication

Client applications such as:

the figures tool

the structure tool

the Author Word editing tool

the Translator Word editing tool

are all standalone applications. None of these applications is directly linked with the server system.

It is the SEI-BUD control tool that communicates requests to the server and receives responses from it.

Communication by HTTP using a client "XMLDispatcher"; this tool is part of the SAG XOoof framework and allows a "business object" method to be activated in a way that is transparent to the programmer, hiding from him the following intermediate details:

the XMLDispatcher serialises a Java data structure representing the input parameter of the request in XML and packages it in an HTTP request;

the HTTP request arrives at a server (JBoss) in a Servlet;

the Servlet then passes that request to the XMLDispatcher, which makes it follow the corresponding business object by deserialising the parameter and calling the method invoked;

the reverse path is followed for the request return parameter.

The description of communications between the control tool and the server is explained in more detail in the section about the Repository Manager sub-system.

All client applications (Author Word editing tool, figures tool, etc.) share directories with the control tool. The access paths for these shared directories are specified in the configuration files for these tools (".ini" file).

More specifically, the structure of the directories is as follows:

`classes`: Java classes common to all tools

`CtrTool`: control tool

`StrTool`: Structure tool

`FigTool`: Figures tool

`WordAutAbb`: Word authoring tool for ABB publication

`WordAutVl0`: Word authoring tool for Volume 0 publication

`files`: directories of objects for editing, one directory per view:

`com`: remarks

`data`: budgetary figures

`diff`: differentials (change-marked texts) for printing

`global`: fragments in global view (repository format)

`heading`: headings of budget items

`nmc`: publication nomenclature

`text`: documents in text view

`text+note+heading`: documents in text + note + heading view

`text-note+heading`: documents in text + heading view

The control tool supplies these directories with files (objects for editing) to be displayed or modified by client applications. Client applications supply these directories with modified files (objects for editing) to be updated on the server system using the control tool.

Requests and request parameters

The different requests sent to the server and their input and output parameters are described in the section "Business object forms" in this manual.

6.1.2.1.2. Data exchanged between the SEI-BUD control tool and an SEI-BUD client application

If the response to a request sent to the server contains the object for editing that was asked for, the control tool copies the attached file (in the expected format) to the shared directory corresponding to its view.

This is what happens when:

a consultation is accepted. The control tool creates a file c![alias]![language]![sequence_number].[format] in the directory shared with the client application.

a reservation is accepted. The control tool creates a file r![alias]![language]![sequence_number].[format] in the directory shared with the client application.

Generally, whatever the client application, the exchange file contains various general information concerning the reserved or consulted object for editing (alias, consultation/reservation date, user who made the consultation/reservation, etc.). We will call this general information metadata.

The files (objects for editing) received from the control tool are then displayed and/or modified by the user using the appropriate client application (which offers the user the files for the consultations/reservations carried out by displaying the metadata).

The changes made using the appropriate client application are saved keeping the original filename. The only exception to this is when the structure tool creates a second file (a "transaction" file) with the same name but a different extension: ".mtt".

Updates to the object for editing are made by selecting the object from the list of requests and clicking on the update button.

6.1.2.2. An example of the Figures "chain"

The example shown below is based on the use of the figures tool. The figures tool is one of the links in the Figures "chain".

A request to consult budgetary figures proceeds in the following way:

depending on the nomenclature, request for consultation via the control tool

transmission of the request to the server by HTTP

acceptance of the request by the server, creation of a "Processing Report" object, transfer of the request to an asynchronous form and return of an acknowledgement

reception of the acknowledgement by the control tool and display of the status "being processed" for the request in question

asynchronous processing of the request on the server

extraction of figures from the XML database

conversion of data from "repo" format to the format expected by the figures tool

enrichment of the "figures" format with the metadata

attachment of the result to the "processing report" and change in status of this report to "accepted" or "rejected"

polling of the status by the control tool and detection of the change of status;

repatriation of the response by the control tool

extraction of the figures file from the response message by the control tool and saving of the file in the directory shared with the figures tool (../files/data)

display of the budgetary figures via the figures tool.

6.1.2.3. TimeZone

Under Unix, it is very important that the environment variable TZ is correctly initialised. If it is not, the times displayed by the control tool will be incorrect. Indeed the Java Virtual Machine uses the notion of time zone in its handling of time.

For example, in Europe (Paris time), the value ECT (European Central Time) must be given. So:

export TZ=ECT

6.1.2.4. Metadata file format

Every object for editing that resides on a client workstation is associated with a metadata file. This metadata file is created by the control tool when it receives an object for reservation or consultation from the server. This file has the same name as the file containing the object for editing, but it ends with the extension ".inf".

This metadata file is in XML format; an example of a file corresponding to a reservation of budgetary figures for title 7 is given below:

```
<?xml version="1.0" encoding="utf-8"?>

<sections>

<section id="repo">

<item id="publication">ABB</item>

<item id="year">2004</item>

<item id="step">AP</item>

<item id="part">ABBAP2004VOL4</item>

<item id="alias">07</item>

<item id="view">Chiffres</item>

<item id="format-name">xml-data-abb</item>

<item id="language">en</item>

<item id="action">Réserver</item>

<item id="date">Tue Jul 08 11:48:02 GMT+02:00 2003</item>

<item id="user">pc</item>

<item id= »role »>auteur</item>

</section>

</sections>
```

The metadata are as follows:

"**publication**" is the name of the publication;

"**year**" is the budget year;

"**step**" is the phase in the preparation of the budget (here, "AP" for "Avant Project" (or "Preliminary Draft")); if the creation of the publication is not broken up into steps, this item of data may be empty;

"**part**" gives the full identifier of the logical publication (in the example above, the name of the "ABB" publication is followed by the phase "AP" followed by the year "2004" and the volume identifier ("VOL4");

"**alias**" is the identifier known by users (here, "07" for title 7);

"**view**" is the view used for reservation or consultation;

"**formatname**" is the name of the format (here, "xml-data-abb" for figures from the ABB publication)

"**language**" gives the language of the object for editing

"**action**" indicates the action carried out by the user to obtain the object for editing (reservation or consultation)

"**date**" gives the date when the object for editing was obtained

"**user**" is the name of the user who obtained the object for editing

"**role**" is the role of the user when the object for editing was obtained.

6.1.3. Technical description of the control tool

6.1.3.1. Initialisation of the control tool

The control tool is launched by calling the static method `main` of the class `CtrToolApp`. This method sets the "look and feel" of the user interface and creates an instance of the class `CtrToolApp`.

The constructor of the class `CtrToolApp` creates the different graphic components (by creating a `MainFrame` class). Next, the dimension of the main window is set and this main window is made visible. Control then returns to the graphic system (Swing, as it happens) and is based on a response to user events.

The class `Globals` gives a large amount of static data and a set of utility methods. It is this class that, for example, registers a reference (`mainFrame`) to the main programme window.

Most of the initialisation of the control tool takes place in the method `jbInit` of the class `MainFrame`:

display of a progress bar

initialisation of a log file

reading of parameters from the file `CtrTool.ini`

selection of a server if no current server is given

obtaining of the username and password

creation of a connection with the SEI-BUD server

construction of the components of the user interface

reading of objects for editing in relation to the selected server, for the current user and residing on the workstation (these objects are stored in an XML file)

6.1.3.2. Interactions with the server

The tables below show the different requests made by the client workstation for a number of typical operations.

Initialisation

Client class	Client method	Server object	Server method
MainFrame	checkApplicationVersion	Server	getVersionFile
CtrlPwdDlg	checkPassword	Server	login
CtrlPwdDlg	checkPassword	Server	enableCompression
CtrlPwdDlg	checkPassword	Server	getDefaultPermissions
UsrInfo	selectDocs	AbbDocument	select
UsrInfo	selectDocs	TranslationMemoryManagerBO	getAvailableFormats
UsrInfo	selectDocuments	AbbDocument	select

Selection of a publication

Client class	Client method	Server object	Server method
PanelNomenc	DocChanged	AbbDocument	getDocumentAccessDescriptor
PanelNomenc	DocChanged	AbbDocument	getAvailableLanguages
PanelNomenc	DocChanged	AbbDocument	listBranches
PanelNomenc	populate	AbbDocument	getToc
PanelNomenc	getDocumentAccessDescriptor	AbbDocument	getDocumentAccessDescriptor

Selection of an entry in the nomenclature

Client class	Client method	Server object	Server method
PanelNomenc	getDocumentAccessDescriptor	AbbDocument	getDocumentAccessDescriptor
DocSelect	deletePublication	AbbDocument	getDocumentAccessDescriptor
DocSelect	checkDoc	AbbDocument	checkDocument_async
DocSelect	addLang	AbbDocument	getAvailableLanguages
DocSelect	addLang	AbbDocument	addLanguage_async
DocSelect	getRootXPathDocument	AbbDocument	getAvailableLanguages
DocSelect	getRootXPathDocument	AbbDocument	getCurrentRootToc
DocSelect	attachVersionLabel	AbbDocument	attachVersionLabel

Reservation of remarks in author mode of a nomenclature element

Client class	Client method	Server object	Server method
PanelNomenc	getConcurrentLockInfo	AbbDocument	getConflictingLocks
PanelNomenc	reservDoc	AbbDocument	checkout_authoring_async
ObjList	syncWithServer	ProcessingReport	search_anyState
ObjList	processFoundItem	ProcessingReport	get
ObjList	processFoundItem	ProcessingReport	delete

Update of a reservation

Client class	Client method	Server object	Server method
PanelObjEd	buttonMajClicked	AbbDocument	checkin_async
ObjList	syncWithServer	ProcessingReport	search_anyState
ObjList	processFoundItem	ProcessingReport	get
ObjList	processFoundValidation	ProcessingReport	get
ObjList	processFoundItem	ProcessingReport	delete
ObjList	processFoundValidation	ProcessingReport	delete

AMD Load

Client class	Client method	Server object	Server method

AMDLoad	unloadLoadExport	AmdVolume	loadUnloadExportToc_async

Display history/follow-up

Client class	Client method	Server object	Server method
HistoryDlg	buttonGetUpdatedFileClicked	Followup	getUpdateData
HistoryDlg	buttonPrintErrorClicked	Followup	getErrorData

Follow-up state information

Client class	Client method	Server object	Server method
StateDlg	buttonRefreshClicked	AbbDocument	getAvailableLanguages
StateDlg	buttonRefreshClicked	AbbDocument	getTocWithState
StateDlg	buttonHistoryClicked	Followup	getFollowupCriteriaDescriptor
StateDlg	buttonHistoryClicked	Followup	get
StateDlg	buttonHistoryClicked	Followup	select

Generate translation quality check

Client class	Client method	Server object	Server method
PanelNomenc	generateTranslQualitCheck	AbbDocument	getAvailableLanguages
PanelNomenc	generateTranslQualitCheck	TranslQualityCheck	getTMXConfig
PanelNomenc	generateTranslQualitCheck	AmdUser	getIdFromLabel
PanelNomenc	generateTranslQualitCheck	ReportManager	translationQualityCheckRequestAsync

Send Poetry

Client class	Client method	Server object	Server method
PanelNomenc	buttonSendPoetryClicked	AbbDocument	getVersionLabels
PanelNomenc	buttonSendPoetryClicked	TranslationServicePoetryUserDemand	getOrigineService
PanelNomenc	buttonSendPoetryClicked	AbbDocument	getGroupLanguageConfig
PanelNomenc	buttonSendPoetryClicked	SeibudTranslationUserDemand	createAndGenerateRequestsAsynch

Translation Request

Client class	Client method	Server object	Server method
SeibudRequestsImpl	getSourceLanguagesForCurrentFragment	AbbDocument	getORILanguages
SeibudRequestsImpl	checkPoetryRequestVersionNumber	SeibudTranslationUserDemand	checkRequestVersionNumber
SeibudRequestsImpl	checkPoetrySendRequest	TranslationServicePoetryUserDemand	checkTranslationServicePoetryUserDemand

Translation Follow-up

Client class	Client method	Server object	Server method
SeibudTrFollowupRequestsImpl	getSeibudDemandsForQuery	SeibudTranslationUserDemand	query
SeibudTrFollowupRequestsImpl	getDemand	SeibudTranslationUserDemand	getInfo
SeibudTrFollowupRequestsImpl	getRequest	PoetryTranslationLVRequest	getInfo
SeibudTrFollowupRequestsImpl	getRequest	PoetryRevisionLVRequest	getInfo
SeibudTrFollowupRequestsImpl	deleteDemand	SeibudTranslationUserDemand	delete
SeibudTrFollowupRequestsImpl	closeDemand	SeibudTranslationUserDemand	close
SeibudTrFollowupRequestsImpl	getDocumentContent	PoetryDocument	getInfo
SeibudTrFollowupRequestsImpl	readDocumentContent	PoetryDocument	getInfo
SeibudTrFollowupRequestsImpl	consultDocumentContent	PoetryDocument	readOnlyAsync
SeibudTrFollowupRequestsImpl	editDocumentContent	PoetryDocument	readOnlyAsync
SeibudTrFollowupRequestsImpl	openDocument	PoetryDocument	getInfo
SeibudTrFollowupRequestsImpl	editDocument	PoetryDocument	readOnlyAsync
SeibudTrFollowupRequestsImpl	getReceivedDocuments	PoetryDocument	getDocuments
SeibudTrFollowupRequestsImpl	getFollowUpMessage	ProcessingReport	get
SeibudTrFollowupRequestsImpl	getRequests	SeibudTranslationUserDemand	query
SeibudTrFollowupRequestsImpl	getRequestDetails	SeibudTranslationUserDemand	getInfo
SeibudTrFollowupRequestsImpl	getLanguageGroups	AbbDocument	getGroupLanguageConfig

SeibudTrFollowupRequestsImpl	getErrorDetail	Followup	getErrorData
SeibudTrFollowupRequestsImpl	getDocuments	PoetryDocument	query

Versions of a publication

Client class	Client method	Server object	Server method
VersionDlg	radioButtonByLabelClicked	AbbDocument	getVersionLabels
VersionDlg	getVersions	AbbDocument	selectVersions
VersionDlg	getLifeCycleStateNames	AbbDocument	getLifeCycleStateNames
VersionDlg	attachVersionLabel	AbbDocument	attachVersionLabel

Operations on a publication

Client class	Client method	Server object	Server method
LockDlg	buttonForceReleaseClicked	AbbDocument	forceRelease
PanelNomenc	getSourceLangList	AbbDocument	getLanguagesParagraphs
PanelNomenc	consultDoc	AbbDocument	getVersionLabels
PanelNomenc	consultDoc	AbbDocument	checkout_readOnly_async
PanelNomenc	commonMerge	AbbDocument	merge_async/mergeReadOnly_async
PanelNomenc	printDiff	AbbDocument	getVersionLabels
PanelNomenc	printDiff	AbbDocument	diff_author_async/diff_corrector_async/diff_translator_async
PanelNomenc	consultTmDoc	AbbDocument	prepareTMDocument
PanelNomenc	buttonAllLocksClicked	AbbDocument	getLocks
PanelNomenc	buttonEndPhaseClicked	AbbDocument	getLocks
PanelNomenc	buttonEndPhaseClicked	AbbDocument	endPhase
PanelNomenc	buttonRetAuthClicked	AbbDocument	startAuthoringPhase
PanelNomenc	buttonCreateBranchClicked	AbbDocument	createBranch
PanelNomenc	createPublication	AbbDocument	copyDocument_async
PanelNomenc	removePublication	AbbDocument	drop
PanelObjEd	updateTm	TranslationMemoryManagerBO	checkin_async
PanelObjEd	update	AbbDocument	checkin_async/checkin_non_blocking_translation_async
PanelObjEd	buttonAnnulClicked	AbbDocument	release_async/release_non_blocking_translation_async
PoetryWSDialogInputOutputParameters	PoetryWSDialogInputOutputParameters	TranslationServicePoetryUserDemand	getConfiguration
SeibudRequestsImpl	getSourceLanguagesForCurrentFragment	AbbDocument	getORILanguages
SeibudRequestsImpl	checkPoetryRequestVersionNumber	SeibudTranslationUserDemand	checkRequestVersionNumber
SeibudRequestsImpl	checkPoetrySendRequest	TranslationServicePoetryUserDemand	checkTranslationServicePoetryUserDemand
SysConsole	executeAction	Server	command

File Transfer

Client class	Client method	Server object	Server method
SysXfer	getFile	Server	getFile
SysXfer	putFile	Server	putFile

6.1.3.3. Management of the table of contents

Before carrying out any operation on a publication, it is necessary to obtain its table of contents. The table of contents of a publication is obtained in a particular language by the method `getToc` of the object `AbbDocument`. This is a synchronous request that returns the data in the form of a XOo°f recursive structure `TocEntry`. The table of contents is a tree structure, each entry of which is described by:

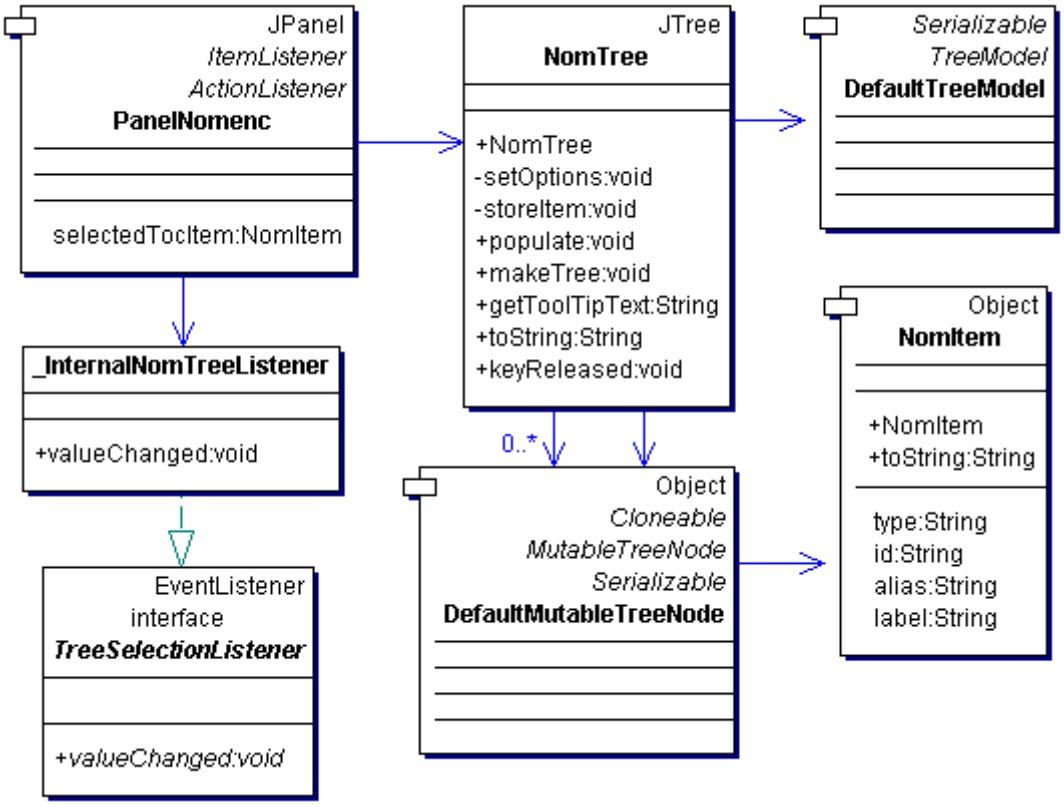
an identifier that will be used to determine the corresponding document fragment; this identifier is a versioned XPath consisting on an XPath, a version number, the language and a branch name (for example: "/abb[1]/nmc_section[1]#1385#en#main")

an alias, which is an identifier in the nomenclature known to users

the heading of the entry in the table of contents

any sub-elements in the table of contents.

The structure `TocEntry` received by the server is then presented using the class instance `NomTree`. A simplified class diagram shows the MVC (Model View Controller) operation used:



The class `PanelNomenc` is the starting point and creates the instance `NomTree` with which a selection model is associated, which implements the interface `TreeSelectionListener`. The data are stored in the tree using `NomItem` objects.

Each change of selection in the nomenclature causes the information regarding this fragment to be refreshed using a `getDocumentAccessDescriptor` request on the server object `AbbDocument`. This request is used to obtain a data structure describing the operations the user is allowed to carry out. The control tool uses this data to activate only the client functions for which the user has the required rights.

6.1.3.4. List of objects for editing

On the workstation, the control tool maintains a list of the objects for editing manipulated by the user; this list appears in the form of a table in the lower part of the main screen of the control tool. This list in fact consists of `ProcessingReport` objects. These objects are stored on the server and the control tool

regularly (every 10 seconds) launches a request to refresh this list. Just to recall, a `ProcessingReport` object is created by the SEI-BUD server for a large number of requests: in the synchronous part of the request the server creates a `ProcessingReport` object and replies to the client with an acknowledgement after posting a message to itself to "continue the job". This `ProcessingReport` is used to represent the state of the object for editing during processing.

So the control tool regularly refreshes the state of `ProcessingReports`. Some of these objects may still be located on the user's workstation. The control tool keeps a permanent list of them, which is saved as an XML document at the end of each session and reloaded at the start of the next session. This XML document is stored in the same directory as the initialisation file "CtrTool.ini" using the syntax [ServerName].[UserName].

The list of `ProcessingReports` is maintained in the class "Global" (`objList` data item). This list is implemented by the class `ObjList` and each of its elements by the class `ObjItem`.

6.1.3.5. Activating the tools using the control tool

The user can select a `ProcessingReport` from the list and carry out one of the accessible operations on the object using the buttons. The user can also "double-click"; this activates the tool so that the object for editing can be displayed or edited. Activation is performed by the Java package "`javax.activation`" (`MailcapCommandMap` class) and uses the configuration file "`mailcap`"; here is an extract giving the default and the entry for the figures tool:

```
seibud/*;; x-java-view=com.softwareag.belgium.seibud.CtrTool2.TextEditor

seibud/*;; x-java-edit=com.softwareag.belgium.seibud.CtrTool2.TextEditor

seibud/xml-data-abb;; x-java-view=com.softwareag.belgium.seibud.CtrTool2.FIGEditor

seibud/xml-data-abb;; x-java-edit=com.softwareag.belgium.seibud.CtrTool2.FIGEditor
```

6.1.3.6. List of control tool classes

All the classes that implement the control tool are part of the several packages. Here is the list of packages and classes in alphabetical order:

`com.softwareag.belgium.seibud.CtrTool2`:

`AbstractTreeTableModel` is an abstract class that implements the interface `TreeTableModel` (this interface extends the interface `TreeModel` of the package `javax.swing.tree`); this class contains a number of facilities for managing trees;

`Connection` is the class that manages requests to the SEI-BUD server;

`CtrlPwdDlg` extends the class `PwDlg` in the SEI-BUD package of common utilities `com.softwareag.belgium.seibud.SeiUtils` and is used to enter the name of the user and his password;

`CtrToolApp` is the main class of the application;

`DynamicTreeTableModel` extends the class `AbstractTreeTableModel`; this class uses the Java "reflection" mechanism to retrieve values;

`FollowDlg` is a dialogue used to fill in the necessary parameters in a request for follow-up information;

`Globals` contains many statistical data as well as several utility methods;

`HistoryDlg` is the dialogue used to display history/follow-up data such as the list of consecutive reservations;

`HTMLViewer` extends the class `ExeLauncher` and is used to launch the display of an object for editing in HTML format (for example a difference printout);

`ImpDiffDlg` is a dialogue that enables the user to enter the various parameters required for a difference printout request;

`InfoDlg` is a dialogue used to display detailed information on an object for editing (for example, on a failed consultation or reservation);

`JTreeTable` extends the class `JTable` of the package `javax.swing` and configures its behaviour, in particular using the sub-class `TreeTableCellRenderer` and the class `TreeTableModelAdapter`; this class is used to display follow-up information in the form of a table;

`LoadDlg` is a dialogue that displays a progress bar during initialisation of the control tool;

`LockDlg` is a dialogue used to display reservations;

`MainFrame` extends the class `JFrame` in the package `javax.swing`; this class manages the main application window; its method `jbInit` performs most of the initialisation of the control tool;

`NomItem` represents a nomenclature element as a tree element (`NomTree` class);

`NomTree` extends the class `JTree` in the package `javax.swing` and is used to represent a nomenclature tree; each element of the tree is represented by an instance of the class `NomItem`.

`ObjItem` represents an object for editing;

`ObjList` represents a list of objects for editing (`ObjItem`);

`OptNetDlg` is a dialogue used to edit the network and server configuration options;

`PanelFollow` inherits from `JPanel` and is used in the follow-up screen to enter the various parameters relating to follow-up;

`PanelNomenc` inherits from `JPanel` and contains the various data and actions associated with a nomenclature;

`PanelObjEd` inherits from `JPanel` and contains the data for different requests initiated by the user (reservations, consultations, updates, etc.);

`PanelPoetry` inherits from `JPanel` and contains the interactive components that enable the parameters for a request for translation by the Poetry system to be entered;

`PoetryDlg` is the dialogue for a Poetry translation request; it uses `PanelPoetry`.

SaxParserObj extends the class DefaultParser of the package org.xml.sax.helpers; this class is used to import the list of the user's objects for editing residing on the workstation and for a particular SEI-BUD server;

SeparatorComboBoxRenderer is a class that extends BasicComboBoxRenderer in order to create a combobox that supports a separator as an element;

StateDataModel extends the class DynamicTreeTableModel and is used by the class JTreeTable to display follow-up information in the form of a table;

StateDlg extends the class JFrame and contains the visual components for displaying the follow-up state information;

StateItem extends the class DefaultMutableTreeNode of the package javax.swing.tree and represents an item of follow-up information;

SwingWorker is derived from the corresponding class Swing and makes it possible to work with several "threads" in the Swing graphics framework;

TableMap extends the class AbstractTableModel and is used by TableSorter;

TableSorter extends the class TableMap and is used in several dialogues to represent a table that can be sorted by columns;

TreeTableModel is an interface that extends the interface TreeModel of the package javax.swing.tree and is used by the class AbstractTreeTableModel;

TreeTableModelAdapter extends the class AbstractTableModel in the package javax.swing.table and is used by the class JTreeTable;

UsrInfo represents the information associated with a user connected to a SEI-BUD server; this class maintains notably a series of information received from the server, in particular the role, privileges and permissions;

com.softwareag.belgium.seibud.CtrTool2.Editors:

ADEPTEditor extends the class ExeLauncher and is used to launch the ADEPT editing tool from an object for editing displayed in the control tool;

FIGEditor extends the class ExeLauncher and is used to activate the figures tool from a corresponding object for editing;

STREditor extends the class ExeLauncher of the package com.softwareag.belgium.seibud.SeiUtils and is used to activate the figures tool to edit or consult a corresponding object for editing;

TextEditor extends the class ExeLauncher in the package com.softwareag.belgium.seibud.SeiUtils and is used to launch a text editor determined by the system on the basis of the file extension of the object for editing;

TRNEditor extends the class ExeLauncher of the package com.softwareag.belgium.seibud.SeiUtils and is used to activate the structure tool from the corresponding object for editing;

WORDABBEditor extends the class ExeLauncher of the package com.softwareag.belgium.seibud.SeiUtils and is used to activate the Word remark editing tool from the corresponding object for editing in the ABB publication;

WordVloEditor extends the class ExeLauncher of the package com.softwareag.belgium.seibud.SeiUtils and is used to activate the Word remark editing tool from the corresponding object for editing in the Volume 0 publication;

com.softwareag.belgium.seibud.CtrTool2.poetry: This package contains all the classes related to the GUI allowing the creation of a Translation Request to Poetry.

CodeValue represents link between code and its value;

DateEditor extends AbstractCellEditor and it is used to display dialog for editing dates;

DateRenderer extends JLabel and it is used to display dates in correct format;

LanguageModel extends the class AbstractTableModel and it is used for language tables in PoetryPanel;

Phase represents phase of publication (Authoring, Translation, Correction, Publishing, Published);

PoetryPanel is panel of PoetryWSDialog and represents poetry related data;

PoetryWSDialog is a dialog for display of send poetry information;

PoetryWSDialogInputOutputParameters represents input and output parameters of PoetryWSDialog;

Request represent request type (Revision, Translation);

SeibudPanel is panel of PoetryWSDialog and represents seibud related data;

SeibudRequests is the interface that declares the methods that the PoetryWSDialog can use to ask for information to Seibud; Using this interface to specify the communication between the interface and the Server ensures that the two are loosely coupled.

SeibudRequestsImpl is the implementation of the SeibudRequests interface;

com.softwareag.belgium.seibud.CtrTool2.trfollowup: This package contains all the necessary classes for the Translation Followup.

Demand is interface that declares the methods that can be used to get information for the user translation demands;

DemandImpl is implementation of the Demand interface;

Document is interface that declares the methods that can be used to get information for the documents;

DocumentImpl is implementation of the Document interface;

FollowupDialogIOParameters represents input and output parameters of PoetryWSDialog;

ListElements defines a list of elements used in Demand and Request interfaces;

ListProperties defines a list of properties used in Demand, Document and Request interfaces;

Request is interface that declares the methods that can be used to get information for the poetry translation requests;

RequestImpl is implementation of the Request interface;

SeibudTrFollowupRequests is the interface that declares the methods that can be used to get for information from Seibud; Using this interface to specify the communication between the interface and the Server ensures that the two are loosely coupled.

SeibudTrFollowupRequestsImpl is the implementation of the SeibudTrFollowupRequests interface;

com.softwareag.belgium.seibud.CtrTool2.trfollowup.controller:

Controller is used to implement all business logic for the translation follow-up information;

com.softwareag.belgium.seibud.CtrTool2.trfollowup.model:

BaseDao is base DAO(Data Access Object) and it is extended by all other DAO objects;

Criteria represents criteria for query poetry translation requests;

CriteriaForDocuments represents criteria for query documents;

LanguageJobDao extends BaseDao and represents LanguageJob data;

LanguageJobDocumentDao extends BaseDao and represents LanguageJobDocument data;

LanguageJobDocumentExtendedDao extends LanguageJobDocumentDao and represents LanguageJobDocumentExtended data;

LanguageJobList extends DefaultTableModel and it is used in RequestDetailDao and ResultsDetails;

RequestDao extends BaseDao and represents Request data;

RequestDetailDao extends BaseDao and represents RequestDetail data;

RequestList extends DefaultTableModel and it is used in RequestsQueriesPanel and ResultsList;

SentDocumentDao extends BaseDao and represents SentDocument data;

SentDocumentsList extends DefaultTableModel and it is used in RequestDetailDao and ResultsDetails;

StandardDateFormater is date formatter;

com.softwareag.belgium.seibud.CtrTool2.trfollowup.view:

DateCellRenderer extends DefaultTableCellRenderer and it is used to set the date format of date cells in a table;

JXFixedSizeDatePicker is GUI extension for display of date fields;

LanguageGroupsModel extends AbstractListModel and is used to present target languages groups in QueryPanel;

LanguageSelectedTableModel extends AbstractTableModel and is used to present source and target languages in QueryPanel;

QueryPanel is a subpanel of the RequestsQueriesPanel panel and represents query related data;

RequestsQueriesPanel is a panel of the TrFollowUpFrame frame and represents query related, result list and result details data;

ResultsDetails is a subpanel of the RequestsQueriesPanel panel and represents result details data;

ResultsList is a subpanel of the RequestsQueriesPanel panel and represents result list data;

TableSizeAdjustor is used to adjust the size of a table;

TrFollowUpFrame is a frame for display of translation follow-up information;

com.softwareag.belgium.seibud.CtrTool2.trfollowup.xslt:

ResultsListXml2HtmFilter extends XSLTFilter and it is used to transform results list from XML format to HTML format;

6.1.3.7. The file CtrTool.ini

This configuration file is in XML format and consists of a series of sections; each section is like a dictionary giving a value for a name. The following is an example of a "CtrTool.ini" file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sections>
<section id="CONFIGS">
<item id="CONFIGLST">prodABB,SEIBUD4I</item>
<item id="CURCONFIG">prodABB</item>
</section>
<section id="prodABB">
<item id="DESC">Serveur SEIBUD4 ABB</item>
<item id="MODE">http</item>
<item id="SERVLET">http://budget.opoce.cec.eu.int/seibud4/servlet </item>
<item id="APP_NAME">seibud/RMGXMLDispatcher</item>
<item id="INITIAL_CONTEXT_FACTORY">org.jnp.interfaces.NamingContextFactory </item>
<item id="PROVIDER_URL">158.167.227.11:1099</item>
<item id="URL_PKG_PREFIXES">org.jboss.naming:org.jnp.interfaces </item>
<item id="HELP">http://No Where1</item> <item id="PROXY_ADR"/>
<item id="PROXY_PRT"/>
</section>
<section id="PATHS">
<item id="FILES">../files</item>
</section>
<section id="LAUNCHER">
<item id="htm-diff-text-abb">C:FilesExplorer.EXE</item>
</section>
<section id="GENERAL">
<item id="REFRESHMSGDELAY">180</item>
</section>
</sections>
```

The section CONFIGS contains the following parameters:

CONFIGLST gives a list of configuration names separated by a comma.

CURCONFIG indicates the name of the current configuration.

For each configuration name mentioned in the CONFIGLST list, there must be a corresponding section with the same name; the section prodABB contains the following parameters:

DESC is the description of the server

MODE is the connection mode of the XMLDispatcher client with the Web server; this connection mode can have the values "http" or "rmi";

SERVLET is the name of the servlet to be contacted at the Web server; this name is used to instantiate an XMLDispatcherHTTPFormClient component; it is only used in the case of an "http" connection;

APP_NAME is the name of the application that will be used to instantiate an XMLDispatcherEJBClient client component by interrogating a JNDI context; this is only used in the case of an "rmi" connection;

INITIAL_CONTEXT_FACTORY, PROVIDER_URL and URL_PKG_PREFIXES are three parameters used to create a JNDI initial context; they are only used in the case of an "rmi" connection;

HELP completes a link in the form of a URL and is supposed to point to the control tool documentation;

PROXY_ADR, PROXY_PRT completes a proxy address and a port number where it is necessary to pass through an http proxy;

The section PATHS can contain a FILES parameter, the value of which is an access path to the directory that will be used to contain any objects for editing received from the server;

The section LAUNCHER is used to complete the entry htm-diff-text-abb, which is meant to determine the access path to an HTML browser (Internet Explorer in the example given);

The section GENERAL gives the interval in seconds between two refreshes of the list of processing reports (ProcessingReport).

The POETRY section can contain the following parameters:

YEAR gives the year specified for the previous Translation Request to POETRY

NUMBER gives the number specified for the previous Translation Request to POETRY

PART gives the part specified for the previous Translation Request to POETRY

VERSION gives the version specified for the previous Translation Request to POETRY

REMARK gives the remark specified for the previous Translation Request to POETRY

CLAIMANT_SERVICE gives the claimant service specified for the previous Translation Request to POETRY

ORIGINE_SERVICE gives the origine service specified for the previous Translation Request to POETRY

CONTACT gives the contact specified for the previous Translation Request to POETRY

RESPONSIBLE gives the responsible specified for the previous Translation Request to POETRY

SECRETARY gives the secretary specified for the previous Translation Request to POETRY

SUPERVISOR gives the supervisor specified for the previous Translation Request to POETRY

6.1.3.8. Java Virtual Machine version and required libraries

This SEI-BUD client tool works with Java Virtual Machine (JVM) version 1.5.

The following table gives the list of standard Java libraries that are used by the control tool.

Library	Version	Description
AbsoluteLayout.jar	5.5	Netbeans Window's layout
activation.jar	1.0.2	Sun library to enable the handling of data encoded with MIME conventions.
commons-collections-3.0.jar	3.0	An extension to the Java Collections Framework
commons-io-1.2.jar	1.2	Utilities to assist with developing IO functionality.
HTTPClient.zip		Classes for HTTP communication between the client and the server
iso-relax.jar		RELAX (Regular Language description for XML) classes
jakarta-regexp-1.2.jar		Regular expression classes
jboss-common-client.jar	JBoss 4.0.4.GA	JBoss common client
jboss-j2ee.jar	JBoss 4.0.4.GA	JBoss J2EE
jcalendar.jar		Java date chooser bean for graphically picking a date
jdom.jar	1.0	JDOM classes
jfcunit.jar	2.08	An extension to the popular testing framework JUnit
jnp-client.jar	2.4.4	JBoss implementation of the client access to the EJBs.
log4j.jar	1.2.14	Logging utilities from the Apache Software Foundation
MDateSelector.jar	1	IIopup calendar to provide date input..
msv.jar		XML Multi Schema Validator classes
qjcc-2006-12-11.jar	2006-12-11	The Quarterback Java Classes Collection
relaxngDatatype.jar		RELAX NG language classes
rmgmsg.jar		RMG messages classes
sagbclassinfo.jar		SAG Business class information classes
SeiUtils.jar		SEIBUD utility classes
serializer.jar	2.7.0	XSLT parser from the Apache Software Foundation
swing-layout-1.0.jar	1.0	Extensions to Swing to create professional cross platform layout
xalan.jar	2.7.0	Xalan classes
xerces.jar		Xerces classes
xercesImpl.jar	JBoss 4.0.4.GA	XML parser from the Apache Software Foundation
xml.jar	1.0.0	Sun parser
xml-apis.jar	1.3.03	XML parser from the Apache Software Foundation
xmldispatcher.jar		SEIBUD Xml dispatcher
xmldispatcher-client.jar		SEIBUD Xml dispatcher client
xmlstruct.jar		SEIBUD Xml structures
xsdlib.jar	1.2.2	Sun Multi-Schema XML Validator
zaxon8.jar	8.0	SAXON XSLT parser

6.2. Structure tool

6.2.1. Principles

The structure tool is one of the SEI-BUD authoring tools. It is used only for ABB publication (not for Volume 0).

It allows the editorial structure (section, part, title, etc.) and aliases (PDB, DB and B) to be displayed and modified.

Changes made to the structure of the author's language version are automatically carried over to other language instances by the server system.

6.2.2. Data exchange

The structure tool is a standalone tool. It is not directly linked with the server system. It is the SEI-BUD control tool that sends requests to the server and receives responses from it.

The control tool and the structure tool have shared directories, the access paths of which are specified in the configuration files for these two programmes (".ini" file). The control tool supplies this directory with files containing the structure to be displayed or modified by the structure tool. The structure tool supplies these directories with files containing the modified structures to be updated on the server system using the control tool.

6.2.2.1. Data exchanged between the SEI-BUD control tool and the SEI-BUD structure tool

If the response to a request sent to the server contains a structure, the control tool removes the SGML envelope from the response message and creates a file in the format expected by the structure tool (XML) in the shared directory.

This is what happens when a structure reservation or consultation is accepted. The control tool then creates the file c![alias]![language]![sequence_number].[format] (for a consultation) of the file r![alias]![language]![sequence_number].[format] (for a reservation) in the directory shared with the structure tool. An appended metadata file is also created by the control tool.

On the basis of the various documents received from the control tool and present in the dedicated directory, the structure tool offers the user the structure reservation files and structure consultation files (by displaying the metadata). The user can then display or modify the structure he has chosen.

When the user saves changes to the structure (made with the structure tool), the new structure is placed in the initial XML file and a transaction file is generated. This transaction file contains a list of all the transactions carried out on the structure since it was reserved. The transaction file takes the same name as the structure file, but with the extension ".mtt".

6.2.3. Functions of the structure tool

It is used to:

display the structure and headings of an object for editing.

modify the structure and aliases of an object for editing.

6.2.3.1. Displaying the structure and headings

The structure and headings are displayed in the form of a tree. For more information concerning the presentation of this tree, please consult the user guide.

6.2.3.2. Modifying a structure

The structure modification operations are:

removal of an item

addition of an item

copying of an item

movement of an item

merge of terminal items

modification of an alias

6.2.4. Data format

The structure of the XML document edited by the structure tool is a subset of ABB grammar. The titles of the different items are given there. The reader should refer to the description of this DTD for more information.

For ABB publication, the following nomenclature elements appear in the structure:`nmc-section`,`nmc-sectpart`,`nmc-revenue`,`nmc-expenditure`,`nmc-title`,`nmc-chapter`,`nmc-article`,`nmc-item`,`nmc-annex`,`nmc-grseq`.

For Volume 0 publication, the following nomenclature elements are manipulated by the structure tool:`volume`,`preamble`,`part`,`annex`,`sect1`to`sect9`.

Two attributes are used for all nomenclature elements: the attribute "`alias`" and the attribute "`id`". The attribute "`alias`" appears in the presentation, at the start of the title of the nomenclature element. The attribute "`id`" is used as an anchor for the generation of meta-transactions (see the section on meta-transactions): this attribute is used to identify the target of transactions.

Concerning the format of metadata, the reader can refer to the corresponding section of the description of the control tool.

6.2.5. Technical description of the structure tool

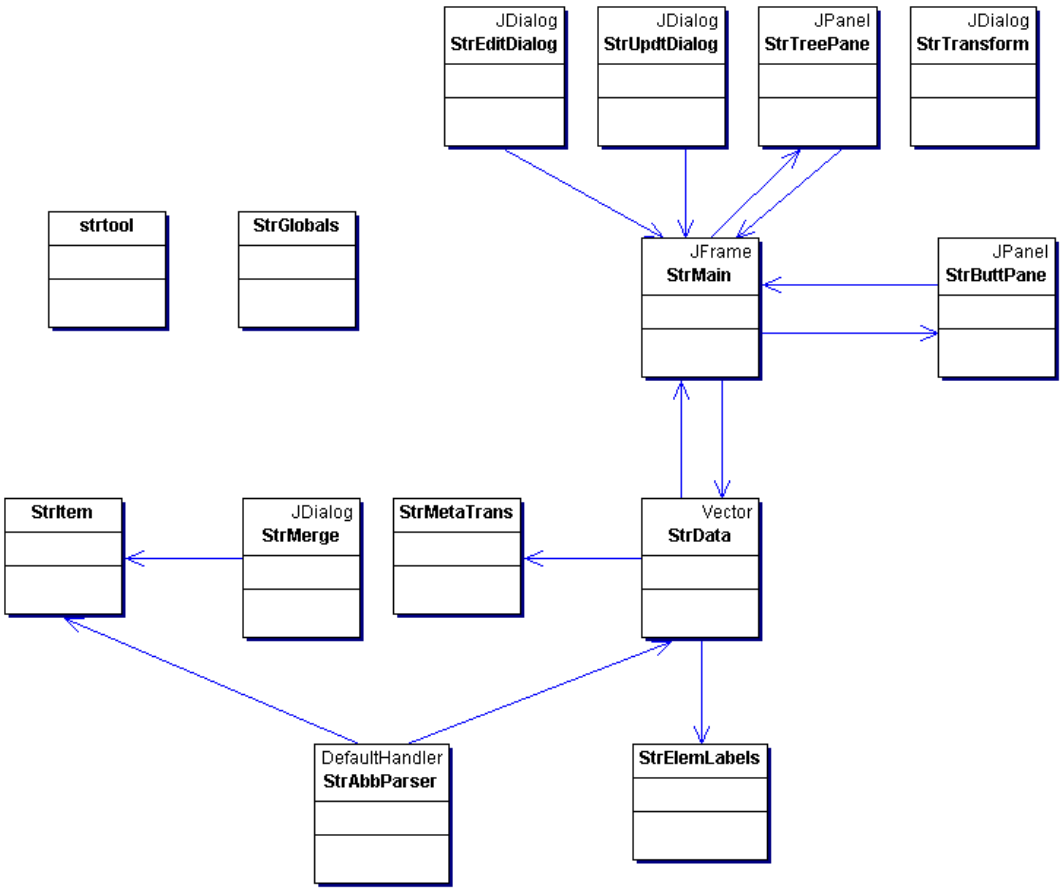
6.2.5.1. Initialisation of the structure tool

The structure tool is launched by calling the static method `main` of the class `strtool`. This method sets the "look and feel" of the user interface and creates an instance of the class `strtool`. The tool accepts an optional argument, which is the name of a metadata file (see the description of the control tool for more information on this metadata file).

The constructor of the class `strtool` creates the different graphic components (by creating a `StrMain` class). Next, the dimension of the main window is set and this main window is made visible. Control then returns to the graphic system (Swing, as it happens) and is based on a response to user events.

The method `jbInit` of the class `StrMain` downloads the nomenclature file, if necessary, and creates the graphic components of the user interface of the structure tool.

6.2.5.2. List of structure tool classes



`StrAbbParser` extends the class `DefaultParser` of the package `org.xml.sax.helpers`; this class is used to read a nomenclature object for editing; it is used by the method `load` of the class `StrData`;

`StrButtPane` extends the class `JPanel` and manages the content of the main window; an instance of this class is created when the application is initialised and this is referenced from the class `StrMain`;

`StrData` extends the class `Vector`; an instance of this class is created when the nomenclature is loaded (method `loadDataFromFile` of `StrMain`); the instance contains the data from the nomenclature and is referenced from `StrMain`;

`StrEditDialog` is the dialog used to capture the parameters needed when a new item is created;

`StrElemLabels` is used to translate element names into headings from parameters given in the configuration file "`StrTool.ini`";

`StrGlobals` contains a series of static methods used in various parts of the application;

`StrItem` implements the representation of a nomenclature element;

`StrMain` extends the class `JFrame` and manages the main application window;

`StrMerge` is the dialogue used to capture the target when nomenclature elements are merged;

`StrMetaTrans` manages the generation of transactions in the format MetaTrans;

`strtool` is the main class of the application; it contains the static method `main`;

`StrTransform` is the dialogue used to capture the destination element of a copy operation;

`StrTreePane` extends `JPanel` and manages the visual representation of the nomenclature;

`StrUpdtDialog` is the dialog used when an alias is changed.

6.2.5.3. The file StrTool.ini

This configuration file is in XML format and consists of a series of sections; each section is like a dictionary giving a value for a name. The following is an example of a "`StrTool.ini`" file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sections>
<section id="main">
<item id="filepath">../../CtrTool2/files/nmc</item>
<item id="help-url">http://..</item>
</section>
<section id="AbbElementLabels">
<item id="enames">nmc-section|nmc-sectpart|nmc-revenue|nmc-expenditure|nmc-title|nmc-
chapter|nmc-article
</item>
<item id="labels">Section|Partie|Recettes|Dépenses|Titre|Chapitre|Article
</item>
</section>
</sections>
```

The section `main` contains the following parameters:

"`filepath`" gives an access path to nomenclature-type objects for editing;

"`help-url`" provides a link (in the form of a URL) to the documentation for the tool.

The section `AbbElementLabels` contains two parameters ("enames" and "labels"), which are two parallel lists separated by vertical bars; these lists constitute a dictionary linking an element name with a heading.

6.2.5.4. Java Virtual Machine version and required libraries

This SEI-BUD client tool works both with Java Virtual Machine (JVM) version 1.4 and JVM version 1.5. Moreover, the migration from JVM version 1.4 to JVM version 1.5 has been successfully performed for the European Parliament setup environment.

The following table gives the list of standard Java libraries that are used by the structure tool.

Library	**Version**	**Description**
xercesImpl.jar	Xerces-J 2.7.1	XML parser from the Apache Software Foundation
xml-apis2_7.jar	Xerces-J 2.7.1	XML parser from the Apache Software Foundation
serializer.jar	2.7.0	XSLT parser from the Apache Software Foundation
msv.jar	1.0	Multi-shema validator library from Sun
iso-relax.jar	2004/11/11	RELAX NG XML validator working with the Multi-shema validator.
xml.jar	0.8	Java XML API by Sun

6.3. Figures tool

6.3.1. Principles

The figures tool is one of a suite of SEI-BUD authoring tools.

It allows the budgetary figures for instances managed by the system (ABB publication) to be displayed and modified.

The budgetary figures are part of the elements considered to be linguistically invariable. Changes to budgetary figures made to the French instance are therefore automatically carried over to the other language instances by the server system.

This function that automatically carries changes to budgetary figures over to all the official languages provides:

a significant improvement in the quality of the publication (synoptic content in all official languages)

a significant improvement in production lead times (changes to budgetary figures do not engender a translation workload)

The figures tool is designed to enable the publication and modification of budgetary figures. The tool also offers:

a budgetary figures export functionality in CSV ("Comma Separated Value") format

a budgetary figures import functionality in CSV ("Comma Separated Value") format

Consequently, it can be used as an interface between the SEI-BUD editing system and an institution's own management system. For example, in the case of the budget, since the publication is the valid version, the following method can be used to update budgetary figures:

reservation of the budgetary figures in a specific publication (publication/part) using the control tool

opening in the figures tool of the object for editing received from the SEI-BUD server

production of a CSV (Comma Separated Value) file using the figures tool export function

use of this CSV file to feed data to the Institution's internal management system

production of a CSV file following modification using the management system

modification of the figures being edited using the figures tool by importing this CSV file

saving of the object for editing

updating of the budgetary figures using the control tool

6.3.2. Data exchanged between the SEI-BUD control tool and the SEI-BUD figures tool

If the response to a request sent to the server contains budgetary figures, the control tool removes the SGML envelope from the response message and creates a file in the format expected by the figures tool (SGML/XML) in the shared directory.

This is what happens when:

a consultation of budgetary figures is accepted. The control tool creates the file c![alias]![language]![sequence_number].[format] in the directory used to hold figures-type objects for editing.

a reservation of budgetary figures is accepted. The control tool creates the file r![alias]![language]![sequence_number].[format] in the directory used to hold figures-type objects for editing.

On the basis of the various documents received from the control tool and present in the dedicated directory, the figures tool offers the user the figures files for reservation or consultation by displaying the metadata. The user can then display or modify the figures file he has chosen.

When the user saves changes to budgetary figures using the figures tool, these changes are saved in a file in XML format using the original file name in the directory used to hold figures-type objects for editing.

On the basis of the files found in this directory, the control tool suggests to the user the reservation files to be sent to the server for the update to take place. It encapsulates the object for editing selected by the user within an update request and sends it to the server.

6.3.3. Functions of the figures tool

It is used to:

display the budgetary figures of an object for editing;

modify the budgetary figures of an object for editing

by encoding

by importing a CSV file

export the budgetary figures in CSV format;

display and modify schedules;

display and modify the relations between budget items;

display and modify human resources.

The figures tool only allows the modification of budgetary figures for "terminal" budget lines (sheets of the budgetary nomenclature). For all budget lines of the type "total" (sheet parent), the tool automatically recalculates the totals.

6.3.3.1. Displaying budgetary figures

After opening a file from the server, the window is split into one, two or three areas (configurable):

one area presents, in a tree structure, the budgetary nomenclature of the object being edited;

the second area presents a grid showing the figures corresponding to the budget lines; this grid contains four items of data for each financial year:

a "differentiated appropriations (da)/non-differentiated appropriations (nda)" column (for expenditures only),

a "compulsory expenditure (ce)/non-compulsory expenditure (nce)" column (for expenditures only),

a "commitments" column,

a "payments" column,

the third area shows in detail the data associated with the nomenclature element selected in the first area.

6.3.3.2. Modifying budgetary figures

The figures tool is used to edit:

figures in budget lines;

the attributes "differentiated/non-differentiated appropriations" or "compulsory/non-compulsory expenditure" (in the case of expenditures);

reserves (for expenditures).

However, it is impossible to modify:

the structure: to do this use the structure tool;

the totals lines: these are calculated automatically by the figures tool

6.3.3.3. CSV import/export

CSV import and export enable the exchange of budgetary figures with other programmes that accept the format (Excel, DBase, FoxPro, etc.). Thus, figures exported from the figures tool can be imported into Excel, a FoxPro database, etc.

In the figures tool, the import is carried out on an object for editing that is already open. During a CSV import, the figures tool checks for the presence in the file of all the aliases of budget lines in the current structure (ignoring the totals lines). If an alias is missing or does not belong in the structure, the tool draws attention to this at the end of the import.

6.3.3.4. Generation/calculation of totals

Totals are calculated automatically by the figures tool every time the figures are modified (opening of a file, CSV import, manual encoding).

Totals have the following structure:

For revenues:

totals for years N, N-1, N-2

For expenditures:

totals of non-differentiated figures for years N, N-1, N-2

totals of differentiated figures for years N, N-1, N-2

totals of differentiated and non-differentiated figures for years N, N-1, N-2

6.3.3.5. Displaying and modifying human resources

Every expenditures title (ABB) can be associated with a human resources table. This table, which is mandatory in the Commission section, is edited in the figures tool.

6.3.3.6. Displaying and modifying the relations between budget items

In the Commission section, each budget line is associated with a list of relations with one or more elements of the structure (element of Title XX or budget line of the traditional nomenclature).

These relations are reused in two cases:

*admin*type relations are used to construct the tables in Title XX;

the aliases mentioned in the*traditional*type relations appear in the column**Traditional nomenclature**in the summary tables of chapters built using an automatic process of the SEI-BUD server.

6.3.4. Format of data from the server

The structure of the XML document edited by the figures tool is a subset of the ABB grammar. The titles of the different items are given there, as are the "bud-data" and "bud-data-extra" elements and their sub-elements. The reader should refer to the description of this DTD for more information.

As regards the format of metadata, the reader can refer to the corresponding section of the description of the control tool.

6.3.5. Structure of the CSV import/export file

CSV (Comma Separated Values) files, used to exchange data between a number of different applications (Excel, DB4, Access, etc.), are defined as follows:

they are text files composed of lines separated by Carriage Returns/Line Feeds;

each line consists of a fixed number of fields. The position of these fields is fixed and pre-defined (non-modifiable);

all fields must be present, even if they are empty;

fields cannot contain Carriage Returns/Line Feeds;

each field is separated from the others by a separator. This separator is defined under Windows (Main Group, Configuration Panel, "International" section, "List separator" parameter). By default, this separator is the semi-colon (;);

each field may be delimited by quote marks ("). This is mandatory if the field includes the list separator.

N.B.: This description is somewhat stricter than the standard CSV definition.

The figures file exported by the figures tool is a file in CSV format. For each budget line of the "figures" object being edited, a "CSV line" will be produced.

6.3.5.1. CSV format for a "BUDGET" type publication

For a "BUDGET" type publication, each "CSV line" contains:

the alias (number of the budget line), and

for each budget year (N, N-1 and N-2):

the wording "differentiated/non-differentiated appropriations" ("da" or "nda")

the wording "compulsory/non-compulsory expenditure" ("ce" or "nce")

the outturn amount if the line is a "revenues" ("REV") line, or the commitment amount if the line is an "expenditure" ("EXP") line

the payment amount if the line is an "expenditure" ("EXP") line

the reserve commitment amount if the line is an "expenditure" ("EXP") line

the destination (alias) of the reserve commitment amount if the line is an "expenditure" ("EXP") line

the reserve payment amount if the line is an "expenditure" ("EXP") line

the destination (alias) of the reserve payment amount if the line is an "expenditure" ("EXP") line

the type of budget line:

"" (empty) for an indeterminate line (parent of a "revenue" line and an "expenditure" line)

"REV" for a "revenue" type line (REVENUE)

"EXP" for an "expenditure" type line (EXPENDITURE)

1	alias	mandatory
2	da/nda year N	if "expenditure" line
3	ce/nce year N	if "expenditure" line
4	outturn/commitment year N	
5	payment year N	if "expenditure" line
6	commitment reserve year N	if "expenditure" line
7	commitment destination year N	if "expenditure" line
8	payment reserve year N	if "expenditure" line
9	payment destination year N	if "expenditure" line
10	da/nda year N-1	if "expenditure" line
11	ce/nce year N-1	if "expenditure" line
12	outturn/commitment year N-1	
13	payment year N-1	if "expenditure" line
14	commitment reserve year N-1	if "expenditure" line
15	commitment destination year N-1	if "expenditure" line
16	payment reserve year N-1	if "expenditure" line
17	payment destination year N-1	if "expenditure" line

18	da/nda year N-2	if "expenditure" line
19	ce/nce year N-2	if "expenditure" line
20	outturn/commitment year N-2	
21	payment year N-2	if "expenditure" line
22	commitment reserve year N-2	if "expenditure" line
23	commitment destination year N-2	if "expenditure" line
24	payment reserve year N-2	if "expenditure" line
25	payment destination year N-2	if "expenditure" line
26	the type of budget line	"" or "REV" or "EXP"

The fields take the following format:

alias, destination: the alias as presented in the publication

da/nda:

"da": differentiated appropriations

"nda": non-differentiated appropriations (default value for expenditures)

"": "undefined" and "revenue" type lines, or by default, "nda" for expenditures

ce/nce:

"ce": compulsory expenditure

"nce": non-compulsory expenditure (default value for expenditures)

"": "total" and "revenue" type lines, or by default "nce" for expenditures

amounts:

"nnn...nnn[.nn]": positive number to 2 decimal places max. (e.g.: "1234", "1234.12")

"-nnn...nn[.nn]": negative number to 2 decimal places max. (e.g.: "-1234", "-1234.12")

"-": 0

"f.t.r.": "for the record"

"": unspecified

6.3.6. Technical description of the figures tool

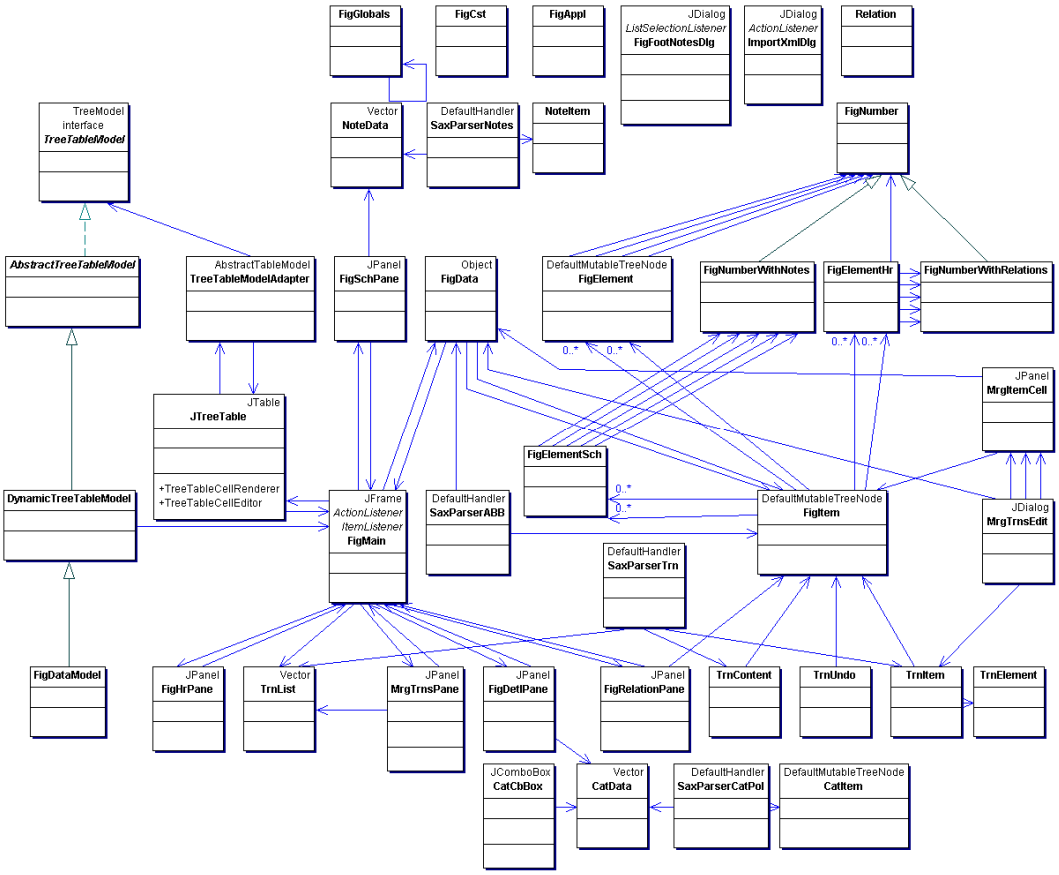
6.3.6.1. Initialisation of the figures tool

The figures tool is launched by calling the static method `main` of the class `FigAppl`. This method sets the "look and feel" of the user interface and creates an instance of the class `FigAppl`. The tool accepts an optional argument, which is the name of a metadata file (see the description of the control tool for more information on this metadata file).

The constructor of the class `FigAppl` creates the different graphic components (by creating a `FigMain` class). Next, the dimension of the main window is set and this main window is made visible. Control then returns to the graphic system (Swing, as it happens) and is based on a response to user events.

The constructor of the class `FigMain` may download a figures file, and creates the graphic components of the user interface of the figures tool.

6.3.6.2. List of figures tool classes



`AbstractTreeTableModel` is an abstract class that implements the interface `TreeTableModel` (this interface extends the interface `TreeModel` of the package `javax.swing.tree`); this class contains a number of facilities for managing trees;

`CatCbBox` extends the class `JComboBox` of the package `javax.swing`; it is used for category/policy selections in the component `FigDetlPane`; this class is used in conjunction with the class `CatData`;

`CatData` extends the class `Vector` and is used to manage the list of different categories/policies; these data are loaded from an XML configuration file "catpol.xml";

`CatItem` extends the class `DefaultMutableTreeNode` and represents a category/policy element;

`DynamicTreeTableModel` extends the class `AbstractTreeTableModel`; this class uses the Java "reflection" mechanism to retrieve values;

`FigAppl` is the main class of the application; it contains the static method `main`;

`FigCst` is a class that groups together the constants common to the whole application;

`FigData` represents the data in a "figures" object for editing; it contains the identification data of the object for editing and the methods for loading and manipulating the object;

`FigDataModel` extends the class `DynamicTreeTableModel` and is used by the class `JTreeTable` to display the figures in the form of a table;

`FigDetlPane` extends the class `JPanel` and manages the detailed display of figures for a budget item; it is referenced from `Figmain`;

`FigElement` extends the class `DefaultMutableTreeNode` and represents the figures for an item and relating to a single financial year; it is referenced by `FigItem`;

`FigElementHr`, like `FigElement`, represents the human resources for an item and single financial year; it is also referenced by `FigItem`;

`FigElementSch` is the same as `FigElement` and `FigElementHdr` but for schedules;

`FigFootNotesDlg` is the dialogue used to insert or edit a note;

`FigGlobals` contains a series of global static data and static methods used globally in the various application classes;

`FigHrPane` extends the class `JPanel` and is used to display the human resources figures;

`FigItem` extends the class `DefaultMutableTreeNode` and manages the figures of a budget item, using in particular, instances of `FigElement`, `FigElementHr` and `FigElementSch`;

`FigMain` extends the class `JFrame` and manages the main application window;

`FigNumber` manages a budgetary figure like a string of characters and contains a series of methods that are used to manipulate a budgetary amount;

`FigNumberWithNotes` extends the class `FigNumber` and is also used to manage notes relating to a budgetary amount;

`FigNumberWithRelations` extends the class `FigNumber` and is also used to manage relations with other budget items;

`FigRelationPane` extends the class `JPanel` and is used to display relations with other budget items;

`FigSchPane` extends the class `JPanel` and is used to display schedules;

`ImportXmlDlg` is the dialogue used to import figures in XML;

`JTreeTable` extends the class `JTable` of the package `javax.swing` and configures its behaviour, in particular using the sub-classes `TreeTableCellRenderer`, `TreeTableCellEditor` and `TreeTableTextField`; this class is used for figures in the form of a table;

`NoteData` extends the class `Vector` and represents a list of notes;

`NoteItem` represents a note;

`Relation` represents a relation between budget items;

`SaxParserABB` extends the class `DefaultHandler` of a SAX parser in order to load the budgetary figures from an ABB object for editing; this class is used by the class `FigData`;

`SaxParserCatPol` extends the class `DefaultHandler` of a SAX parser in order to load the budget category and policy configuration data; this class is used by the class `CatData`;

`SaxParserNotes` extends the class `DefaultHandler` of a SAX parser in order to load the standard ABB notes from an XML file ("`abb-fr.xml`" for notes in French in the ABB publication);

`TreeTableModel` is an interface that extends the interface `TreeModel` of the package `javax.swing.tree` and is used by the class `AbstractTreeTableModel`;

`TreeTableModelAdapter` extends the class `AbstractTableModel` in the package `javax.swing.table` and is used by the class `JtreeTable`.

6.3.6.3. The file "FigTool.ini"

This configuration file is in XML format and consists of a series of sections; each section is like a dictionary giving a value for a name. The following is an example of a "`FigTool.ini`" file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sections>
<section id="fig">
<item id="filepath">../../CtrTool2/files/data</item>
<item id="help-url">http://..</item>
</section>
</sections>
```

The section `fig` contains the following parameters:

"`filepath`" gives an access path to "figures" objects for editing;

"`help-url`" provides a link (in the form of a URL) to the documentation for the tool.

6.3.6.4. The configuration file "catpol.xml"

The categories/policies configuration file is an XML file. In the figures tool, it is used to load a list of predefined choices.

The format of this file can be described by the following DTD fragment:

```
<!ELEMENT CATS (CAT*)>
<!ELEMENT CAT (LABEL*, POL*)>
<!ATTLIST CAT ID CDATA #REQUIRED>
<!ELEMENT LABEL (#PCDATA)>
<!ATTLIST LABEL LG CDATA #REQUIRED>
<!ELEMENT POL (LABEL*)>
<!ATTLIST POL ID CDATA #REQUIRED>
This is an extract of a configuration file:
<?xml version="1.0" encoding="UTF-8"?>
<CATS><CAT ID="0">
<CAT ID="1">
<LABEL LG="FR">Agriculture</LABEL>
 ...
<LABEL LG="SV">Jordbruk</LABEL>
```

```
<POL ID="1.010">
<LABEL LG="FR">Dépenses agricoles (hors ...</LABEL>
<LABEL LG="DA">Landbrugsudgifter ...</LABEL>
</POL>
</CAT>
</CATS>
```

6.3.6.5. Java Virtual Machine version and required libraries

This SEI-BUD client tool works both with Java Virtual Machine (JVM) version 1.4 and JVM version 1.5. Moreover, the migration from JVM version 1.4 to JVM version 1.5 has been successfully performed for the European Parliament setup environment.

The following table gives the list of standard Java libraries that are used by the figures tool.

Library	Version	Description
xercesImpl.jar	Xerces-J 2.7.1	XML parser from the Apache Software Foundation
xml-apis2_7.jar	Xerces-J 2.7.1	XML parser from the Apache Software Foundation
serializer.jar	2.7.0	XSLT parser from the Apache Software Foundation
jdom.jar	1.0	DOM API for Java provided by www.jdom.org
activation.jar	1.0.2	Sun library to enable the handling of data encoded with MIME conventions.

6.4. The Word-based ABB editing tool

6.4.1. Functions of the Word-based ABB editing tool

The types of "text" in a budget document that can be modified using the Author Word editing tool are: headings, objectives, remarks, the legal basis and reference acts. These "text" types do not include notes that go with budgetary figures or schedules. These notes are modified using the Figures editing tool.

The Author Word editing tool is specially designed to display and modify the "text" in a budget document fragment. It is associated with two types of editorial object: remarks and headings.

The benefits of having a specific tool are:

that it makes editing simpler: the author modifies the text (headings or remarks) without having to worry about the figures;

it guarantees, by means of a validation procedure, that changes are correctly integrated in all language versions of the budget document at update.

No constraints are imposed on the author when editing text; there are, however, certain rules to be respected. So if the author removes the words "Legal basis", the legal basis paragraphs will be joined on to the remarks. It should be noted that the author can also see the budgetary figures and modify them in his Word document, but any changes made to the figures are ignored by the server during a remarks editing session.

There are a number of reasons why Word was chosen as the remarks text-editing tool. The main reasons are:

all the authors are familiar with using Word, which was already available on all workstations, so there was no need to purchase an additional tool, nor to learn how to use it;

Word has a large number of additional functions and productivity tools (spell-checker, grammar-checker, thesaurus, macros, and even computer-assisted translation tools).

However, the version of Word currently used (Word 97) does not have an XML editing facility (this facility is expected in Microsoft Office suite version 11). There are quite a few "plugins" and "addons" for Word that will edit XML text, but none of the commercially available solutions are effective or satisfactory. The XML validation of documents edited using Word is therefore performed by a "tailor-made" solution. This solution requires the use of dedicated styles for ABB publications.

During editing, the user can at any time save his document in XML (and thus also validate it). If the document is not valid, the cursor moves to the problem paragraph. Validation errors are usually due to an incorrect choice of style.

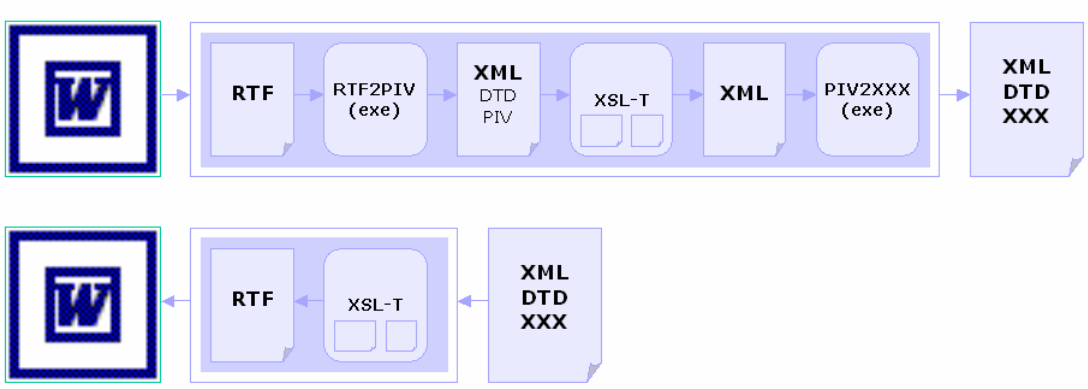
6.4.2. Technical description of the Word-based ABB editing tool

6.4.2.1. Tool architecture

The Word ABB tool in its broadest sense (at least, what is installed on author workstations) contains not only the ABB template, but also the XML to RTF and RTF to XML conversion filters.

With the exception of several Word macros and style definitions, the conversion and validation processes are carried out outside Word, by independent executable programmes.

The tool can be described using the following diagram:



When an **XML document file is being produced from Word**, a Word macro causes the following steps to be executed:

writing of the document in RTF format

activation of the programme "`rtf2piv`" to produce an XML pivot file

activation of the XSLT processor "`saxon`" to produce another intermediate XML file

activation of the programme "`wrx_parse`" to validate and convert this intermediate file into an XML document corresponding to the target DTD

When an **RTF document is produced from an XML document**, it is a Windows command file ("xml2rtf.cmd") that executes the following operations:

activation of the XSLT processor "saxon" to produce an RTF file from the XML document;

activation of Word to open the resulting RTF document.

The XML format mentioned above is the same as the "wrx" format; this "wrx" format is produced from the presentation format using certain constructions such as the cell size attributes (see DTD documentation).

The various technical steps in each conversion are described in the sections ""Repo" to RTF conversion" and "RTF to "repo" conversion", which are subsections of "Conversion filters and tools".

6.4.2.2. The Word template for the ABB publication

The editing tool actually uses two templates:

one template ("`seibud_wrx_97.dot`") is valid for all publications; this template is normally placed in the Word startup folder so it is always loaded. It is used only to provide a button that allows the user to open an object for editing from the dedicated directory without having to navigate through the file system.

the other template ("`abb.dot`") is specific to the ABB publication, and is placed in the templates folder and loaded only when an ABB document is being edited.

Macros

The macros in the ABB document template are split into three modules:

The "`Execute`" module is used to launch another programme and wait until its execution is terminated;

The "`Validate`" module essentially carries out the "F0_ValidateDocument" function, which activates the various stages of the conversion process and enables the cursor to be placed at the location of the error in the document;

The "`Macros`" module contains several macros associated with the buttons displayed on the "`wrx_seibud`" toolbar, namely:

`SeiBudSaveToXml` to save the document in XML (and validate it)

`SeiBudInsertDash`, `SeiBudInsertLQuote`, `SeiBudInsertRQuote`, `SeiBudInsertPM` to insert the corresponding characters

`SeiBudTableGalleryDialog` to open the table gallery

Styles

The styles used for an ABB document are mainly paragraph styles.

The following styles are used only to delimit the different sections using a keyword:

End of Annex

Figures

Figures Enlargement

Legal basis

Objective

Remarks

The other styles used are:

Annex

Article

CHAPTER

EXPENDITURE

1 Heading 1

1.1 Heading 2

1.1.1 Heading 3

1.1.1.1 Heading 4

1.1.1.1.1 Heading 5

Item

PART

REVENUE

SECTION

Table subtitle

Table Title

TITLE

Normal

6.4.2.3. Java Virtual Machine version and required libraries

This SEI-BUD client tool works both with Java Virtual Machine (JVM) version 1.4 and JVM version 1.5. Moreover, the migration from JVM version 1.4 to JVM version 1.5 has been successfully performed for the European Parliament setup environment.

The following table gives the list of standard Java libraries that are used by the Word-based ABB editing tool.

Library	Version	Description
saxon.jar	6.5.5	Free XSLT parser written by Michael Kay.

6.5. The Word-based Volume 0 editing tool

6.5.1. Functions of the Word-based Volume 0 editing tool

Functionally, this tool is like the ABB tool. It differs from it in terms of the DTD governing the structure of edited documents and in terms of the set of styles that can be used.

6.5.2. Technical description of the Word-based Volume 0 editing tool

The architecture of the Volume 0 editing tool is identical to that of the ABB editing tool. All that differs is the DTD, the Word styles used and some conversion filters.

6.5.2.1. The Word template for the vl0 document

The editing tool actually uses two templates:

one template ("`seibud_wrx_97.dot`") is valid for all publications; this template is normally placed in the Word startup folder so it is always loaded. It is used only to provide a button that allows the user to open an object for editing from the dedicated directory without having to navigate through the file system.

the other template ("`seibud_vl0_97.dot`") is specific to the VL0 publication, and is placed in the templates folder and loaded only when a VL0 document is being edited.

Macros

In addition to the macros functionally identical to those of the ABB editing tool, there are macros developed for the VL0 “publishing” functionalities, grouped in the module “`MacroWizardSplitVl0Files`”.

The VL0 “publishing” functionalities are:

the splitting of a multi-lingual document into several tomes,

the resolution of placeholders.

The VL0 “publishing” wizard is composed of 4 forms:

`DlgWizardSelectVl0Files`, for selecting input files,

`DlgWizardSelectVl0OutDirectory`, for selecting output directory,

`DlgWizardSelectVl0SectCut`, for selecting cut points when splitting a document,

`DlgWizardVl0CutInVolumes`, shows the progress and status of the work performed.

The cut points proposed to the user are either part elements of a VL0 document, or amendments of a Plenary Text report. Part elements start with a paragraph of “Titre objet” style, amendments are followed by a bookmark, whose name is prefixed by "amdbmk".

Placeholders are used for data that might change in the document. They have the character style “Placeholder”.

Placeholders

Name	Description
DOCLANGUAGE	The language code of the current document, followed by a section break “Next Page”
LANGUAGE (or LA)	The language code of the current document
NUMPAGES	The number of pages in (the tome of) the document
NUMTOMES	The number of tomes in the document
PAGE	Number of the current page
TOC[N]	The table of contents of (the tome of) the document, showing [N] levels
TOME	Number of the current tome

Styles

The styles used have been provided from LegisWrite 4.0 for Word 97:

volume heading is associated with SectionTitle style,

part heading is associated with Titre objet and Sous-titre objet styles,

sect1, sect2, … are associated with styles Heading 1, Heading 2, … respectively,

Annexe titre (globale) ¶

Confidentialité ¶

Emission ¶

1. HEADING 1 ¶

1.1. Heading 2 ¶

1.1.1. Heading 3 ¶

1.1.1.1. Heading 4 ¶

Heading 5 ¶

Heading 6 ¶

Heading 7 ¶

Heading 8 ¶

Heading 9 ¶

LANGUE ¶

`literal` ¶

`literallayout` ¶

Nom de l'institution ¶

Normal ¶

Placeholder ¶

Préliminaire titre ¶

Préliminaire type ¶

Référence institutionelle ¶

Référence interinstitutionelle ¶

ReusedInfo ¶

SECTIONTITLE ¶

Sous-titre objet ¶

Statut ¶

Table subtitle ¶

Table Title ¶

TableUnits ¶

Titre objet ¶

TOC Heading ¶

6.5.2.2. Java Virtual Machine version and required libraries

This SEI-BUD client tool works both with Java Virtual Machine (JVM) version 1.4 and JVM version 1.5. Moreover, the migration from JVM version 1.4 to JVM version 1.5 has been successfully performed for the European Parliament setup environment.

The following table gives the list of standard Java libraries that are used by the Word-based Volume 0 editing tool.

Library	Version	Description
saxon.jar	6.5.5	Free XSLT parser written by Michael Kay.

6.6. The Adept-based correcting tool

6.6.1. Functions of the Adept-based correcting tool

From a design point of view, the Corrector editing tool integrates perfectly with the SEI-BUD architecture and meets the specific needs of the Budget preparation procedure. In fact, whether it is the

preliminary draft Budget (PDB),

the draft Budget (DB),

or the Budget (B) (subject to certain restrictions concerning amendments)

that is being prepared, the Corrector editing tool allows correctors to make changes to the content of the budget (PDB, DB or B) in all languages.

From a design point of view, the Corrector editing tool has a dual purpose:

to provide the user (corrector) with a tool that is suited to the requirements of budget content amendments (in the PDB, DB or B phase);

to ensure, as far as possible without the knowledge of the user (Author), that changes are checked so the repository can then be updated with a correct instance.

Unlike the Author Word editing tools, etc., based on standard PC tools, the Corrector tool in question is a "native SGML" editing tool. For tables, the Adept-based editing tool uses CALS grammar to edit budgetary documents.

The ArborText Corrector editing tool is one of the modules of the SEI-BUD Corrector suite.

This tool is based on Adept, but is customised for SEI-BUD use. There are several reasons for this customisation:

to improve user comfort while reviewing budgetary documents during production;

to control user changes so that an update can subsequently be made to the Repository.

Most of the functions offered to improve user comfort are also designed to limit the user to a**specific**way of using the editor.

The Adept-based Corrector editing tool is a standalone tool. It is not linked directly with the server system. It is the control tool that sends requests to the server and receives responses from it. The control

tool and the Corrector editing tool have shared directories, the access paths of which are specified in the configuration files for these two programmes. The control tool supplies this directory with files containing the object for editing to be displayed or edited by the Adept-based Corrector editing tool. The editing tool supplies this directory with files containing the modified object for editing to be updated on the server system using the control tool.

The Adept-based Corrector editing tool is used to:

display the nomenclature of the file to be edited and position the cursor at a particular element of structure

display the data of an object for editing

modify the data of an object for editing

modify the remarks (deletion, addition, modification of paragraphs, lists, footnotes, etc.)

modify the text of tables (schedules, tables of Member States and other tables of remarks).

6.6.2. Technical description of the Adept-based correcting tool

There are several parts to the configuration of the Adept editor:

The DTD is specific to a particular publication (ABB or Volume 0) and is a "presentation" variant of the "repo" DTD.

Layout rules are expressed in the form of a FOSI (Formatted Output Specification Instance) and are also specific to a particular publication.

Menus and additional commands are programmed in ACL (Adept Command Language); some ACL modules are specific to a publication and others are common to both publications.

6.6.2.1. Menu

The SEI-BUD menu has been added to the standard Adept*Editor menu. The SEI-BUD menu contains the following functions:

Go to alias...	GoToAlias()
Keyboards	goes to the sub-menu Seibud.Keyboards
Insert italics	see popup menu
Remove italics	see popup menu
Insert superscript	see popup menu
Remove superscript	see popup menu
Insert citation	see popup menu
Remove citation	see popup menu
Insert Note	see popup menu
Remove note	see popup menu
Convert paragraphs to a list	see popup menu
Convert the list to paragraphs	see popup menu
Convert list item to list introducer	see popup menu
Convert list introducer to list item	see popup menu
Insert Non-breaking space	"insert($non_breaking_space)" command
Insert Em-Dash	"insert($em_dash)" command
SuperUser	RequestSuperUser()
Check Synoptism	Check_Synoptism()

Next Error	Goto_NextError()
Clear Errors	Clean_Doc()

The pop-up menu functions are described in the next section.

The function*GoToAlias*uses the command*find_attr_value*to search for an attribute with the same value as that given by the user. This command does not distinguish between attributes. It searches for the requested value among all the attributes of all the elements, for example, the identifiers.

The menus for the Greek and French keyboards use the commands*set keymap=greek*and*set keymap=french.*The keymap "French" is predefined in Adept*Editor, while the keymap "greek" is predefined in*pubrc*.

The function*RequestSuperUser*demands a password for access to all the Adept*Editor functions, i.e. the full editor menu.

The functions Check_Synoptism, Goto_NextError and Clean_Doc relate to the validation of synoptism.

6.6.2.2. Popup menu.

Extra options have been added to the popup menu. They are listed below, with their corresponding functions:

Insert Italics	InsertItalics()
Remove Italics	RemoveItalics()
Insert Superscript	InsertSup()
Remove Superscript	RemoveSup()
Insert Citation	InsertQuotation()
Remove Citation	RemoveQuotation()
Insert Note	InsertFootNote()
Remove Note	RemoveNote()
Convert Selected Paragraphs to a List	ConvertToList()
Convert List to Paragraphs	ConvertFromList()
Convert List Item to List Introducer	ConvertToListIntro()
Convert List Introducer to List Item	ConvertFromListIntro()

The popup menu is created in the file*popupmnu.cmd*, in the function SetPopupMenu, using the function*menu_add*.

ACL does not permit these options to be greyed out according to context, so they are always accessible.

6.6.2.3. List conversions

The functions for handling lists are located in the file*listproc.cmd*.

The function*ConvertToList()*first checks whether this is a valid operation. If so, it checks whether a selection has been made. If not, or if it does not cover the whole paragraph, the paragraph containing the cursor is selected.

Then, if more than one paragraph is selected, the user is asked whether the first paragraph should be converted to a list introducer.

The conversion itself depends on whether the paragraphs to be converted are located within a list. If they are, an ITEM must be inserted, and in this ITEM a P, and in this P a LIST, and for each P an ITEM in

which to put the P. Otherwise, a P must be inserted, and in this P a LIST, and for each P an ITEM in which to put the P.

The function*ConvertFromList()*first checks whether this is a valid operation (that the cursor is in a list, or a list has been selected). The conversion depends on whether the list to be converted is itself located within a list. If it is, only the tags of the LIST elements and of the P containing this LIST must be removed, and INT.LI and CLOSE.LI converted to ITEM. If it is not, the LIST and the P must be removed, as well as the tags INT.LI, CLOSE.LI and ITEM, so that their content is preserved.

The function*ConvertToListIntro()*checks that the cursor is indeed at the first item in a list. This item is then converted to INT.LI. If the ITEM is the only item on the list, an empty item is added behind the new INT.LI.

The function*ConvertFromListIntro()*checks that the cursor is indeed in an INT.LI. This is then converted into an ITEM.

6.6.2.4. Other functions of the popup menu

The other functions of the popup menu are located in the file*divers.cmd*.

The function*InsertItalics()*simply inserts an FT element by calling up the function*insert_tag*, and modifying the value of the attribute type to HT. The function*insert_tag*inserts an empty element if nothing is selected, and surrounds the selection if any has been made and if the insertion is permitted.

The function*RemoveItalics()*checks whether the cursor is located in an FT with the attribute TYPE = HT. If it is, the tag is removed without its contents, using the function*delete_tag*.

The function*InsertSup()*simply inserts an FT element by calling up the function*insert_tag*, and modifying the value of the attribute type to SUP. The function*insert_tag*inserts an empty element if nothing is selected, and surrounds the selection if any has been made and if the insertion is permitted.

The function*RemoveSup()*checks whether the cursor is located in an FT with the attribute TYPE = SUP. If it is, the tag is removed without its contents, using the function*delete_tag*.

The function*InsertQuotation()*simply inserts a QT element by calling up the function*insert_tag*. The function*insert_tag*inserts an empty element if nothing is selected, and surrounds the selection if any has been made and if the insertion is permitted. To display the automatically generated opening and closing quote marks, the command*set gentext=on*is executed.

The function*RemoveQuotation()*checks whether the cursor is located in a QT. If it is, the tag is removed without its contents, using the function*delete_tag*.

The function*InsertFootNote()*first checks whether the context allows the insertion of a footnote. If it does, and if there is nothing selected, a footnote is inserted, its TYPE attribute is changed to NUM, and an empty P is inserted there. If a selection has been made, a NOTE is inserted around the selected text, the attribute TYPE is set to NUM, and the content of the NOTE must be re-selected to be able to insert a P around this content.

The function*RemoveNote()*looks for a NOTE element surrounding the cursor. If there is none, the function does nothing. Otherwise, the footnote is selected and the user is asked to confirm the removal of

the footnote. The selection is then removed following confirmation, using the command*delete_mark*.

6.6.2.5. Presentation (FOSI)

Marking elements are presented declaratively, by defining the presentation characters, (fonts, colours, spacing, etc.) according to information such as the element name, attribute value and context in which the element is found. This presentation is defined interactively (using the Document Architect tool) and produces an SGML source file, which is then "compiled".

6.6.2.6. Initialisation of the Adept-based correcting tool

When a DTD is loaded, in a given directory, Adept*Editor reads and executes the file named*pubrc*. This file contains the commands to read and execute the files in which the functions described, programmed in ACL, are located.

A command file is read and executed using the*source*command. Executing a file generally consists of declaring and defining the functions that can then be called up (using menus, for example).

The SEI-BUD menu is loaded by the function*menu_load*, which is also located in the file*pubrc*.

6.7. The XMetal-based correcting tool

6.7.1. Functions of the XMetaL-based correcting tool

XMetaL is an editor for editing XML documents. It provides validation. It supports UTF-8 character encoding and the CALS table model, which are used in the budgetary documents. The outline of the document can be displayed in a separate pane (the structure view). The selections in the structure view and main document view are synchronized.

It is customized for two SEI-BUD DTDs (ABB and VL0), in order to improve user comfort while revising budgetary documents, and to control user changes so that an update can subsequently be made to the Repository.

The correcting tool is used by the user (corrector) to modify any textual content – not figures, images – of the budgetary documents in any language.

There are two modes of use of the correcting tool:

‘author’ mode, i.e. with the permission to insert/delete elements which are not inline and with modifications carried over all linguistic versions of the document,

‘translator’ mode, i.e. with the permission to insert/delete only inline elements and with modifications applied only to the reserved linguistic version of the document.

The XMetaL-based Corrector editing tool is a standalone tool. It is not linked directly with the server system. It is the control tool that sends requests to the server and receives responses from it. The control tool and the Corrector editing tool have shared directories, the access paths of which are specified in the configuration file for the control tool.

The control tool supplies this directory with files containing the objects for editing to be displayed or edited by the XMetaL editor, along with an identification card, containing information about the request addressed to the server, such as the mode of use (‘author’ or ‘translator’).

The editing tool supplies this directory with files containing the modified objects for editing to be updated on the server system using the control tool.

A launcher executable programme allows the activation of the XMetaL editor when the user ‘double-clicks’ the object for editing in the control tool.

The customization is loaded on the basis of the document type declaration of the XML document. The customizations set display options, add a toolbar and a menu, the buttons and commands of which are associated with custom macros.

Macros perform the following functionalities:

synoptism check between the modified document and the original,

conversion of paragraphs to a list and the reverse,

insert/remove`ft`and`qt`elements,

insert special characters (non-breaking space and em-dash).

6.7.2. Technical description of the XMetaL-based correcting tool

The SEI-BUD customizations contain, for each DTD (ABB and VL0):

the Document Type Definition (*.dtd*) or XMetaL compiled rules file (*.rlx*),

the XMetaL customization file (*.ctm*),

the XMetaL toolbar definition file (*.tbr*),

the cascading style sheets to format the main document view and the structure view (*.css*),

scripts (*macros*) in JScript (*.mcr*),

additional XSLT style sheets and an independent executable programme to perform the synoptism check.

6.7.2.1. Rules file

The DTDs are the ‘presentation’ variant of the ABB and VL0 DTDs.

The attribute type of the`id`attribute of the nomenclature elements of the ABB DTD has been changed to`CDATA`.

The DTDs are deployed in the form of XMetaL compiled rules files.

6.7.2.2. Customization file

The customization files are used to set a white-preserving behaviour for all elements containing text, and to designate the`p`element as a paragraph element (i.e. if**Enter**is pressed, XMetaL inserts a`p`).

6.7.2.3. Macros

The macros of the SEI-BUD customizations are programmed in JScript. They are associated with events and with the SEI-BUD toolbar buttons and menu commands.

Event macros

*On_Document_Open_Complete*macro creates a reference copy of the document in its original version that will be used for the synoptism check during the whole editing session. It is the control tool that deletes that file when finishing the editing session. This macro also reads the xml identification card of the object for editing to determine the mode of edition ('author' or 'translator').

The mode of edition determines whether:

synoptism check is suggested when closing the document (*On_Document_Close*macro),

the user may run the macros of conversion from/to list.

*On_Update_ElementList*macro removes the elements used for presentation purpose only from the element list (`prefix`element).

*On_Update_UI*macro sets the read-only permission on read-only elements, and disables the macros that cannot be used in the current view (some macros do not work in all views)

Command macros

*Save and check synoptism*macro implements a functionality specific to the 'translator' mode of edition. It compares the modified document to its original version to determine if 'non-inline' elements have been inserted/deleted. Both versions are normalized by an XSLT style sheet (`pre_diff.xslt`) and compared using the`xdiff`independent executable program. The result is transformed by the`elem-xupd.xslt`XSLT style sheet and the content is analysed in order to display the (first) non-synoptic elements to the user by selecting them in both versions of the document.

*Convert to list*and*Convert list to paragraph*macros are defined to carry out these conversion without having to insert/delete all the elements that define a list. They have been implemented for the**Normal**view of XMetaL. The elements`list`,`int.li`,`item`are inserted/deleted in conformance with the DTD.

*Toggle*macros (bold, italic, superscript, subscript, quotation) insert/delete`ft`or`qt`elements around the selection in the document.

*Insert non-breaking space*and*Insert Em dash*macros are shortcuts to insert these characters.

6.7.2.4. Launcher and deployment

The deployment of the customization and the activation of XMetaL when the user 'double-clicks' on the object in the control tool are performed by programmes –`xmetal_copy.exe`and`launcher.exe`respectively – which read the following registry key of the local machine zone:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\xmlfile\shell\Edit in &XMetaL\Command
```

6.8. The Word-based amendment editing tool

6.8.1. Functions of the Word-based amendment editing tool

The Word-base amendment editing tool is based is customization of the Windows based Word processor.

It is used by **authors** and **translators** to edit or translate the text parts of the budget: headings, remarks, legal basis and references, notes, justifications.

The benefits of having a specific tool are:

that it makes editing simpler: the author modifies the text (headings or remarks) without having to worry about the figures;

it guarantees, by means of a validation procedure, that changes are correctly integrated in all language versions of the budget document at update.

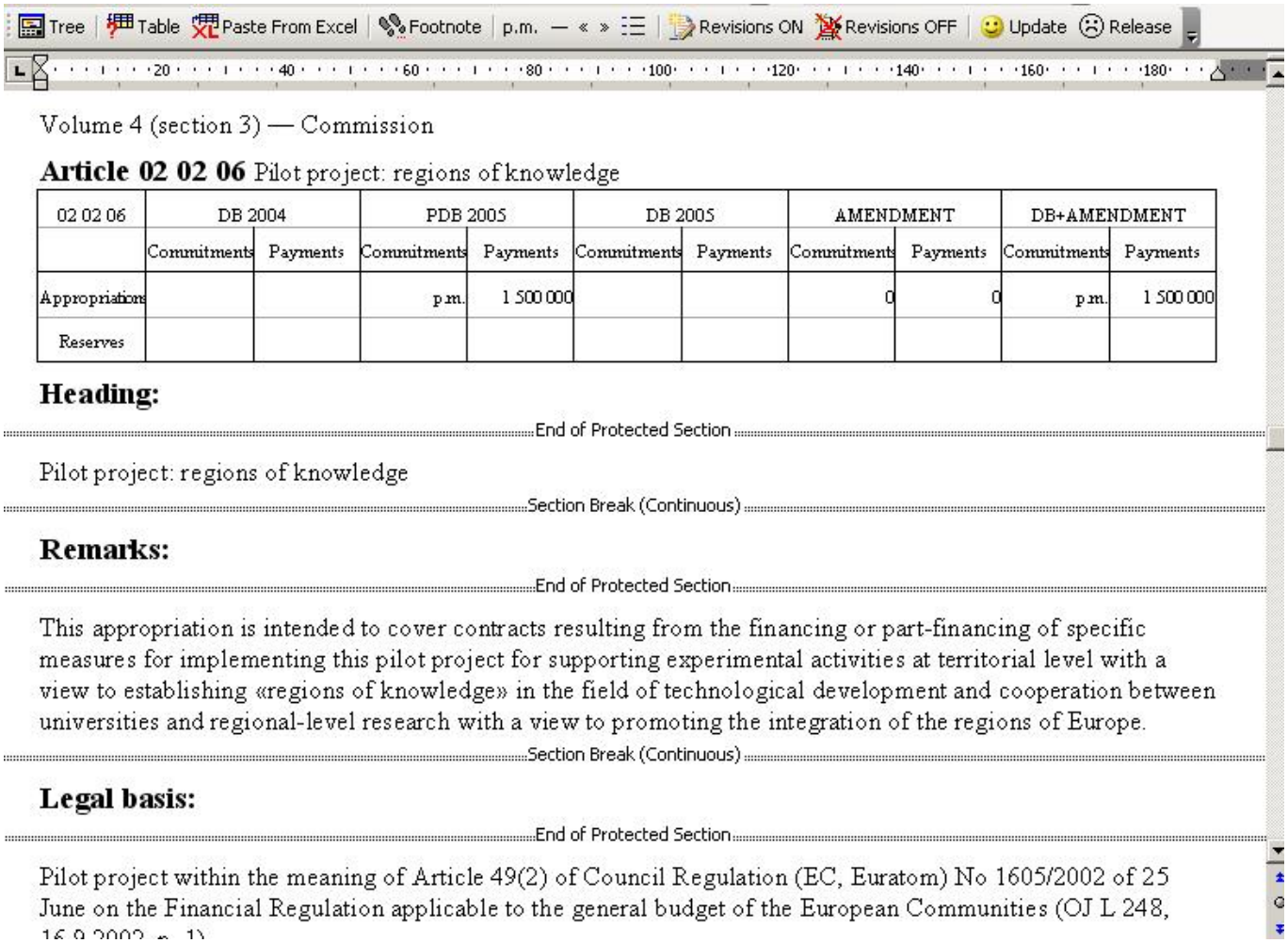
There are a number of reasons why Word was chosen as the remarks text-editing tool. The main reasons are:

all the authors are familiar with using Word, which was already available on all workstations, so there was no need to purchase an additional tool, nor to learn how to use it;

Word has a large number of additional functions and productivity tools (spell-checker, grammar-checker, thesaurus, macros, and even computer-assisted translation tools).

6.8.1.1. The amendment author functions:

The amendment document is presented as a sequence of protected and un-protected sections. Protected sections contain information that cannot be modified with the Word-based editing tool but is useful for authoring work: protected information includes structural components boundaries for headings, remarks, legal references etc and figures information. Un-protected sections basic text content that can be filled and modified by the author.



In addition to all standard Word functions, the customization adds one main menu and a toolbar with the following features:

`atree`navigation command that opens a window allowing navigation within the structure of the amendment (this feature is in fact useless since it is redundant with the built-in document map feature with properly defined styles);

the`table`commands opens a selection window showing various predefined tables that can be picked-up and included within the document;

the`Paste from Excel`command can be used to transfer table data from the Excel application;

the`Footnote`command is used to manipulate budget document footnotes (the user does not have access to the standard Word footnote feature);

there are four specific commands that can be used to insert specific strings: "p.m.", em-dash, left and right quotes, and em-dash list item indicator;

the`Revision ON`and`Revision OFF`buttons can be used to toggle display of revision marks;

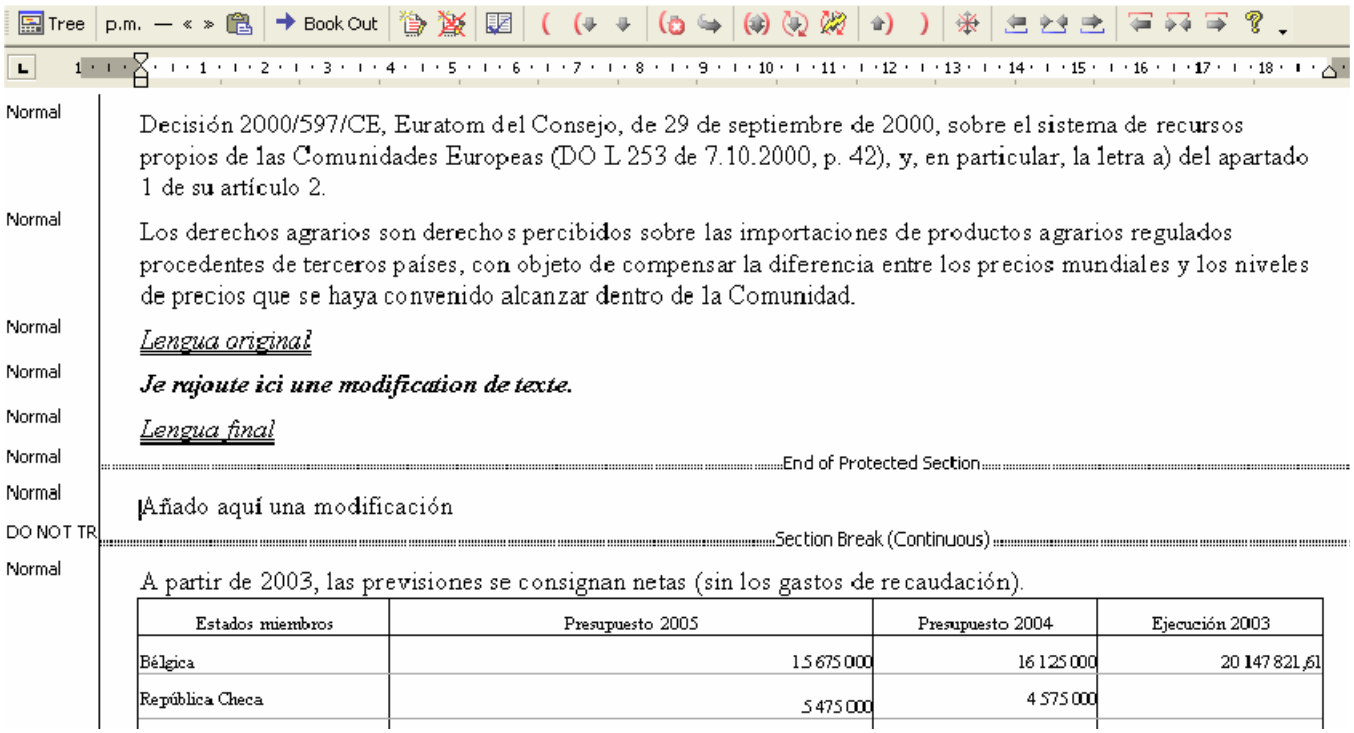
and finally, the most important commands are 

`Release`and`Update`used to respectively cancel the reservation and commit changes.

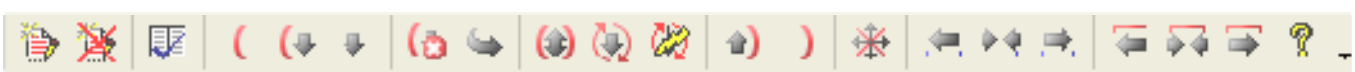
The template also customizes some of the standard Word commands like "Open" and "Save", it disables some other commands and adds a menu item in the help menu in order to get help about the tool.

6.8.1.2. The amendment translator functions

In the Word-based translator editing tool, the amendment document is also presented as a sequence of protected and un-protected sections; the protected section includes the source language version of the text to be translated.



The additional functionalies available in the Word-based translator editing tool are the same as those available for the author, except that:



there is an additional set of commands (available in a menu and a toolbar) for accessing Trados Translator WorkBench (TWB) facilities;

a`Paste unformatted text`is added as a shortcut to the Paste Special / Unformatted text standard command;

a`Book Out`command is added (it is equivalent to the`Update`command of the author);

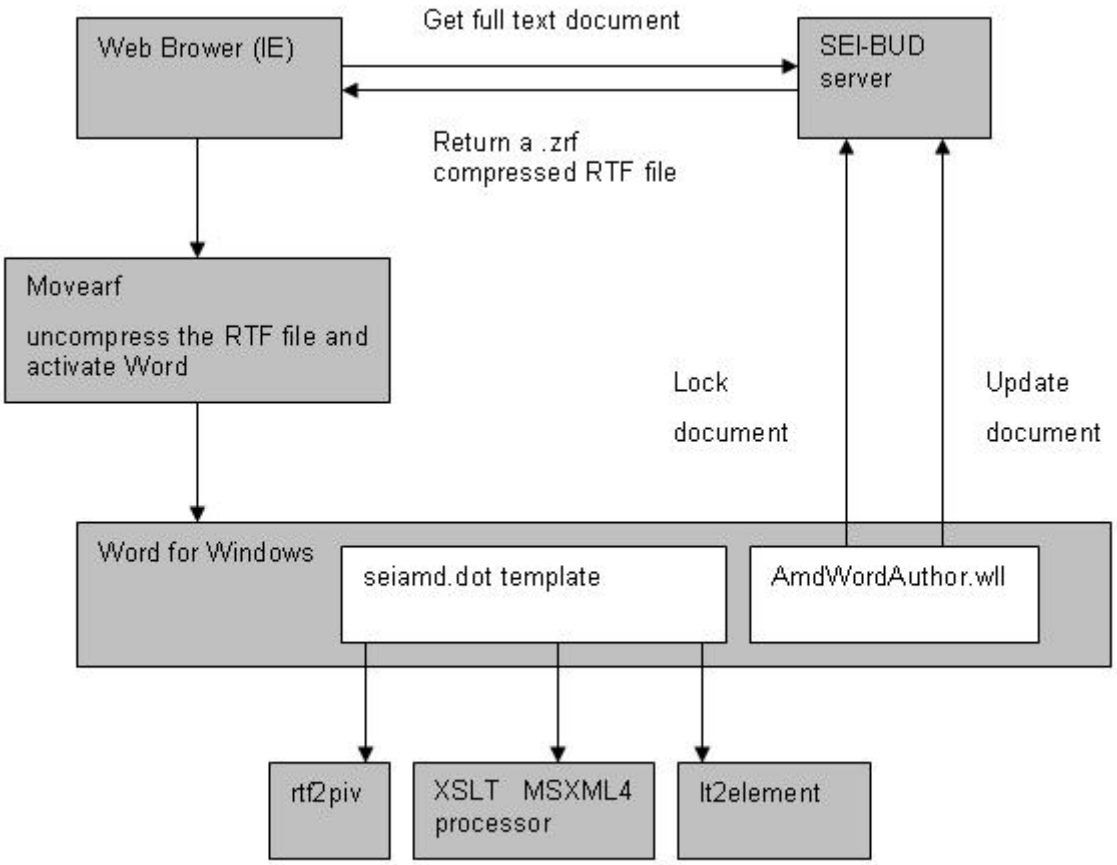
a`Check Synoptism`command allows the translator to check that the translated document has the same basic text components (paragraphs, lists, ...) as the author document.

The template also customizes some of the standard Word commands like "Open" and "Save", it disables some other commands and adds a menu item in the help menu in order to get help about the tool.

6.8.2. Technical description of the Word-based amendment editing tool

6.8.2.1. Tool architecture

The various components intervening in the Word-based amendment editing solutions can be roughly depicted with the following block diagram:



The `movearf` program is responsible to launch Word. Word is customized using Visual Basic for Applications for applications code inside Word templates and a Word Link Library (WLL). There is 3 different word templates depending on the user role: authors (`seiamd.dot`), translators (`seiamd_trad.dot`) and other users that cannot modify the document (`seiamd_cons.dot`).

When saving the document after modifications, a series of operations are performed on the document before committing the document to the SEI-BUD server; these operations involve the following components: `rtf2piv` to convert RTF to XML (pivot), XSLT processing using the MSXML4 package and the `lt2element` program that will convert pivot XML into target DTD compliant XML.

6.8.2.2. Metadata attached to the amendment document

The following information is added to the RTF document file when by the SEI-BUD server:

the amendment identifier (referring to the id_institution_amendment column)

information identifying the amended publication:

the budget year (e.g. "2005");

the type of amended publication (e.g. "ABB")

the budget preparation step of the amended publication (e.g. "AP");

an integer specifying the instance number of the publication;

the amendment label;

the amendment number (optional);

an indicator specifying if the amendment is multilingual;

the base language;

the amendment phase type (e.g. "PE");

the format (e.g. "FT");

the document language;

the version;

the user having checked-out the document;

the date of check-out;

This information is by the SEI-BUD server in to the XML document checked-out; it is then converted to RTF and put in a protected section. The contents of this protected section is used:

by the movearf program to know how to reserve the document for edition (lock) and how to work in the local environment depending on the user role;

when saving the document, this information is kept along in the conversion and sent back to the server on commit; it is used by the WebInt servlet code to issue a checkin request to the application server;

it is also used by the WLL code to unlock the document after successful checkin.

6.8.2.3. The "seiamd.dot" template

This template is attached to documents that are intended to be modified by authors.

The "seiamd.dot" template contains the following Visual Basic for Applications (VBA) macros; corresponding to the above-mentioned commands:

BudgetOpenTree: open the document structure dialog

BudgetTable: insert/update table from table gallery

BudgetPasteTableFromExcel: paste from selected excel data

BudgetItalic: put selection in italic

BudgetFootnote: insert a budget type footonote

BudgetInsertPM: insert the "p.m." string

BudgetInsertMDash : insert a em-dash

BudgetInsertLQuote: insert a left quotation mark

BudgetInsertRQuote: insert a right quotation mark

BudgetReleaseDoc: release document lock and discard changes

BudgetUpdateDoc: save document and commit changes

BUDRevisionsON: show revision marks

BUDRevisionsOFF: hide revision marks

InsertEmdashList: insert a list type em-dash.

The`BudgetUpdateDoc`macro uses the`ValidateDocument`procedure (`validate`module) and then calls the`BudUpdateDoc`function of the WLL. This validation procedure causes the following steps to be executed:

saving of the document in RTF format

take a temporary copy of the RTF document;

clean-up of the document copy, that is removing all "SKIPML" sections containing information that does not make part of the translated document but is merely help information for the translator (such as source language text);

remove all section breaks;

activation of the programme "`rtf2piv`" to produce an XML pivot file;

apply the`restoreStructure.xsl`stylesheet to the pivot file, producing a intermediate file named`structured.xml`;

attempt to convert quote characters into`qt`elements using the`lt2element.exe`program; if the resulting file is not well-formed, this step is not used;

apply the`piv2amd.xsl`stylesheet producing a named pivamd.xml;

apply the`hpos2colspec.xsl`stylesheet converting`hpos`attributes into`colspec`attributes.

6.8.2.4. The "seiamd_cons.dot" template

This template is attached to documents that can only be viewed but not modified.

The "seiamd_cons.dot" template contains the following Visual Basic for Applications (VBA) macros; corresponding to the above-mentioned commands:

BudgetOpenTree

BudgetReleaseDoc

BUDRevisionsON

BUDRevisionsOFF

6.8.2.5. The "seiamd_trad.dot" template

This template is attached to documents intended to be modified by translators or correctors.

The "seiamd_trad.dot" template contains the following Visual Basic for Applications (VBA) macros; corresponding to the above-mentioned commands:

BudgetOpenTree

BudgetFootnote

BudgetPasteTableFromExcel

BudgetItalic

BudgetInsertPM

BudgetInsertMDash

BudgetInsertLQuote

BudgetInsertRQuote

BUDDeactivate

AMDBookOut

BUDRevisionsON

BUDRevisionsOFF

InsertEmdashList

PasteUnformattedText

CheckSynoptism

MultipleOpenTranslateToFuzzy

RunFuzzyTrad

The`AMDBookOut`macro first checks the linguistical synoptism by calling the`CheckSyno`procedure, then uses the`ValidateDocument`procedure (`validate`module) and finally calls the`AMDTradBookOut`function of the WLL. As in the case of an author document, the validation procedure causes the following steps to be executed:

saving of the document in RTF format

take a temporary copy of the RTF document;

clean-up of the document copy, that is removing all "SKIPML" sections containing information that does not make part of the translated document but is merely help information for the translator (such as source language text);

remove all section breaks;

activation of the programme "`rtf2piv`" to produce an XML pivot file;

apply the`restoreStructure.xsl`stylesheet to the pivot file, producing a intermediate file named`structured.xml`;

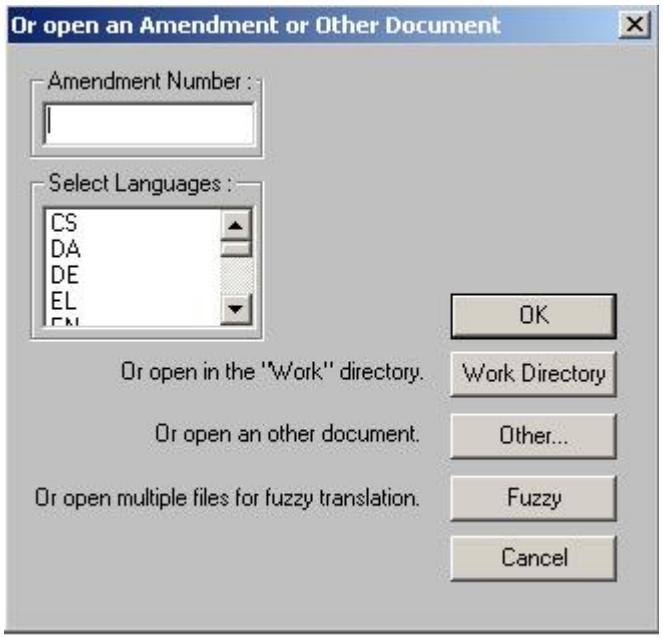
attempt to convert quote characters into`qt`elements using the`lt2element.exe`program; if the resulting file is not well-formed, this step is not used;

apply the`piv2amd.xsl`stylesheet producing a named pivamd.xml;

apply the`hpos2colspec.xsl`stylesheet converting`hpos`attributes into`colspec`attributes.

The`CheckSyno`procedure is written in VBA. It examines all paragraphs of un-protected sections and checks if their count is the same in the current document version than the original one.

This template also customizes the "File/Open" command to open the following dialog box:



The`Work Directory`button can be used to display the contents of the working directory;

the`Other`button opens the standard Open dialog box of Word;

the`Fuzzy`button can be used to open several documents for fuzzy translations with Trados.

Interface with Trados Translator WorkBench

The translators of amendments get a Word document with protected sections; protected sections enclose the material that should not be translated. The problem of this approach is that Translators Workbench (abbreviated as TWB; from the German company Trados) does not support protected sections.

Remark: based on the experience acquired during several years of Word customization, it appears that it is a bad practice to use Word tools with too much customization. From a users point of view it often a disturbing. From a developer point of view, it is a real nightmare to support the successive versions of Word. This is why a strict minimum of customization has been done for the non-amendment documents of SEI-BUD (amendment documents still use an intrusive customization for historical reasons).

The actual version of the "seiamd_trad" template is based on version 6.5 of Trados (the template is named "trados6.dot". The previous version was based on version 4.5 and included most of the Trados VBA (Visual Basic for Applications) code with lots of corrections and fixes; this situation led to difficulties in upgrading the template with new versions or updates from Trados.

In an attempt to minimize the amount of work needed to integrate Trados updates, two alternatives were examined:

The first alternative is to create a wrapping layer to make calls to the external Trados template

The second alternative is to copy all Trados stuff into the "seiamd_trad" template, create a wrapping layer that makes internal calls to the copied Trados code

The first alternative, while being the cleanest one, is not possible for the following reasons:

New Word documents are open for footnote translations and there is public way to take control of document creation in order to attach the "seiamd_trad" template and get access to amendment specific functionality

There are still bugs in the current version of Trados template

The second approach has been taken and only 3 Trados modules differ from their original version:

tw4winMain.bas

tw4winMenu.bas

tw4winEnable.bas

When a new version of Trados must be supported, the following actions must be taken:

Export all seiamd_trad modules

Export all modules from the new version of the Trados template

Export all modules from the old version of the Trados template (all modules starting with "tw4win" plus "AutoExec.bas", "FileClose.bas", "FileExit.bas", and "ToolsProofing.bas")

Perform a three-way merge of the 3 modules that actually differ from the original version

Import the new Trados modules and the 3 merged ones

Check that all references are still available and if applicable, add toolbar and menu items for new Trados commands

Perform extensive tests

The Trados template is deactivated. The "seiamd_trad" template contains a dedicated toolbar with a copy of all tools available in the original Trados template; these copies of tools make calls to the wrapping layer. It also contains a copy of the Trados menu (named "SEI-Trados"); however, each menu item makes calls to the wrapping layer. The development environment also contains a "seiamd_trad_toolbar.dot" file containing a toolbar used to generate the "SEI-Trados" menu ("tw4winMenu.Add" and "tw4winMenu.Check" functions).

The following changes have been made to the original Trados modules:

tw4winMain

- the "sOpenFootnoteWindow" has been adapted in order to attach the "seiamd_trad" template to the footnote document
- the "bCloseFootnoteWindow" has been adapted to make sure the footnote document is unprotected, to remove color highlighting and to permit detection of which type of document is being closed. Also paragraph information is saved before footnote processing and restored after.
- problems have been fixed in the "fNextTextParagraph" function
- the "nGetNextFootnote" function has been adapted to fix problems

tw4winMenu

this module contains helper functions used to create a menu from a toolbar and change the labels depending on the user interface language

the list of commands has been adapted to refer to the wrapping layer

the menu and toolbar names have been modified

tw4winEnable

the menu and toolbar names have been modified

Note that the "amdtranslator.rtf" document can be used to load the toolbar and the menu in order to get access to the "Multiple Translation to Fuzzy" functionality.

6.8.2.6. The "movearf.exe" program

This program is registered to open the ".zrf" document files. For instance, when clicking on a Full Text document amendment icon inside the Web browser, the Web server sends a ".zrf" document in a compresses RTF format. The Web browser calls the system to open that document with the default program for this extension: the`movearf`program.

This program is written in C++. It decompresses the ".zrf" file into an RTF file and puts that file in the proper location based on settings specified in the "SEI-BUD.ini" file

6.8.2.7. The "AmdWordAuthor.wll" Word Link Library

This component is a Word specific dynamic link library (WLL) used as an Addin to Word. It is loaded by the above-described templates and contains interface functions that are called from within VBA code inside Word templates. It is written in C++ and contains the following entries:

BudItalic to set the current selection in italics

EditCopy to the current selection

EditCut to cut the current selection

EditPaste to paste the current selection

BudOpenTree to open a dialog for navigating the document structure;

BudPasteTableFromExcel: to paste copied Excel selected data into a table;

BudTable: to open a dialog allowing selection of a table for insertion or modification;

BudFootnote: to insert/update footnotes under specific control;

BudHelp: to get help about the Word-based amendment editing tool;

FileSaveAs: to restrict the user from changing the filename of an amendment document;

FileOpen: performs differently depending on the user role (author or translator); for a translator, it opens a customized open dialog box (see above) and for an author, it adds some more checks when an amendment document is open.

BudInsertPM: inserts the "p.m." string

BudInsertMDash: inserts the em-dash character;

BudInsertLQuote: inserts a left quotation mark;

BudInsertRQuote: inserts a right quotation mark;

BudReleaseDoc: is used to cancel editing changes and release document reservation;

BudUpdateDoc: is used to commit editing changes;

BudInit: initializes the WLL in authoring mode;

BudInitAmdTranslator: initializes the WLL in translation mode

AMDTradBookOut: commit translator changes;

AMDFuzzyTrad: performs a MultipleOpenTranslateToFuzzy in TWB jargon; this is some code wrapped around the "RunFuzzyTrad" TWB macro; this code performs some additional checks before running and macro and also un-protects and protects back the document. Indeed, TWB macros do not properly support protected sections and each TWB operation must be enclosed with un-protect/protect commands.

AMDTradRun: starts the translator workbench;

6.8.2.8. The "rtf2piv" program

The "rtf2piv" program is used to convert an RTF file into an XML document. Is an executable SAG tool developed by SAG used in many SAG projects, it is available both on Windows and on Solaris. This program is described in more details in the "Conversion filters and tools" section of the "The Repository Manager and Communications sub-system" chapter.

6.8.2.9. The "lt2element" program

The "lt2element" program implements a quite nice trick: it converts element names enclosed with general entity character references to real XML tags. For example, it will convert the string "`<TRANSACTIONS>`" into the string "`<TRANSACTIONS>`".

6.8.2.10. The "hpos2colspec.xsl" stylesheet

The column width information extracted from RTF files is available with the "hpos" attribute that defines the right boundary of a table cell, including its half of the space between cells in twips. This stylesheet is used to transform this cell information in terms of column width, according to the CALS table model. It also generate "colspec" information.

6.8.2.11. The "piv2amd.xsl" stylesheet

This stylesheet converts pivot XML documents into XML documents compliant to the DTD for amendment documents.

6.8.2.12. The "restoreStructure.xsl" stylesheet

This stylesheet is used to remove text that was hidden in Word and to enclose in a SEIAMD-ELEMENT the text that formatted with the SEIAMD-ELEMENT style; this later style actually hides text and is included in protected sections of the document (what a trick isn't?).

It has to be noted that the result of this conversion is used to check whether it is still well-formed: if it is not well-formed, it probably means that the user has unprotected the document and tweaked with hidden information.

6.8.2.13. The "SEI-BUD.ini" file

The "SEI-BUD.ini" configuration file is be located in the Windows root directory usually`c:\winnt`or`c:\windows`). It contains configuration information needed by client tools and is divided into 3 sections:

the "**[SEI-BUD]**"contains general installation information:

the "**Home Directory**"entry specified the root directory of the amendment tools (e.g. "`C:\local\seibud`");

the "**[SEI-BUD Word Tool]**" contains information relevant to the Word-based tool:

the "**Data Directory**" entry specified where document files are stored (e.g. "`C:\local\seibud\files`");

the "**Program Directory**" entry indicates where to find the Word Link Library ("`C:\local\seibud\WordWLL`" and is used by template code to load that library);

the "**AMDWll**" entry specifies the filename of that WLL (normally "`AmdWordAuthor.WLL`");

the "**Standard Footnotes**" entry indicates where to find predefined footnotes text (by default: "`C:\local\seibud\WordWLL\StandardFootnotes.txt`")

the "**Tables Templates**" entry specifies the location of the gallery of table templates ("`C:\local\seibud\WordWLL\TableGallery`");

the "**Stylesheet Directory**" entry contains the path of the directory containing stylesheet source code ("`C:\local\seibud\WordWLL\rtf2amd`");

the "**ZipExtension**" entry specifies the extension name for compressed RTF files (should always be "`zrf`");

the "**Analyse Clipboard**" entry seems not used anymore;

the "**Open Non Budget Documents**" entry indicates whether or not it is permitted to open non-budget documents using the standard open dialog box from within Word (default is "1");

the "**Online**" entry indicates whether the client is connected to the server or not

the "**compressed**" entry indicates if the modified document must be sent to the server in compressed form or not (by default, it is "1");

the "**AMDWll Languages**" entry specifies the list of languages used to load the language list in the open dialog "`CS.DA.DE.EN.EL.ES.ET.FI.FR.HU.IT.LT.LV.MT.NL.PL.PT.SK.SL.SV`"

the "**Registry Language Key**" entry indicates the registry key pertaining to the "DocEP" package where to find settings about document language (e.g.: "`Software\European Parliament\DocEP\User Settings\Settings`")

the "**Registry Language Value**" entry specifies the default value for the previous entry ("`DocumentLanguage`")

the "**Use Proxy**" entry indicates how to connect through the HTTP protocol to the server: either via a named proxy or using the registry configuration.

the "**[Connection:Production]**" section contains information needed to make connections to the server; there can be several sections of this type to reflect several different servers (e.g. [Connection:Test]):

the "**Translators.Bookout.Transfer.Method**" entry indicates the transfer method used to commit the changed document to the server; the only available method is "`HTTP`";

the "**out**" entry indicates the bookout directory ("`P:\APPLS\SEI-AMD\OUT\`");

the "**Online**" entry indicates whether the client is connected to the server or not

the "**Http Cmd**" entry specifies the URL to be used for committing the changed document to the server (e.g.: "`http://localhost:8081/WebintServlet/servlets`");

the "**File Server**" entry specified the path to be used for document file storage concerning the three following entries (e.g.: "`C:\local\seibud\files`");

the "**work**" entry specifies the document working directory (e.g.: "`C:\local\seibud\files\amd\`");

the "**in.bak**" entry specifies the backup directory for input document files (e.g.: "`C:\local\seibud\files\in.bak\`");

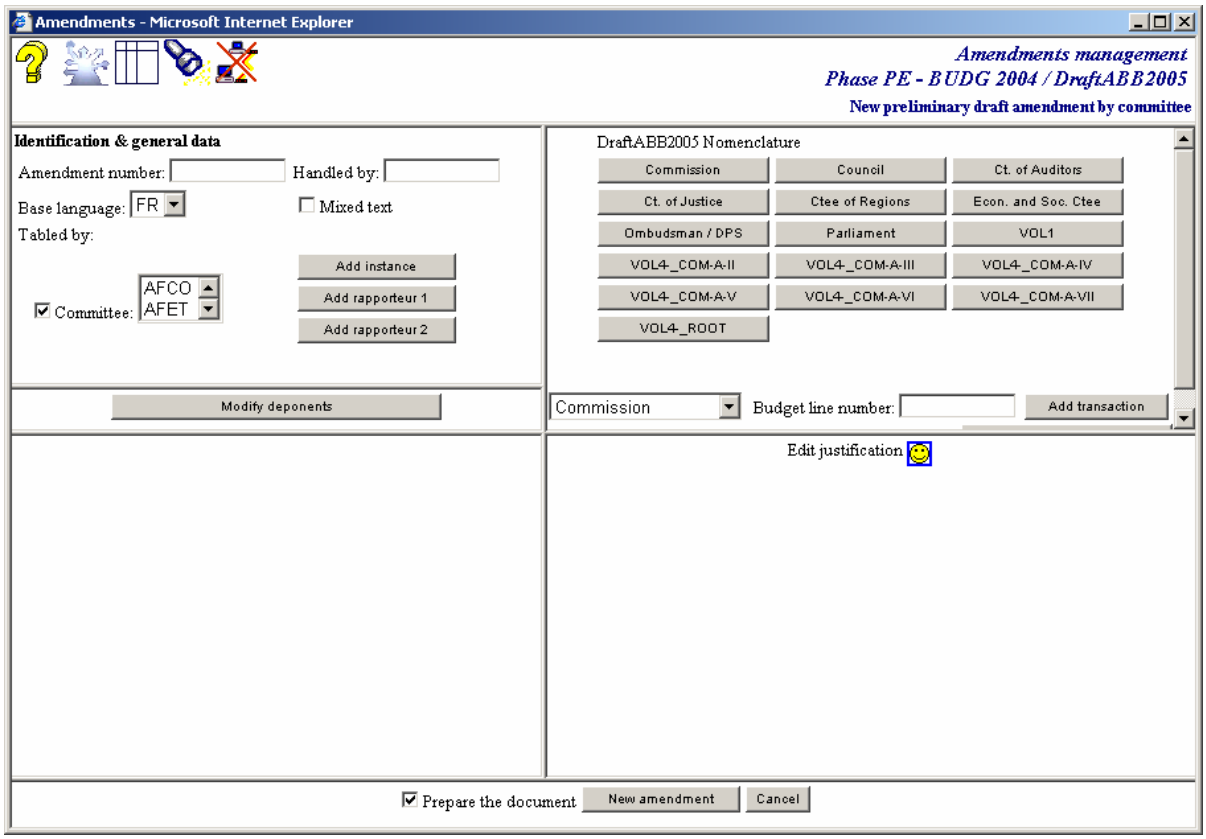
the "**out.bak**" entry specifies the backup directory for output document files (e.g.: "`C:\local\seibud\files\out.bak\`");

6.9. The Web-based interface to amendments

6.9.1. General Principles

6.9.1.1. The visible part of the iceberg

Webint generates HTML 4.0 with a small amount of CSS 1 styles. HTML frames are heavily used to minimize the flicker effect when Webint is accessed through poor communication lines:



Most of the screens are divided into 3 main parts:

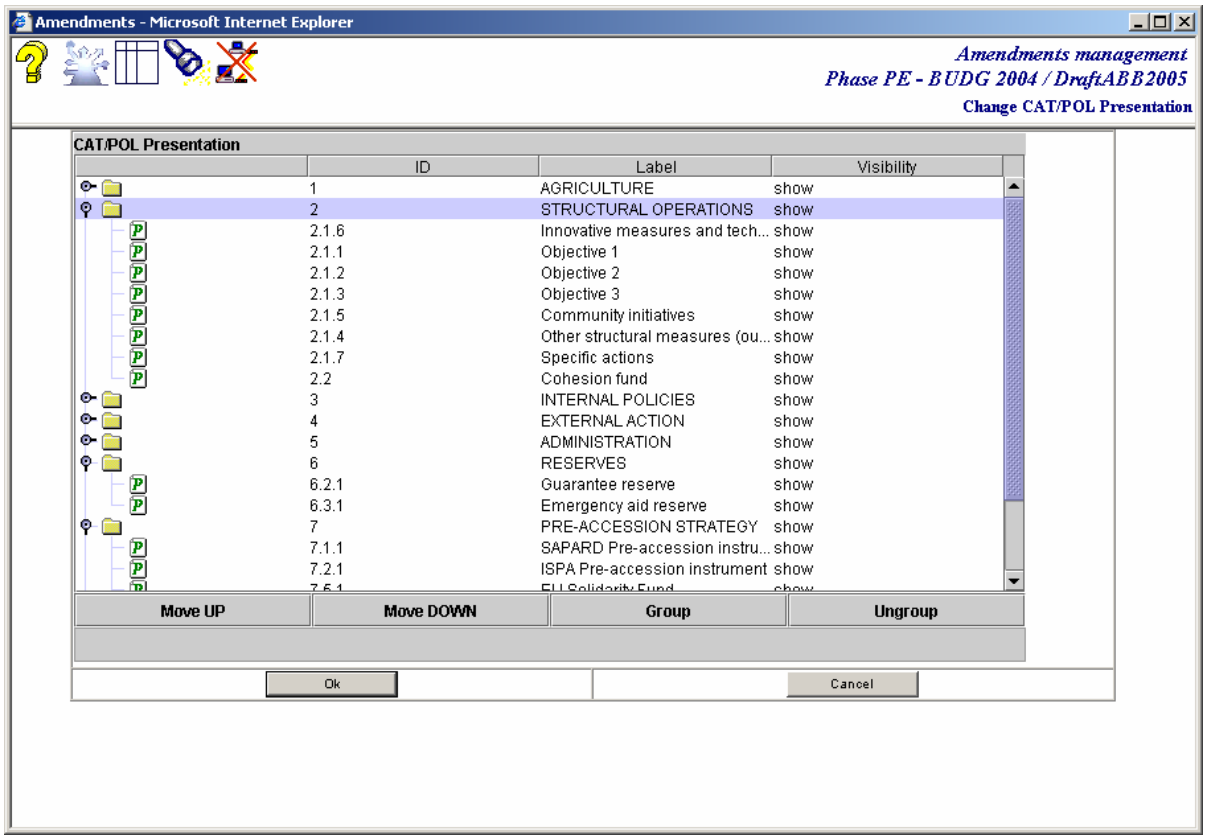
The*header*is the most static part. Only the subtitle varies.

The*main part*in the middle of the screen contains most of the data. Here the user interacts with the system to enter the figures, to build the transactions, to list the amendments or to input various other data.

The bottom part mainly contains 2 buttons: one to confirm the current action and one to cancel it.

6.9.1.2. Java applets

At some moments, a more graphically oriented interaction is needed. These special usages are covered by Java applets running in a Sun 1.4.x or Sun 1.5.x virtual machine:



Concretely, the Java applets are used in the following situations:

The nomenclature i. e. the structure of a budget document is shown in a tree-like fashion. The user may select one or more budget documents. Two variants exist according to whether the selected rows should be shown separately.

The user may input the votes in a special list of amendments where she tells the system whether an amendment has been approved, rejected or withdrawn. Moreover, she may group some amendments together for which a single decision is taken.

A special applet handles the list of amendments for which a report should be generated. The list is very dynamic because the order of the amendments is important and because the user may freely add or delete amendments.

The CAT/POL codes and labels are shown in a tree to build up a presentation. A presentation defines a particular way of organizing the categories and the politics.

6.9.1.3. Client-side Javascript

Moreover, Webint uses client-side Javascript mainly to validate the user input and to construct the response to be send back to the server. For instance, in the list of amendments, you may select several amendments and send them to translation. A client-side Javascript scripts looks for all the selected amendments, builds a long string referencing all selected amendments and puts the result in a hidden field. The hidden field is sent to the server to make the real translation.

The most complex Javascript is the one associated to the amendment figures and to the manual schedules.

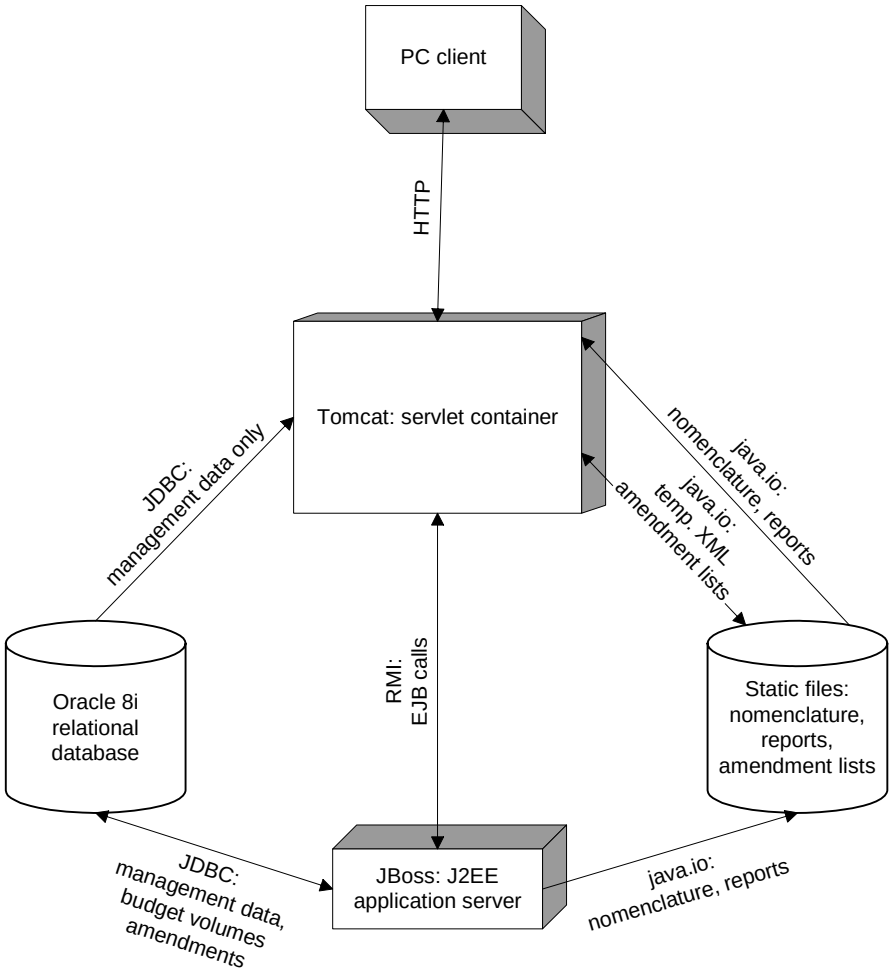
6.9.1.4. Behind the scenes Java works

The whole Webint application is implemented as a collection of Java servlets. The servlets are packaged in a file called`WebintServlet.war`. Jakarta Tomcat, version 3, runs the servlets. No JSP are used because when the development of the application started, JSP was not out yet.

A Java 1.4 virtual machine must run Tomcat as some Java classes use features that are only found in the 1.4 Java libraries.

6.9.1.5. Architecture: the big picture

The general data flow is as follows:



Inside Tomcat, the Webint servlets handle all the incoming calls and pass them on to some helper classes.

6.9.1.6. Logical subdivision of work

Following a*Model-View-Controller*paradigm, the Webint servlets handle the*View*part and the*Control*part.

The*View*part is the main responsibility of Webint. Webint converts the raw data into HTML presentation artefacts like lists and tables.

The*Control*part is not done entirely in Webint. It is limited to the sequencing of the screens, to the partial validation of the input, to the handling of the access control and to the session management.

Webint never directly modifies the persistent data, but it passes over the update requests to the application server (JBoss).

As can be deduced from the big picture, Webint does not ask the application server to retrieve the persistent management data; it rather accesses the Oracle management database by its own. This means that Webint and the application server must share a common database model and every change to the database model may impact Webint. The reason for this asymmetry is to enhance the performance and to alleviate the workload on the application server. Moreover, the architecture has evolved from historical constraints. In the past years, before the introduction of the JBoss application server, a proprietary solution based on SiT (Sgml Integrated Toolkit) was used. That solution was not able to handle the requested volume of data and there was no API available to marshal/unmarshal comfortably the SGML/XML data.

On the other hand, the generation and retrieval of the editorial data, the amendments, is exclusively done via the JBoss application server. The reports are the only exception. They are asked once. The application server stores them in the file system and Webint retrieves them from there whenever the user asks for it. This is done in an asynchronous way.

6.9.2. Session management

HTTP is a stateless protocol. By default, a session only lasts for the duration of a single HTTP request. The Java servlet specifications always included a mechanism to preserve sessions. The specifications recommended two methods to identify a session through several HTTP requests:

*Cookies*have been invented by Netscape. Small pieces of data are stored by the browser. They are Web site specific and sent along every request made to a specific site. A session identifier may be stored in such a cookie to help the web site find the right session.

*A session identifies*may be stored directly in every page associated to the session. Either every URL or every form of the page must contain the session identifier. In a form, the session identifier is stored as a hidden field.

Webint has been written before the finalisation of the first Java servlet specification. Therefore, it makes usage of a single cookie called “AMDSESSION”, but it contains a proprietary API to manage the session and to retrieve associated the session data.

In Webint, only some servlets may serve as entry points to build a new session:

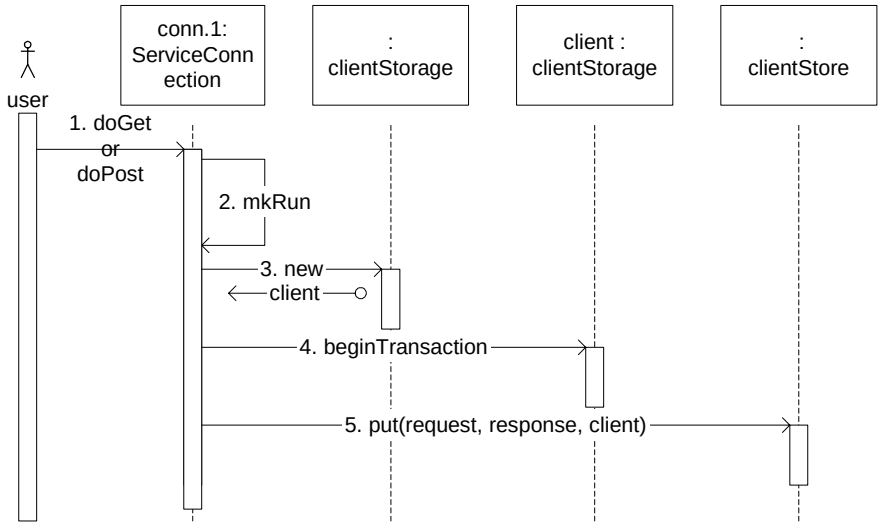
Consultation.java

ServiceConnection.java

ServiceLogin.java

ServiceLoginRepo.java

The following interaction diagram shows what happens when a request arrives at the ServiceConnection.java entry point:



All the data persistent to a session is kept in`clientStorage`object. An object of this this type is passed as a parameter to every method of Webint that wants to access the persistent data.

The cookie is created in the`clientStore.put`call and transmitted to the`HTTPServletResponse`object`response`.

In the previous diagrams, there is a call to the method`beginTransaction`. This method does not initiate a database transaction, but only a transaction on the`client`object. As a matter of fact, every HTTP request accessing a`clientStorage`object must absolutely call`beginTransaction`before going on and must finish the request with an`endTransaction`call. The methods`beginTransaction`and`endTransaction`implement a semaphore. This ensures that a`clientStorage`object is not accessed concurrently by more than one request at a time. A concurrent access might corrupt the data!

ServiceConnection.java is the most important entry point because only sessions created there are effectively used over several HTTP requests.

The following servlets use an existing session:

`ServiceAmdCol2.java`

`ServiceAmdManag.java`

`ServiceAmdSearch.java`

`ServiceButtons.java`

`ServiceConnection.java`

`ServiceCreate.java`

`ServiceDrawHeader.java`

`ServiceEdition.java`

`ServiceEditTransaction.java`

`ServiceEditTransactions.java`

`ServiceExit.java`

`ServiceGenerateCSV.java`

`ServiceList.java`

`ServiceMainFrame.java`

`ServiceQuickEdition.java`

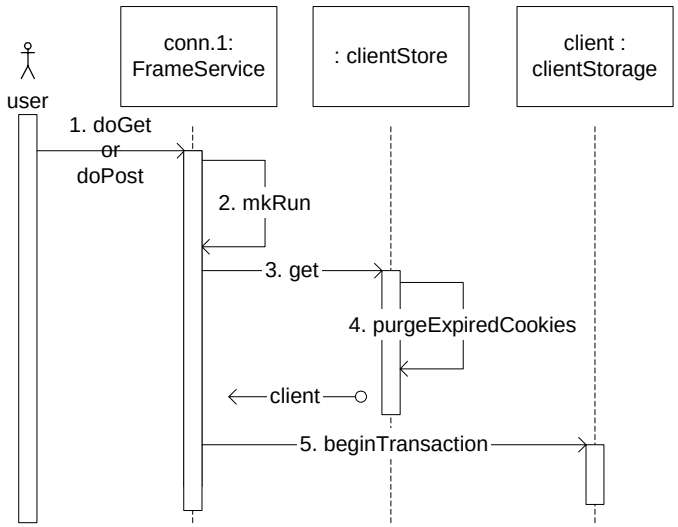
`ServiceSearchBudgetLines.java`

`ServiceTransactionsEdition.java`

`ServiceValidate.java`

`ServiceVrac1.java`

They retrieve the session as shown below:



Most of the time, FrameService.mkRun is called, but the request is always adressed to one of its subclasses. All of the servlets that do not initiate a session must work the same.

Unused sessions are kept for a certain amount of time. When they expire, the are purged at the next`clientStore.get`method call using`clientStore.purgeExpiredCookies`. The delay is given by the parameter COOKIEDELAY of the Webint.ini configuration file. The default value is 1 hour expressed in seconds.

Finally, the following servlets are either stateless or they are never called directly:

`FrameService.java`(never called directly)

`getAmd.java`(stateless)

`putAmd.java`(stateless)

XslParser (stateless)

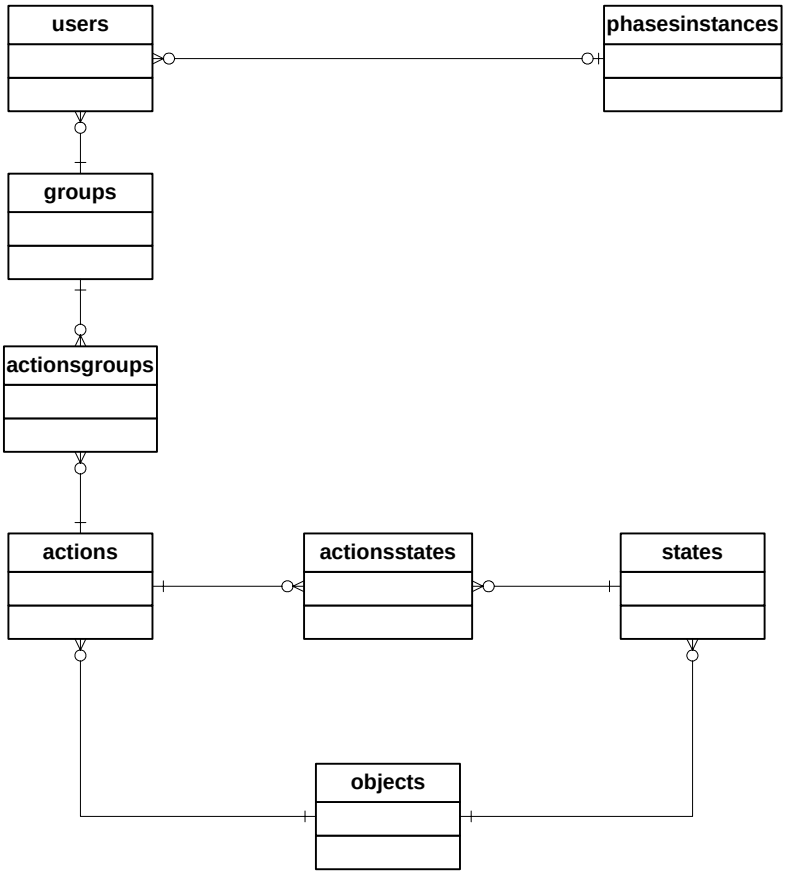
The stateless servlets bypass the session system.

6.9.3. Rights management

Webint does not use a default HTTP access authentication scheme. Instead, a custom scheme is used that manages the user rights in the same way as the other objects of the system.

The current access management distinguishes groups of users that are allowed to access some phases. Moreover for each group, an administrator can decide which actions may be executed by a group. A separated administrators' manual describes the complex interaction from the Webint perspective.

The database objects involved in this process are:



The tables`objects`,`states`,`actions`and`actionsstates`have an historical meaning. Until 2003, the`objects`table reflected all the classes that where actually used in the application server. The`states`,`actions`and`actionsstates`reflected the actual state transition diagrams that defined the permitted behaviour of the objects. The`actionsgroups`table defined therefore all the events that a group was allowed to trigger.

Starting from 2004, these tables have been frozen to a fixed value. A utility program called`datadump`allows to populate the tables from a set of predefined XML files.

Every button of the application is mapped to an`action`entry. The member of a group will see the button if a matching`actionsgroups`entry exists. The Java classes involved in this process are mainly`clientStorage`and`ServiceButtons`.

Moreover, the overall permitted actions per object must be defined in the configuration file called`permissions.ini`.

Only 2 servlets handle the user access authentification:`Consultation`and`ServiceConnection`. The latter is the usual servlet that is called to validate the username and password. It compares the given data with the content of the database. If the authentication succeeds a valid cookie is issued and the user may enter its session. Otherwise, an error message is returned.`Consultation`performs a restricted login as a special user called "guest". This user must nevertheless be correctly configured as a valid user in order to make the`Consultation`servlet work.

6.9.4. SQL Connection Handling

The very first version of Webint did not manage the connections. Whenever a user made an HTTP request to Webint, a new SQL connection was created. Very quickly, two problems appeared:

the creation of a SQL connection is very slow in Oracle;

if too many users accessed the system, Oracle would return an error because the maximum number of connections was reached.

To solve these problems, the classical solution is to use a connection pool. The connection pool:

opens initially a minimum number of connections;

opens more connections when they are needed;

limits the maximum number of open connections;

when the maximum allowed number of open connections is reached, the connection pool puts the request in a waiting queue;

reuses a connection whenever a user closes a request.

Freely available Java connection pooling code has been used to implement this policy. The involved Java classes is`DbConnectionBroker.java`. It does the real pooling, it is simple, yet powerful.

The current implementation of the connection pool tries 10 times to get a connection and waits 2 seconds between each trial in the case it cannot find an available connection.

The connection pooling goes hand in hand with the transaction management. In Webint, a simple scheme has been adopted:

```
1 SQL query = 1 pooled connection = 1 Oracle transaction.
```

This is possible because Webint never updates the SQL database. A transaction never needs to span more than 1 SQL query.

6.9.5. Handling of UTF-8 strings

Webint must support the input and output of multilingual strings potentially containing many accented characters and special characters. For instance, the most frequent special character is the em-dash. This character is extensively used to indicate a zero amount.

As every modern web browser understands the UTF-8 character set, Webint generates exclusively UTF-8 character strings. This has important consequences:

all the generated HTTP responses contain a header that specifies the UFT-8 character set:`Content-type: text/html; charset=UTF-8`; the Webint Java constant`project.norme`contains the used value;

whenever there is an explicit conversion to bytes, the encoding must explicitly be “UTF-8”; the Webint Java constant`project.stringEncoding`contains the value to use.

In the other direction, all the generated form elements contain an attribute that tells the browser to encode the characters using UTF-8:`<FORM NAME="form" METHOD="POST" ACCEPT-CHARSET = "UTF-8">`.

Telling the browser to generate UTF-8 is not enough. The problem is that neither Internet Explorer 6.0 nor Mozilla 7.x specify in their request that they are sending UTF-8 characters. Tomcat, the servlet container hosting the Webint servlets, assumes in such cases that it receives characters in the default encoding. On the Windows platforms it is indeed UTF-8. On the Solaris 8 Sparc platforms, however, it is ISO 8859-1. The most simple solution to this problem would have been to launch Tomcat in a Java virtual machine with a specific encoding “`java -Dfile.encoding=UTF-8`”. Unfortunately, this does not work for Webint because this would change the default encoding to read the different configuration files too.

The problem has finally been resolved by a post-processing of the query parameters done in`FrameService.parseQueryString`: if Tomcat executes on a Sparc architecture, all the parameters are submitted to the FrameService.convertISO88591ToUTF8 method before they are used. Otherwise, no transformation is applied.

6.9.6. Java Applets

All the graphically complex interactions are executed by applets. They can either be executed by a Sun 1.4.x Java Virtual Machine or a Sun 1.5.x Java Virtual Machine. Tests executed in environments that come very close to those that are in use at the European Parliament have shown that the applets work as expected when executed by both Java Virtual Machines.

Most of the time, a URL is passed as parameter to these applets. The URL refers to an XML file. The applets download themselves this XML file and display it. The user interacts with the applet and modifies it. When the user thinks that he has finished working with the applet data, she pushes an HTML button. The button triggers JavaScript code embedded in the HTML. The code queries the applet and sends the result back to the server. The standard dynamic HTML-Java programming technique is used to expose the methods of the applet to the browser. It is enabled by the parameter "`scriptable`" of the servlet call in the HTML page. For instance, here is the HTML code to call the `IntegrationManip` applet:

```
<APPLET CODE = "IntegrationManip.class" CODEBASE = "../applet" archive="seiamd-applets.jar"
HEIGHT = 440 WIDTH = 770 NAME = "orderintegration">
<PARAM NAME = url VALUE = "../servlets/create" >
<PARAM NAME = xml VALUE = "../tmpxml/311.26.43.7.6.9.200406102004194430.xml.gz" >
<PARAM NAME = "institutions" VALUE ="Parliament,Council,EI"
<PARAM NAME = "scriptable" VALUE = "true">
<PARAM NAME = "idsave" VALUE = "197555">
<PARAM NAME = "url2" VALUE = "..">
</APPLET>
```

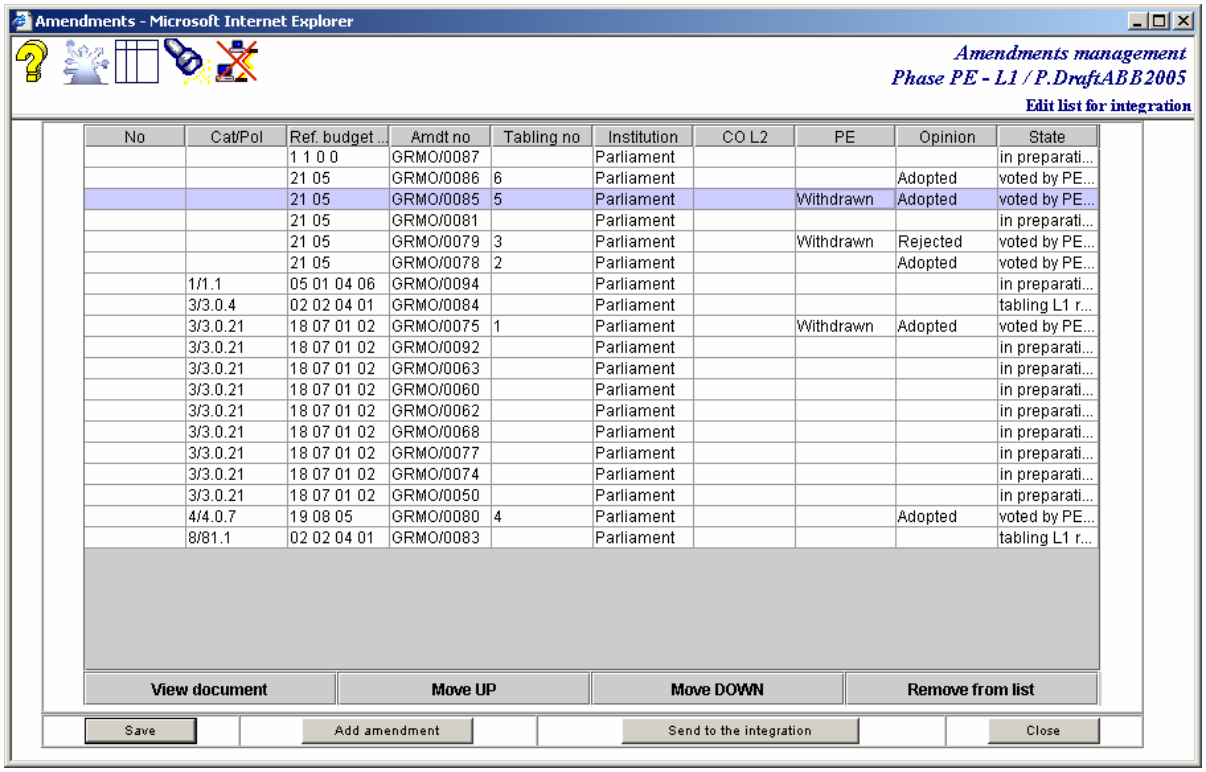
The name of this applet is `orderintegration`. The `xml` parameter contains the URL to the initial data. The following JavaScript function simply calls public Java methods of the applet to build the query string to be sent to the server:

```
function saveValues() {
  document.orderintegration.buildQueryString();
  document.form.select.value=document.orderintegration.getSelect();
```

```
  document.form.ids.value=document.orderintegration.getIds();
  document.form.institutions.value=document.orderintegration.getInstitutions();
}
```

Using introspection, the JavaScript engine locates the right Java applet and calls the methods prefixed by `document.orderintegration` in the previous example.

6.9.6.1. IntegrationManip



This applet allows editing the list of amendments that should be integrated in a budget document. The order of the amendments is important. There are buttons to move the amendments and a button to remove an amendment from the list. The list is a multiple selection list. The selection does not need to be contiguous.

The input is an XML file where each entry has the following format:

```
<SECTION ID="197083" LBALIAS="1 1 0 0" AMDNO="GRMO/0087" STATE="in preparation" TPVOTE=" "
COMM="0.0" PAY="0.0" REV="0.0" AMDCATPOL=" " NOORDER="" DECISION_PE=" " OPINION=" " TABLING="
" DECISION_CO=" " INSTIT="Parliament" URLDOC="../../RT/RT.OR.amd197084.zrf"></SECTION>
```

`ID`: `id_institutionamendment`, identifier of the amendment;

`LBALIAS`: alias of the main budget line modified by the amendment;

`AMDNO`: official name of the amendment;

`STATE`: the state of the amendment in the whole procedure (in preparation is the starting one; this value simply reflects the internal state of the JBoss AmdItem object);

`TPVOTE`: how the decision has been taken: one of the `tpVote*` codes defined in the configuration file `codes.ini`;

`COMM`: the commitments (attached to the main budget line);

PAY: the payments (attached to the main budget line);

REV: the revenues (attached to the main budget line);

AMDCATPOL: the policy code associated to the amendment;

NOORDER: always empty on input, on output, it gives the order of the amendment in the list;

DECISION_PE: the decision of the plenary session of the European Parliament; one of "ADOPTED", "REJECTED" or "WITHDRAWN";

OPINION: the decision of the BUDG committee, one of "ADOPTED", "REJECTED" or "WITHDRAWN";

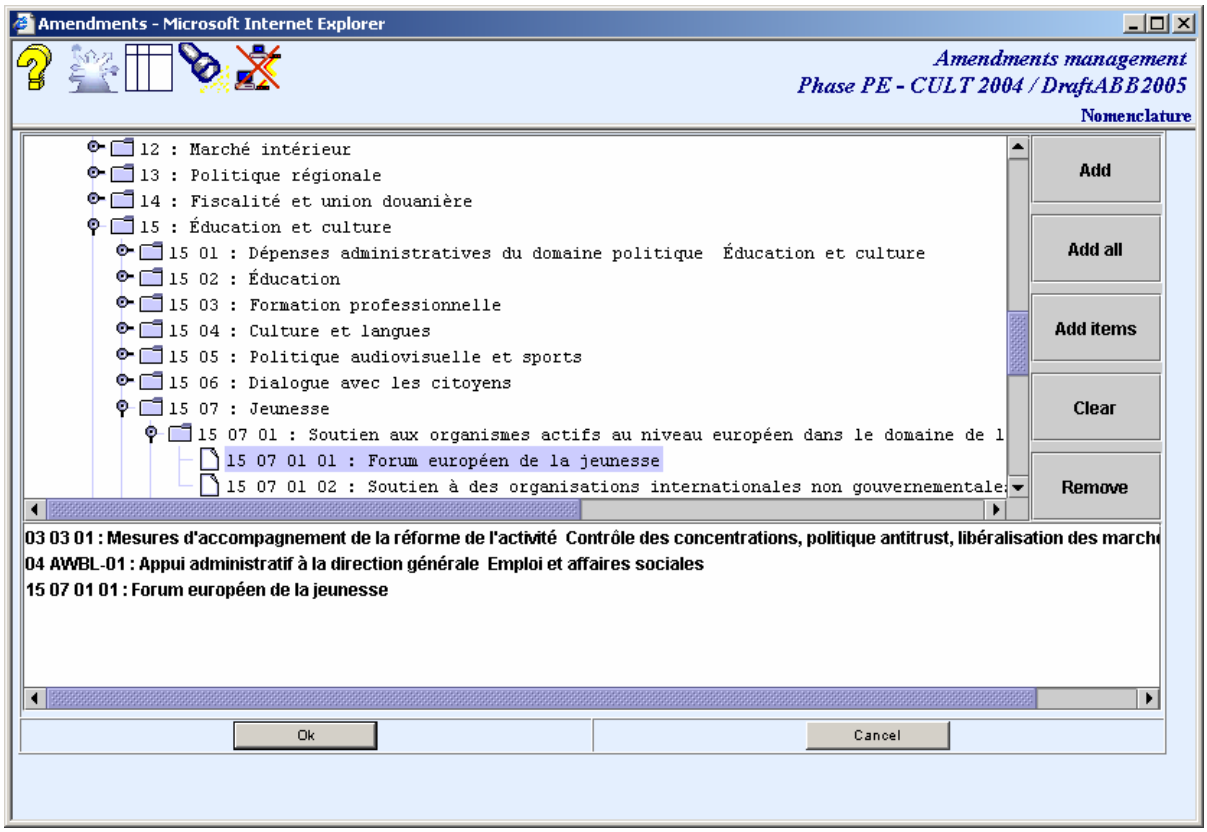
TABLING: the tabling number to the plenary session of the European Parliament;

DECISION_CO: : the decision of the Council; one of "ADOPTED", "REJECTED" or "WITHDRAWN";

INSTIT: the institution currently handling the amendment;

URLDOC: URL of the reduced text associated to the original language of the amendment.

6.9.6.2. NomManip



This applet allows selecting one or more budget lines from the nomenclature of the Budget. It shows the headers and the aliases in a tree-like structure. The selected lines are shown in the lower part of the screen. The order is significant. There are buttons to add one or more lines and there are buttons to suppress some or all the lines from the current selection.

Depending upon the number of parameters, one or two trees are shown. If you specify only the `xml_new` parameter, the applet builds a single tree from the XML file referenced by the URL. If you also specify the `xml_old` and `relations` parameters, the applet builds a double tree view, where left you see the nomenclature referenced by the `xml_old` parameter and right the nomenclature referenced by the `xml_new` parameter. This has been implemented to accommodate on a single screen the traditional nomenclature and the ABB (Activity based Budget) nomenclature. The `relations` parameter describes a mapping between the old and new nomenclature.

The nomenclature files verify a simple DTD.

```
<!DOCTYPE TocEntry [
<!ELEMENT TocEntry (id, alias, title, children-list)>
<!ELEMENT (id|alias|title) (#PCDATA)>
<!ELEMENT children-list (children*)>
<!ELEMENT children (id, alias, title, children-list)>
]>
```

`Id` is the internal identifier of a budget line and `alias` is the well-known official budget line number.

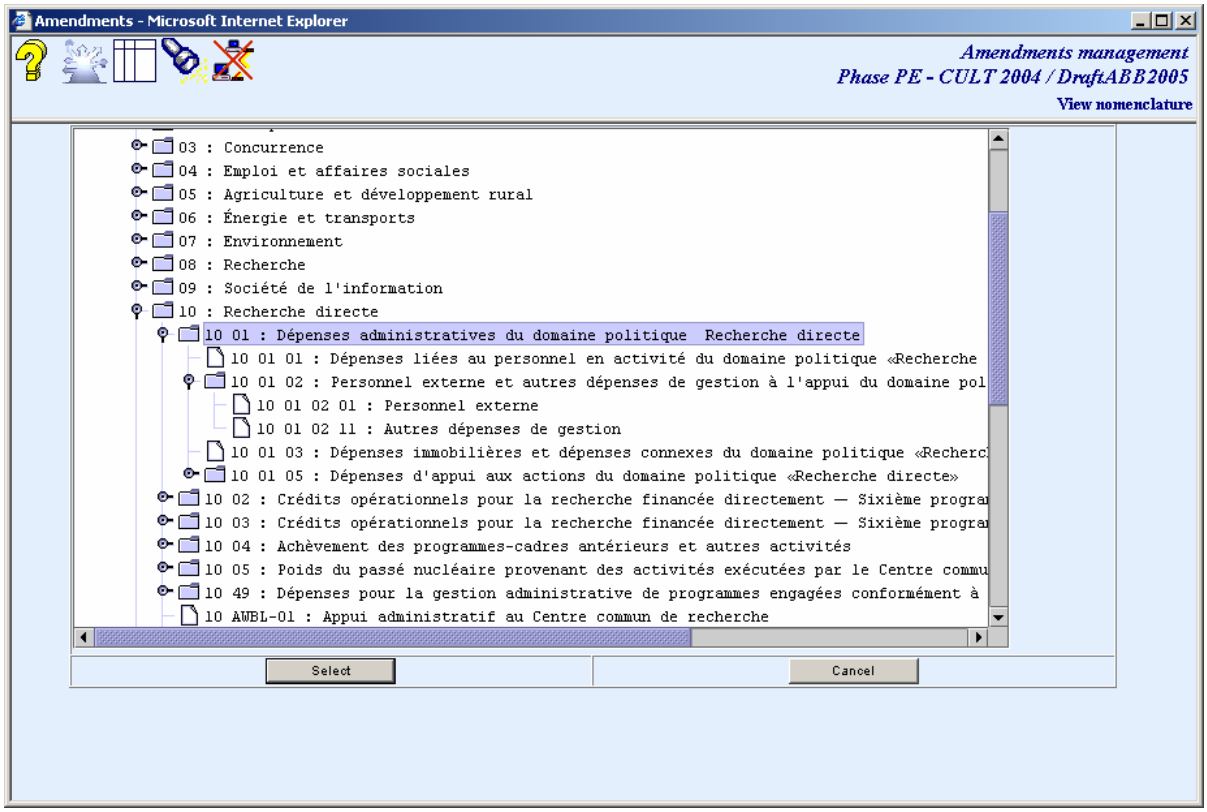
The DTD for the correspondence is also very simple:

```
<!DOCTYPE relations [
<!ELEMENT relations (abb*, bud*)>
<!ELEMENT (abb|bud) #EMPTY>
<!ATTLIST (abb|bud) id CDATA
 aliases CDATA>
]>
```

The `abb` element gives for an alias of the ABB nomenclature (attribute "`id`") the list of corresponding aliases in the traditional nomenclature separated by semicolons in the attribute "`aliases`".

The `bud` element gives for an alias of the traditional nomenclature (attribute "`id`") the list of corresponding aliases in the ABB nomenclature separated by semicolons in the attribute "`aliases`".

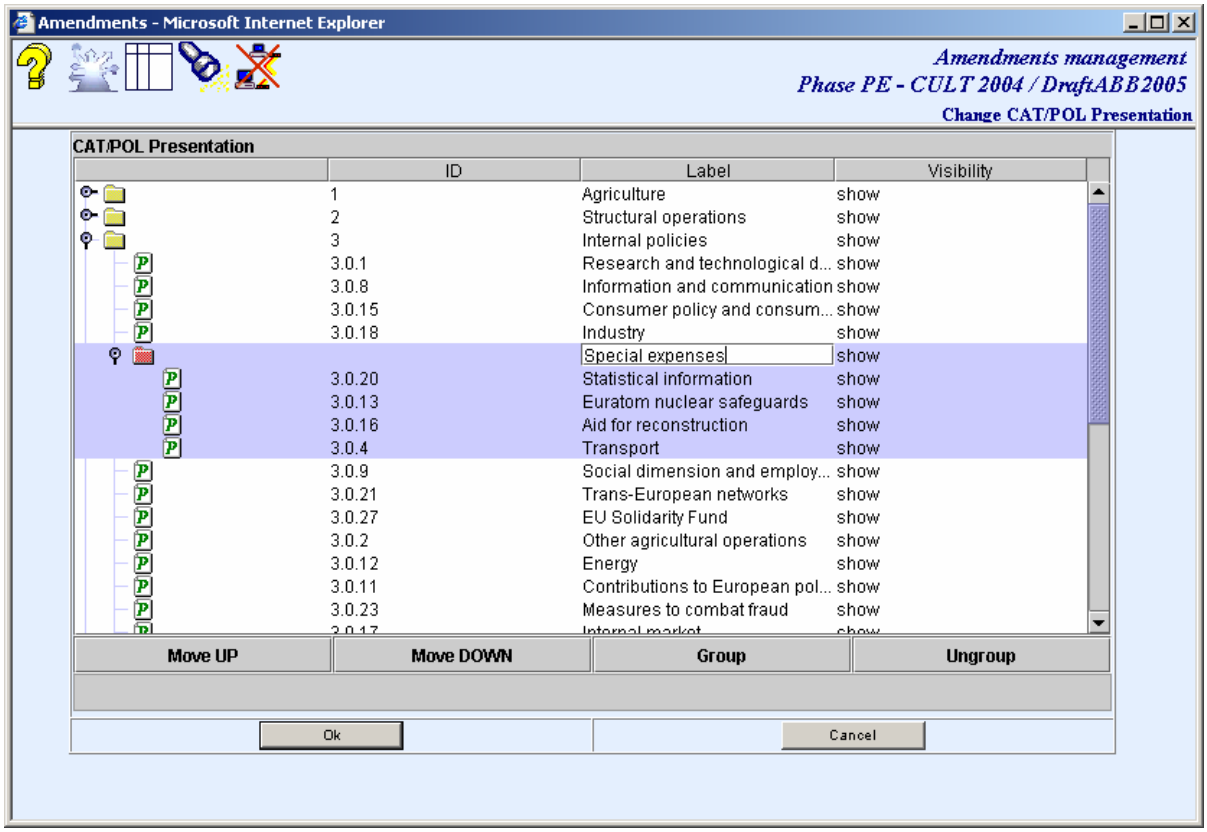
6.9.6.3. NomSelect



`NomSelect` works like `NomManip` except that there is a single parameter "`xml`" and that therefore only the new nomenclature is displayed. Moreover, at most one budget line may be selected at any time.

The format of the XML nomenclature input file is the same as for `NomManip`.

6.9.6.4. PresentManip



Since 2003, all the institutions share a single set of categories and politics codes. Nevertheless some institutions wanted to customize the so-called CAT/POL report in order to accommodate their usual way of looking at the figures of the Budget. This applet aims at grouping some politics and giving them a common title. The groups must not overlap nor include another group.

`PresentManip` offer the possibility of moving several consecutive categories or groups, to group and ungroup and to name the new groups.

From the programmer's perspective, it has been an interesting challenge to combine a Java Swing list view with a Java Swing Tree View. It unveils the expressive power of the Java Swing toolkit.

The format of the XML input file is fairly simple:

```
<!DOCTYPE categoryPresentationStructure [
<!ELEMENT categoryPresentationStructure (category)*>
<!ELEMENT (category) (itemId, label, visibility, entityId, (cluster|policy)*)>
<!ELEMENT (cluster) (itemId, label, visibility, entityId, (policy*))>
<!ELEMENT (itemId|label|visibility|entityId) (#PCDATA)>
]>
```

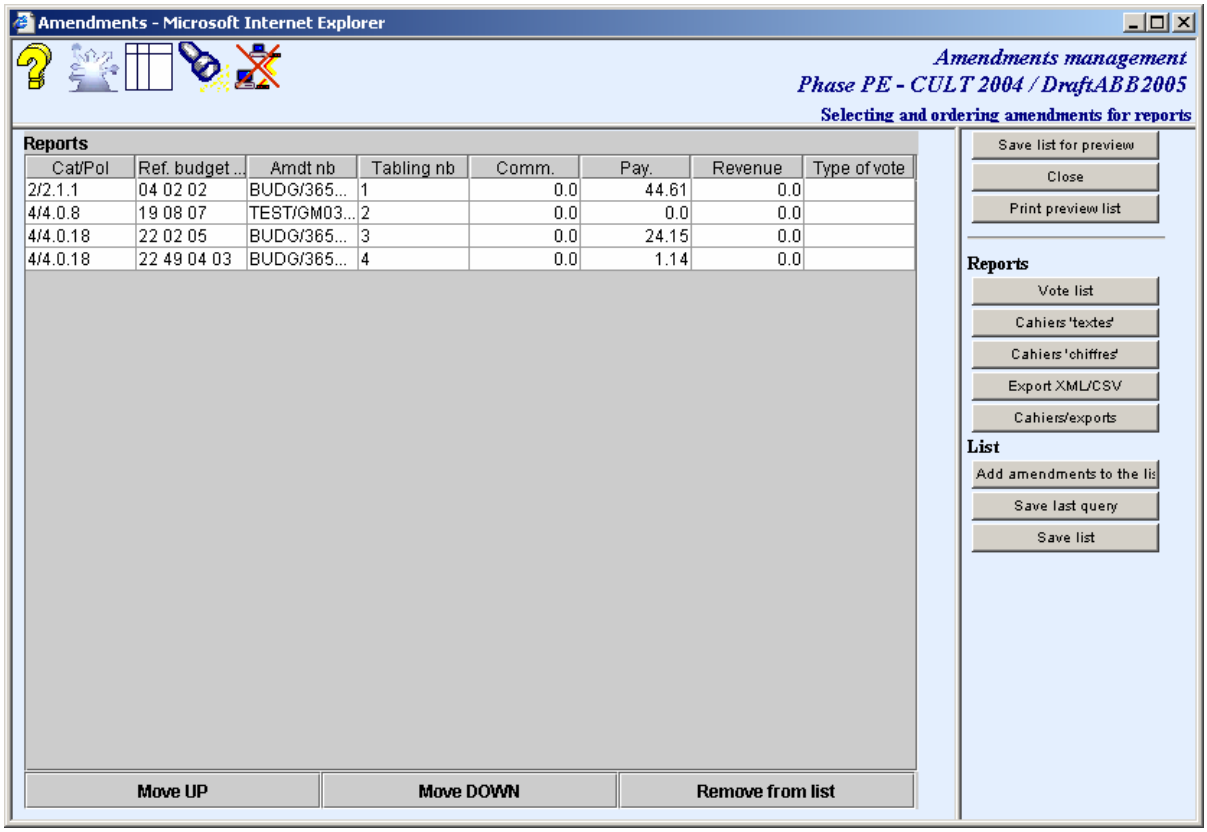
`ItemId` contains the official name of the category policy or cluster (group).

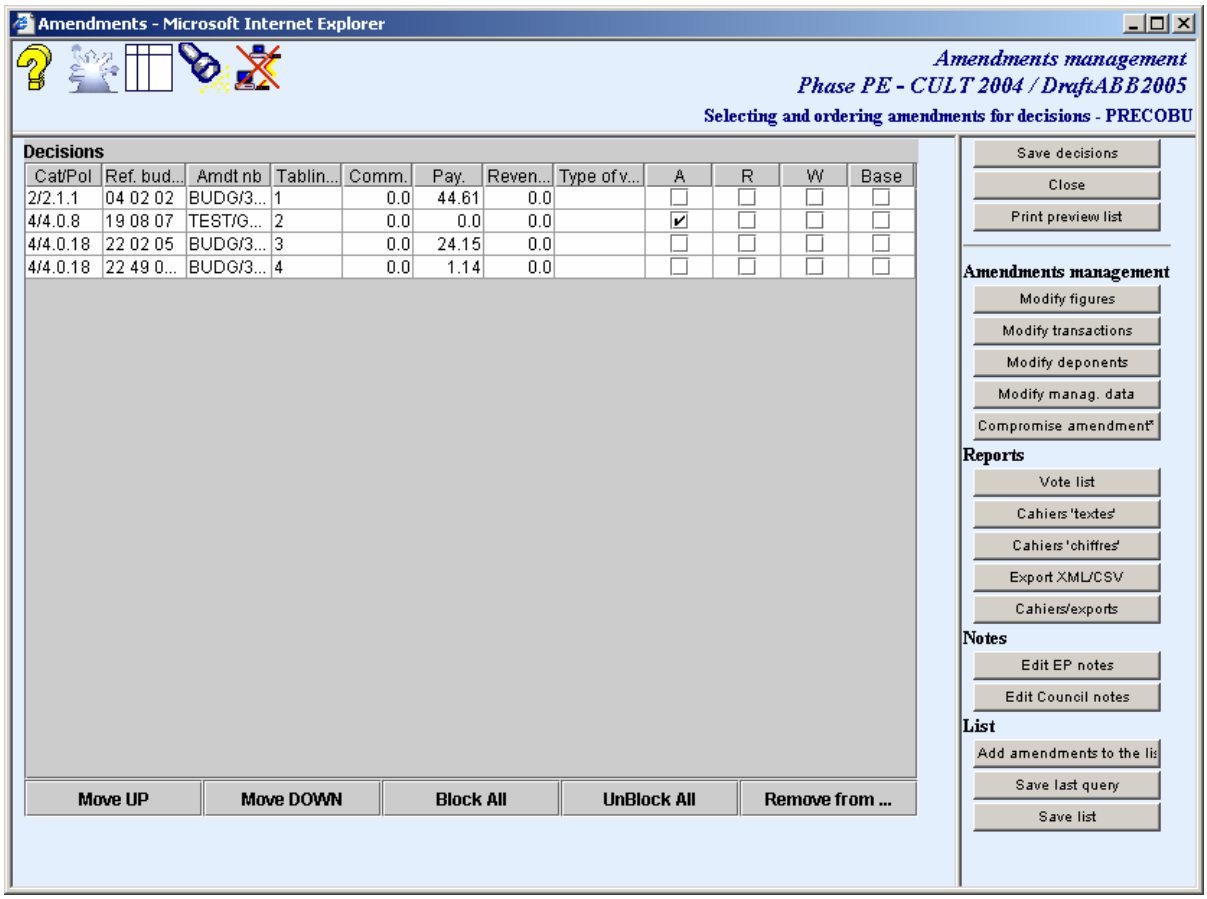
`Label` contains the heading.

`Visibility` contains either "displayed" when the item should be displayed in the CAT/POL report or "hidden" when it must not be shown.

`EntityId` is a number and gives the internal identifier of the category policy or cluster (group).

6.9.6.5. ReportManip





This applet allows editing the list of amendments that should appear in a report. The order of the amendments is important. They will afterwards appear in the same order in the generated report. There are buttons to move the amendments and a button to remove an amendment from the list. The list is a multiple selection list. The selection does not need to be contiguous.

The input is an XML file where each entry has the same basic format as an entry for the `IntegrationManip` applet. However, some additional attributes complete it:

`ACCESS`: 1 if a decision may be taken;

`BASE`: identifier of the base amendment

`BLOCK`: the name of the block containing the amendment; the same decision applies to all amendments of a block;

`FT`: the URL to the full text document associated to the amendment;

`RTDOCUMENT`: the URL to the reduced text document associated to the amendment;

`PE_RTDOCUMENT`: URL to the reduced text document giving the situation of the amendment as approved by the European Parliament;

`STATUS`: the status of the amendment;

`VOLUME`: the volume name containing the main budget line of the amendment;

`IDAMDT`: the `id_amendment` identifier of the amendment;

`ID_INST_AMD`: the `id_institutionamendment` identifier of the amendment;

`MODIF`: for the Council only: true or 1 if the Council wants to modify an amendment;

`SRCID`: `id_institutionamendment` identifier of the CO L2 amendment (for PE L2 amendments only)

`DOC_LOCK`: flag to show whether the original language document associated to the amendment is locked;

`SEL_ROWS`: used to keep the selection from one call to another of the applet.

The same applet is also used in the “Simulation” and “Decisions” screens. In those cases, there are 4 more columns that allow inputting a decision and the vote type (“Decisions” screen only). This applet is also used in the “Council's position on 2d reading” screen. Then, there are even more columns:

`FT` contains a smiley icon to edit the full text of the amendment

`RT` contains a smiley icon to visualize the reduced text of the amendment

`RT PE` contains a smiley icon to visualize the reduced text of the amendment as voted by the Parliament in its first lecture.

`M` replaces the `W` column because the Council can in the second lecture accept, reject or modify an amendment, but not withdraw it. If a Council user checks the box associated to the amendment in that column, she tells the system that she wants to modify the amendment.

The applet knows which screen configuration to show based on three parameters:

“Reports” screen: `QuickReport=”1”; AmdCol2=”false”; Simulation=”false”;`

“Decisions” screen: `QuickReport=”0”; AmdCol2=”false”; Simulation=”false”;`

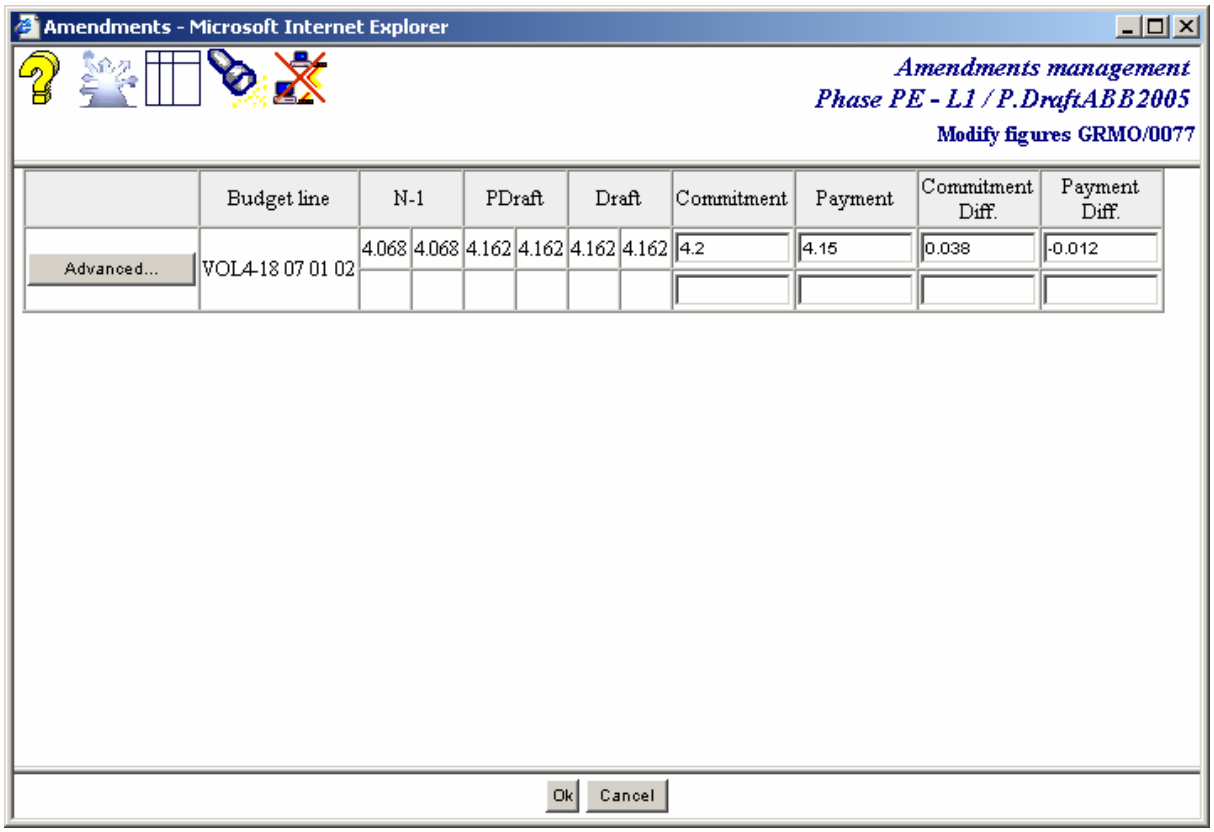
“Simulation” screen: `QuickReport=”0”; AmdCol2=”false”; Simulation=”true”;`

“Council's position on 2d reading” screen: `QuickReport=”0”; AmdCol2=”true”; Simulation=”false”.`

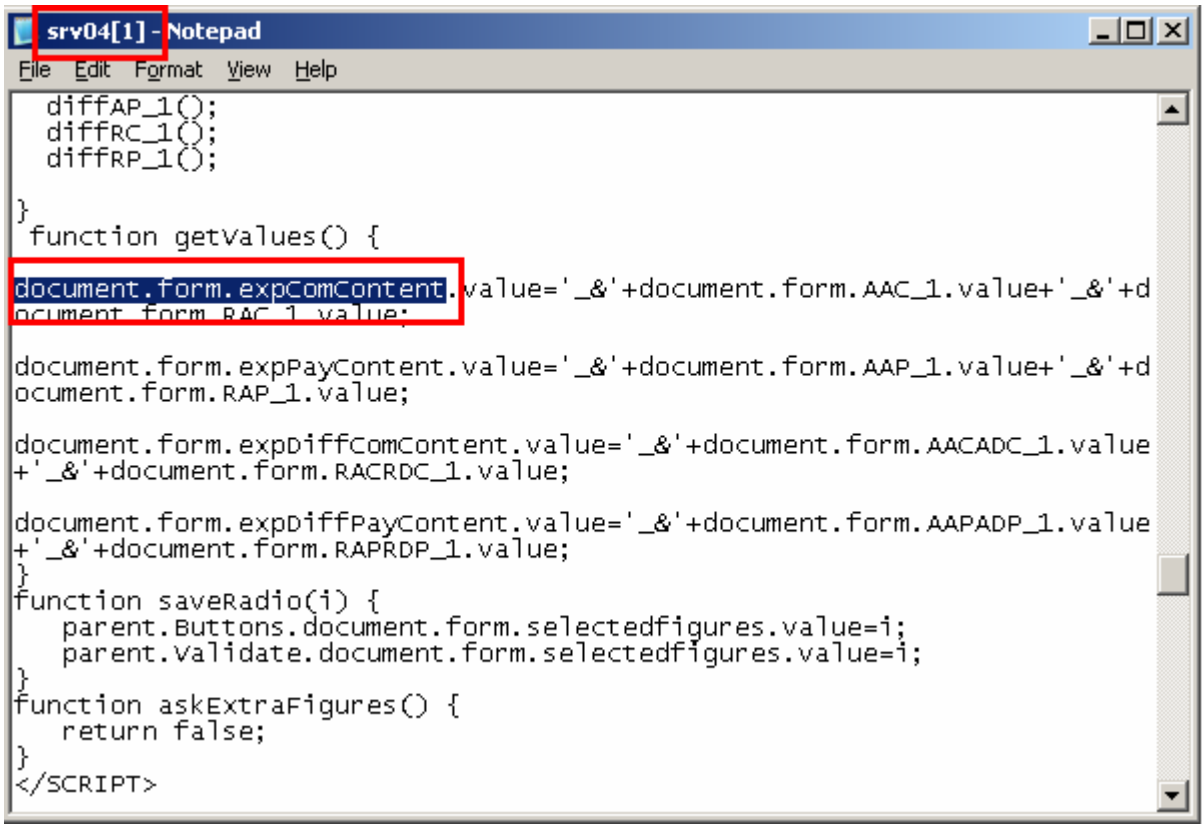
6.9.7. How to retrieve the Java class generating a page

When you want to modify Webint, the most difficult task is to find which object generated a page. Most of the time, you start by locating the page that you want to modify. For instance, suppose that you want to modify the figures edition screen by adding a new type of figures. Of course, it is a very simplified example, because adding a new figure requires a lot of modifications all over the source code. But let us assume that we only want to add a new input field and compute a sum with an existing field.

First, use Webint as a test user and navigate to the page you want to modify:



Next, click inside the frame that interests you and with a right mouse-click, select “View Source”:



The first important piece of information is the name of the applet. If you look at the name of the window showing the source, you see that in our example the source file is called "srv04[1]".`Srv04`is the official name of the applet that has generated the page. The mapping between the official, visible name and the Java class name is defined in the file`dev4.1/webint/web/WEB-INF/web.xml`.

Official, visible servlet name	Java source file inside dev4.1/webint/src/services
amdcol2	ServiceAmdCol2
consultation	Consultation
create	ServiceCreate
csvGenerator	ServiceGenerateCSV
getAmd	getAmd
login	ServiceLogin
putAmd	PutAmd
quickedition.zrf	ServiceQuickEdition
repository	ServiceLoginRepo
searchBudgetLines	ServiceSearchBudgetLines
srv01	ServiceVrac1
srv02	ServiceDrawHeader
srv03	ServiceValidate
srv04	ServiceMainFrame
srv05	ServiceConnection
srv06	ServiceEditTransactions
srv07	ServiceTarnsactionsEdition
srv08	ServiceButtons
srv09	ServiceAmdSearch
srv10	ServiceAmdManag
srv11	ServiceEditTransaction
srv12	ServiceList

Official, visible servlet name	Java source file inside dev4.1/webint/src/services
srv13	ServiceEdition
srv14	FrameService
xslparser	XslParser

The preceding table shows that`ServiceMainFrame`is the servlet that generates the page that we look for. Next, you look for a piece of generated code that might be as near as possible to the place we want to modify. “document.form.expComContent” seems to be a good candidate because you want to add the content of a new field to this value.

A simple search makes you find that the JavaScript is generated in the method`ServiceMainFrame.drawScripts`.

Moreover,`drawScripts`is a private method. Another quick search reveals you that it is only called in`ServiceMainFrame.drawmodifyFiguresOfTransaction`and`ServiceMainFrame.drawModifyFigures`. Now, you know that these are exactly the methods that generate the input fields you look for.

Most of the time, you may also want to add a hidden input field to make the new value available to the servlet when you click on a button. Ninety-nine percent of the time you will modify the servlet`ServiceValidate`.

This was a toy example, but most of the cases are not harder than this one. It gets only complicated if you must add a new screen. In that case, try to first add a button to call the new screen (`ServiceButton`or`ServiceVrac1`), then to generate the right frame (`ServiceCreate`) and finally to create your specific code by inspiring yourself from an existing one.

Another frequent modification is when you want to change the handling of a button. This case is similar to the previous ones:

locate the button you are interested in;

open in your web browser the generated HTML file;

locate the name of the button and the servlet that is called;

the name of the servlet is most of the time explicitly affected to the form in a JavaScript function;

by convention, all HTML forms generated in Webint convey the name of the button in a hidden field called "`callbutton`"; therefore, you only have to spot the JavaScript code that affects a value to`callbutton`;

in the found servlet, search for the button name you found; this gives you a very good approximation of the position of the code that you look for;

follow the method calls.

6.9.8. Configuration files

The description of the configuration files uses several symbolic names:

`$TOMCAT_HOME`is the installation directory of the Jakarta-Tomcat servlet container;

`$RUN`is the main SEI-BUD execution directory.

6.9.8.1. apps-seiamd.xml

Location:`$TOMCAT_HOME/conf`

This is a Jakarta-Tomcat configuration file. The format is available at http://jakarta.apache.org/tomcat/tomcat-3.3-doc/index.html.

It defines the static directories that must be published by Tomcat to make Webint work properly:

`/report`: the directory containing the generated reports;

`/xml`: the directory containing the static nomenclature files to be shown by the different nomenclature applets; there must be one nomenclature file per budget publication and per language;

`/align-data`: the directory containing the alignment files generated by SEI-BUD; they are the basis for the TWB translation memories.

6.9.8.2. code.ini

Location:`$TOMCAT_HOME/webapps/WebintServlet/Ini`

The format of this XML file is

```
<!DOCTYPE SECTION [
<!ELEMENT SECTION (ITEM*)>
<!ATTLIST SECTION ID CDATA>
<!ELEMENT ITEM (#PCDATA)>
<!ATTLIST ITEM ID CDATA>
]>
```

The`ID`attribute of the`SECTION`element must be "`MAIN`".

Every`ITEM`element references a code. This code has not been hard-coded in Webint because it is an external, well-known code. It is at least shared with the Repository Manager.

Some items contain multiple values separated by vertical bars. For instance,`pmApproximation`enumerates all the permitted approximations of “p.m.” (pro memoria).

This file should be modified with extreme care. Modifying a value has an immediate consequence on the Repository Manager. Modifying an`ID`attribute has an immediate consequence on Webint and the Repository Manager.

6.9.8.3. color.ini

Location:`$TOMCAT_HOME/webapps/WebintServlet/Ini`

The format of this SGML file is

```
<!DOCTYPE INIFILE [
<!ELEMENT INIFILE 0 0 (SECTION+)>
<!ELEMENT SECTION – 0 #EMPTY>
<!ATTLIST SECTION ID CDATA
NAME CDATA
ERROR CDATA
BGROUND CDATA
TABLETITLEBG CDATA
WARNING CDATA
TEXT CDATA
```

```
TITLECOLOR CDATA
SUBTITLECOLOR CDATA
EXTRABGCOLOR CDATA>
]>
```

This file describes the colours used by the application. Every SECTION element defines a new configuration. To choose a configuration, Webint examines the CONNECTION entry in webint.ini. It chooses the SECTION element whose NAME attribute equals the CONNECTION entry. If Webint cannot locate it, it chooses the first SECTION element.

The meaning of the attributes is:

ID: internal identifier of the colour configuration;

NAME: name to compare with the connection entry found in webint.ini;

ERROR: HTML colour name of all the texts displaying an error message;

BGROUND: HTML colour for the usual background of the application;

TABLETITLEBG: HTML colour for the title area of screen (upper frame);

WARNING: HTML colour for the warning texts;

TEXT: HTML colour for the usual text;

TITLECOLOR: HTML colour for the titles appearing in the title area of the screen (upper frame);

SUBTITLECOLOR: HTML colour for the subtitles appearing in the title area of the screen (upper frame);

EXTRABGCOLOR: special HTML colour of the figures table background; in 2003, it was used to signal that the figures reference the old member states instead of referencing the old and new member states.

6.9.8.4. EN.ini and FR.ini

Location: $TOMCAT_HOME/webapps/WebintServlet/Ini

The format of this SGML file is

```
<!DOCTYPE INIFILE [
<!ELEMENT INIFILE 0 0 (SECTION+)>
<!ELEMENT SECTION – 0 #EMPTY>
<!ATTLIST SECTION ID CDATA
LABEL CDATA>
]>
```

Webint was initially designed to be a multilingual application. Every label should be available in all languages of the European Union. Resource files should keep the text strings. The only existing files are named after the English and French language codes. More languages could follow.

For the moment, both files are up to date, but only the English one is actually used. Whenever in Webint you make a call to client.scrLang(…), you pass an identifier. All ID attributes are checked against it. If there is a match, the corresponding LABEL attribute is returned.

The aforementioned method could quite easily be changed to make a lookup based on the language of the user and to retrieve the right language string but this is not the case for the moment.

6.9.8.5. error.ini

Location:`$TOMCAT_HOME/webapps/WebintServlet/Ini`

The format of this SGML file is

```
<!DOCTYPE INIFILE [
<!ELEMENT INIFILE 0 0 (SECTION+)>
<!ELEMENT SECTION – 0 #EMPTY>
<!ATTLIST SECTION ID NUMBER
LB CDATA>
]>
```

This file groups all the error messages. The`ID`attribute is the identifier and`LB`is the label.

In Webint baseclasse.Error exposes an interface to associate the identifier to the label. If Webint should one day become multilingual in its interface, this class must be extended to retrieve a label based on a language.

6.9.8.6. mailing.properties

Location:`$RUN/conf`

This is a Java readable property file. It specifies the recipients to whom the late translations notification program should send the emails. The key must be of the form

```
IN.recipients.to.LG
```

where`IN`is an institution [`PE`(Parliament),`CO`(Council) or`COM`(Commission)] and LG a valid language code.

Several recipients may be specified for one key. They are separated by blanks.

6.9.8.7. mailsender.properties

Location:`$RUN/conf`

This is a Java readable property file. It specifies the configuration parameters for the late translations notification program

Key	**Description**
mail.smtp.host	SMTP mail host
mail.mesg.from	Valid sender address for the e-mail notification
mail.mesg.subject	Subject of the notification. The pseudo-tags`<DATE>`or`<LANG>`may be used as placeholder for the current date and for the language division to which the mail is sent.
mail.mesg.part1.title	Title for the list of overdue translations
mail.mesg.part2.title	Title for the list of translations that will soon be overdue
mail.mesg.footNote	The footer of the e-mail or`<NONE>`if no footer is needed
mail.mesg.headNote	The header of the e-mail or`<NONE>`if no header is needed
database.url	Oracle JDBC connection string
database.login	Oracle user ID
database.password	Oracle user password
database.driver	Oracle JDBC drivers to use (usually`oracle.jdbc.driver.OracleDriver`)
debug.active	`True`to debug or`false`otherwise
output.mode	`Trace`to trace or`None`if no tracing is required.
output.stdout	Full path to the standard output trace file
output.stderr	Full path to the standard error trace file
parameter.delay	Delay in days to notify about translations that will be soon overdue. For instance, if you specify “1”, you will get all the translations that expire the day where the notification is built.

labels.amdNumber	Label for “Amendment number” column
labels.targetLanguage	Label for “Target language” column
labels.baseLanguage	Label for “Base language” column
labels.mixedText	Label for “Mixed text” column
labels.institutionAmendment	Label for “Institution amendment” column
labels.translationStartDate	Label for “Translation start date” column
labels.translationEndDate	Label for “Translation end date” column
labels.deadline	Label for “Deadline” column
labels.depot_number	Label for “Depot/plenary number” column
labels.plenary_number	Label for “Plenary number” column
labels.fdr	Label for “FdR” column
RelayLanguages	must be consistent with the rmg-config.xml (TranslationByRelay) Gives the list of relay languages separated by ‘,’ Examples: RelayLanguages=DE;EN;FR

6.9.8.8. nodetype.ini

Location:`$TOMCAT_HOME/webapps/WebintServlet/Ini`

The format of this SGML file is

```
<!DOCTYPE INIFILE [
<!ELEMENT INIFILE 0 0 (SECTION+)>
<!ELEMENT SECTION – 0 #EMPTY>
<!ATTLIST SECTION ID CDATA
LB CDATA>
]>
```

When you want to create a new amendment in the “advanced transactions” creation screen, you need to know which elements of the Budget DTD may be created inside another element. This file makes the mapping. For each element specified in the`ID`attribute, it gives in the`LB`attribute the comma separated list of elements that it may contain according to the DTD.

Modify this file only if there is a change in the Budget DTD!

6.9.8.9. permission.ini

Location:`$TOMCAT_HOME/webapps/WebintServlet/Ini`

The format of this SGML file is

```
<!DOCTYPE INIFILE [
<!ELEMENT INIFILE 0 0 (SECTION+)>
<!ELEMENT SECTION – 0 #EMPTY>
<!ATTLIST SECTION CLASS CDATA
ACTION CDATA>
]>
```

CLASS contains the name of a class as defined in the`OBJECTS`database table and`ACTION`associates a comma-separated list of permitted actions on that class as defined in the ACTIONS database table. This file acts as a filter. Only the actions specified here may enable a button in Webint. Therefore, in order to display a button guarded by an (object, action) pair, all of the following conditions must be met:

the group of the user must have that action enabled in the “Edit group” screen;

the action must be present in the`ACTION`attribute of the related element; the lookup is done by using the object name as a key and by comparing it to the`CLASS`attribute.

See the “Rights management” section for a complete description of how the users may get the authorisation to use the functions of Webint.

6.9.8.10.srvamdcol2.properties

Location:`$TOMCAT_HOME/webapps/WebintServlet/Ini`

This is a Java readable property file. It is used in the handling of the “Council’s position on 2d reading” screen to avoid a strong coupling of Java classes.

6.9.8.11.web.xml

Location:`$TOMCAT_HOME/webapps/WEB-INF/`

This is the Webint servlet configuration file as required by the J2EE 2.2 specifications.

6.9.8.12.webint.ini

Location:`$TOMCAT_HOME/webapps/WebintServlet/Ini`

The format of this SGML file is

```
<!DOCTYPE INIFILE [
<!ELEMENT INIFILE O O (SECTION, SECTION)>
<!ELEMENT SECTION - - (ITEM*)>
<!ATTLIST SECTION ID CDATA>
<!ELEMENT ITEM (#PCDATA)>
<!ATTLIST ITEM ID CDATA>
]>
```

This is the most important configuration file and the most frequently modified. There must be exactly 2 SECTION elements in it. One of it must have the ID "MAIN" and the other the ID "DECISIONS".

The “DECISIONS” section contains, for each institution or entity inside an institution, the name of the field in the AMENDEMENTS database table that will store its decision.

The MAIN section contains the following items:

Item	**Description**
DATABASE	JDBC connection string to the Oracle database
OCI	1 if OCI JDBC drivers have to be used or 0 if the thin drivers have to be used
SERVER	Not used any longer
PORT	Not used any longer
ID	Oracle database login name
PASSWORD	Oracle password
COOKIEDELAY	Maximum age of an unused session expressed in seconds
DEBUGGING	1 to enable debugging, 0 to disable it
ROOT	Not used any longer
LOGDIR	Name of the directory relative to`$TOMCAT_HOME/webapps/WebintServlet/`where the log file should be stored
PROJECTLANGUAGE	`EN`, default language to retrieve the labels
SQLLANGUAGE	`EN`, default language to retrieve some labels in the database
LEADINGCOMMITTEE	Internal identifier to the BUDG committee. It must match the CODICT ID_BODY entry.
TEXTSIZE	Default size of the displayed text (partially used)
JAVAMIMNAME	Not used any longer

MIMINI	Not used any longer
MIMORBNAME	Not used any longer
GREEKSYMBOL	The Greek language identifier, normally “EL”
DTT_MASK	The Oracle long date mask; for instance "DD-MM-YYYY HH24:MI:SS”
DT_MASK	The Oracle short date mask; for instance DD-MM-YYYY”
SCREENHEIGHT	The default screen height
SCREENWIDTH	The default screen width
HTMLENCODING	The character set to use in the HTML pages. The default value is "UTF-8”. Do not modify this value unless you have a valid reason
JAVAENCODING	The character set to use in the Java byte conversions. The default value is "UTF-8”. Do not modify this value unless you have a valid reason
SMILE	The icon in the`$TOMCAT_HOME/webapps/WebintServlet/icons`directory that should be used when a document is available
FROWN	The icon in the`$TOMCAT_HOME/webapps/WebintServlet/icons`directory that should be used when a document is available
TPREADING	One of`L1`or`L2`. The current reading.
CONNECTION	The name of the connection that this Webint serves. Usually one of`Test`or`Production`
FTPROTECTED	1
MINCONNS	Number of initial Oracle connections to open at start up
MAXCONNS	Maximum number of open connections
MAXCONNDAYS	Maximum age of an Oracle database connection. After this delay, the connection is closed.
ORIGINALLANGUAGE	Language identifier for the original (pseudo) language. Usually`OR`
AMENDMENTSYEAR	Budget year. For instance, in 2004, we prepare the 2005 Budget and therefore this value must be 2005
AMDHOMEURL	Root directory where are located all the static files that should be served by Tomcat
BACKUPDIR	Absolute path to the directory where the files are backed up whenever Webint receives them back from a client.

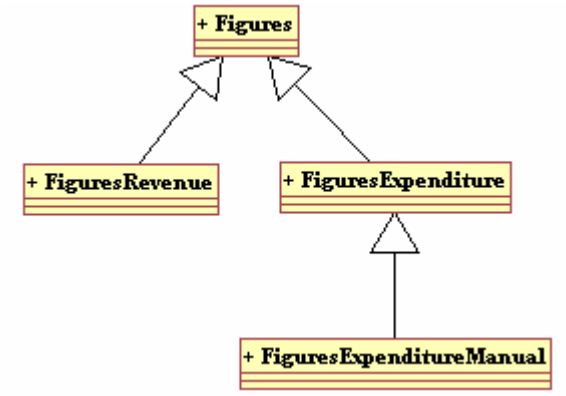
6.9.9. Important Java classes and their usage

Class name	**Description**
Main	This servlet is only called at start-up to initialise all the data that only needs to be loaded once. Among other things, it initialises the project class that is solely composed of static members
baseclasse.Amendment	It collects all the pieces of information about an amendment. The transactions and figures are not stored inside it.
baseclasse.AmendmentReport	It collects all the pieces of information about an amendment when it is aimed at generating an amendment.
baseclasse.Figures and its descendants	It contains the management data about figures. The figures itself are stored in the different subclasses depending of their type.
baseclasse.FiguresColumn	It contains a set of expenditure figures (commitments, payments and the reserves).
baseclasse.MimMess	Not used any longer.
baseclasse.Permission	This class handles builds the interface to manage the rights of an user.
baseclasse.ProgHtml	Most of the Webint-specific code is stored in this class
baseclasse.ProgQuery	Almost all the SQL queries are stored in this class.
baseclasse.project	This class contains only static members. They are loaded at start-up time by the Main class.
baseclasse.TrVector	This is a vector of transactions. Normally all the transactions associated to an amendment are grouped in such a structure.
baseclasse.User	It collects all the pieces of information about an user.
connection.AppServerConnection	The class is the sole interface to the Repository Manager
connection.DBConnectionBroker	This class implements in Webint the connections pool.
gestfifo.clientStorage	All the pieces of information that must be kept for the duration of a session are stored here. Moreover, a semaphore ensures that at most

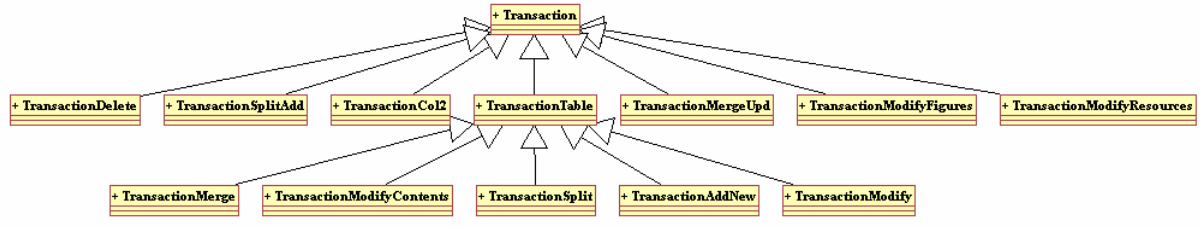
	one request per session executes concurrently. Finally the SQL query to list all the amendments verifying a criterion is build here too.
gestfifo.clientStore	This is the pool of client objects. After an inactivity delay, unused objects are automatically flushed at the next request.
gestfifo.SessionCookie	Class of the "AMDSESSION" cookie used in Webint to keep the session coherence.
handlers.*	This experimental package manages the “Council’ position on 2d reading” screen. The binding between the handles, renderers and widgets packages is only done at execution time.
renderers.*	This experimental package generates the HTML code for the “Council’ position on 2d reading” screen. The binding between the handles, renderers and widgets packages is only done at execution time.
services.FrameService.java	This servlet is the ancestror of all the other servlets. It is never called directly, but some of its methods are used extensively. If offers some basic functions: decoding of the url-encoded input stream, basic handling of a client request, update of the clientStorage members. All the requests for the Repository Manager are build in this class too.
services.GetAmd.java	This important servlet manages the retrieval of amendment documents in RT, FT or TT format from the Repository Manager. It bypasses the session handling system of the usual servlets.
services.PutAmd.java	This important servlet manages the update of amendment documents in FT or TT format coming from the clients. The documents are forwarded to the Repository Manager. It bypasses the session handling system of the usual servlets.
services.ServiceAmdCol2.java	Servlet dedicated to the “Council’s Position on 2d lecture” screen.
services.ServiceAmdManag.java	This servlet mainly handles all the requests to modify the management data associated to an amendment: transactions, decisions, other pieces of information like the tabling number.
services.ServiceAmdSearch.java	This servlet is dedicated to the production of the query screens that ask to enter some criteria to select amendments.
services.ServiceButtons.java	This servlet generates all the button panes shown at the right of the usual lists of amendments and at the right of the list of amendments for the reports and for the decisions.
services.ServiceConnection.java	This servlet handles the authentication of the users at login time. It look into the database to see if the supplied password matches the stored one.
services.ServiceCreate.java	This servlet creates all the frames themselves. Therefore, most of the buttons call this servlet because it is able to rebuild completely the screen.
services.ServiceDrawHeader.java	This servlet is responsible for the upper frame with the icons and the titles that vary according to the asked screens.
services.ServiceEdition.java	This servlet mainly handles the generation of the screens that edit the Parliament, Council and translation notes. Some minor screens like the translation statistics come from here too.
services.ServiceEditTransaction.java	Like the servlets`ServiceEditTransactions`and`ServiceTransactionsEdition`, this servlet manages the simple and advanced transaction edition screens. If you do not find a method here, look at`ServiceEditTransactions`or`ServiceTransactionsEdition`.
services.ServiceEditTransactions.java	Like the servlets`ServiceEditTransaction`and`ServiceTransactionsEdition`, this servlet manages the simple and advanced transaction edition screens. If you do not find a method here, look at`ServiceEditTransaction`or`ServiceTransacti`

	`onsEdition`.
services.ServiceExit.java	This servlet is not used any longer.
services.ServiceGenerateCSV.java	This servlet is dedicated to the generation of the CSV file enumerating all the budget lines that have not been modified by an amendment.
services.ServiceList.java	This servlet generates the three HTML amendment lists: the usual authoring list the detailed authoring list the translation details list
services.ServiceLogin.java	This servlet generates the usal login screen.
services.ServiceLoginRepo.java	This servlet is not used any longer.
services.ServiceMainFrame.java	Most of the time, it generates the HTML frames located at the middle of the screen. The figures and some specific amendment lists are generated here?
services.ServiceQuickEdition.java	This servlet is specialised in retrieving all the amendment documents in RT, FT and TT documents when a user clicks on a smiley icon. Moreover, it also handles the quick edition of document fragments at the creation time of an amendment. The quick edition works only on modify transactions and on justifications.
services.ServiceSearchBudgetLines.java	This servlet is dedicated to the retrieval of all the budget lines that have not been modified by an amendment.
services.ServiceTransactionsEdition.java	Like the servlets`ServiceEditTransaction`and`ServiceEditTransactions`, this servlet manages the simple and advanced transaction edition screens. If you do not find a method here, look at`ServiceEditTransaction`or`ServiceEditTransactions`
services.ServiceValidate.java	Most of the HTML pages at the bottom of the screens are generated here. This pages not only contain the`OK`and`Cancel`buttons, they also generate JavaScript code that gathers the data from the other frames.
services.ServiceVrac1.java	Among other things, you will find here the main screen with the buttons to access the other screens.
services.XslParser.java	This servlet does server-side parsing of XML documents with XSLT stylesheets to generate the print preview lists either in HTML or RTF. The print preview is useful for all the lists of amendments: usual list of amendments; reports; decisions; simulation.
transaction.*	This package stores all the pieces of information about the transactions.
webintutil.Attachment	This utility class allows to build MIME formatted messages with attachments.
webintutil.SqlQuery	This utility class offers a user-friendly interface to make SQL queries.
webintutil.tool	Miscellaneous static methods to handle strings and dates are stored here.
webintutil.SmtpServer	This class handles the sending of e-mails to an SMTP server.
webintutil.sort.*	This package offers a simple interface to sort complex structures. It is mainly used to build up the initial order of the amendments for the reports. For instance, they may either be sorted by category and policy or by category and main budget line alias.
webintutil.xml.*	This package contains utility classes to generate and read XML files.
widgets.*	This experimental package offers the HTML basic elements for the “Council’ position on 2d reading” screen. The binding between the handles, renderers and widgets packages is only done at execution time.

The hierarchy of related figures classes:



The hierarchy of related transaction classes:



6.9.10. Late translations notification

This standalone program is located in the`lateTranslationsMailSender.jar`Java library and it is associated to the Unix script file`latetranlations.sh`.`mailing.properties`and`mailsender.properties`are its configuration files.

The structure of the program is very simple. It makes en SQL query to find all the translations that are overdue. It then groups them by language division and by deadline (already passed or still to come). An HTML formatted file is generated per language and it is sent by e-mail using an SMTP server.

This program is not a daemon. It must be called periodically with a utility program like CRON on Unix machines. A calling frequency of more than 1 time per day is useless as all the computations are done on a daily basis.

6.10. Other client functions

6.10.1. Printing out differences

6.10.1.1.Printout of differences functions

A differential (Printout of Differences) request is sent by the user of the control tool when he wants to view in a "differential" the changes made to a document fragment.

A "differential" is a file in html format that is used to view these modifications.

The SEI-BUD system offers three types of differential: "Author", "Translator" and "Corrector". Every user has access to the type that is most useful to him:

Authors can use them to check the changes that have actually been updated;

Translators can use them to find text for translation more easily;

Correctors can use them to find text for correction more easily.

The three types of differential are accessible for all language versions. To fulfil its role, the "Author" version must be used to check the modified version.

Please refer to the sub-section "The different types of differential" in the section "Producing the "differential" (change-marked text)" or to the user guide for the control tool for more details on the different types of differential.

Please refer also to the sub-section "The differential production process" in the section "Producing the "differential" (change-marked text)" for the technical details.

6.10.2. The Word draft

"Word draft" format is an RTF consultation format used as a proof for the printed version of a publication.

In the case of ABB publication, Word draft format contains a summary table of the budgetary figures in each hierarchy level of the structure; this format also resolves references to other parts of the publication ("reuse-links").

For Volume 0 publication, Word draft format is the same as RTF authoring format, except for the fact that the links ("reuse-link") are resolved.

ABB conversion to "RTF draft" is implemented by the class "`Abb2rtfRO`", which derives from the class "`XSLTFilter`", which is part of the Xef framework. This class is similar to the class "`Abb2rtf`" described in the section on "repo" to RTF conversion, but it runs one extra filter:

`Abb2recap`is a filter that derives from "`XSLTFilter`"; it is made up of three cascading style sheets:

the style sheet "`std-fig.xslt`" removes the formatting from the figures and multiplies them by 100 so it can make calculations using these figures;

the style sheet "`addup-fig.xslt`" calculates the various totals, taking account of the wide variety of cases possible;

the style sheet "`abb2recap.xslt`" formats the totals into summary tables in CALS format; this style sheet includes several other associated style sheets:

`abb2recap-title.xslt`produces the titles of the summary tables

`abb2recap-chapter.xslt`produces the summaries for chapters

`abb2recap-subitem.xslt`handles sub-items

`abb2recap-hr.xslt`handles summaries for human resources

`abb2prs-uti.xslt`handles column and note specifications

`stdtxt-templates.xslt`contains the definitions of standard character strings according to the language of the document;

"abb2wrx" converts "repo" format into the intermediate presentation format "wrx" in preparation for a further conversion to RTF;

"utf82rtf" performs a code conversion from UTF8 to a representation specific to RTF format (using the construction "?);

use of the style sheet "xml2rtf" to convert the presentation format "wrx" into RTF.

7. INTERFACES WITH OTHER SYSTEMS: META-TRANSACTIONS

7.1. Introduction

The Metatrans DTD formalises "high-level transactions" as a mechanism of exchange of information between the SEI-BUD system and other systems such as the SEI-BUD Amendments system (SEI-AMD) or BADGE-BUD.

In the SEI-BUD system, MetaTrans documents are exchanged between the structure tool and the SEI-BUD server. MetaTrans documents are also exchanged between the SEI-AMD system and the SEI-BUD server.

7.2. The MetaTrans logic

The MetaTrans DTD does not aim to be entirely generic and suitable for every DTD. It constitutes a representation of transactions carried out on an ABB budget publication at a higher level than "Xupd" format in that operations such as the merging of budget items have a semantic that is linked with the budget publication.

The aim is therefore to formalise modifications to the budget publication using high-level transactions. These transactions must fulfil two conditions:

they must provide an intelligible representation of the changes made to the document so that validation can be carried out using a suitable tool.

they must contain all the information needed for the automatic generation of low-level transactions ("Xupd"), which will be used to apply these changes to the XML repository.

Four transactions are defined:

the insertion of a new item ("insert")

the updating of an existing item ("update")

the deletion of an existing item ("delete")

the merging of two or more items ("merge")

7.3. The MetaTrans DTD

A MetaTrans document is an XML document containing mainly MetaTrans tagging but also XML fragments of the target publication. Thus, the DTD defines the MetaTrans elements using the prefix "mtt" for the MetaTrans namespace.

The root element of a MetaTrans document is the element "mtt:transactions".

```
<!ELEMENT mtt:transactions (mtt:meta-from,mtt:meta-dest,mtt:transaction+)>
<!ATTLIST mtt:transactions batch-id ID #REQUIRED version CDATA #IMPLIED xmlns:mtt CDATA #FIXED
"">
```

This element contains two pieces of information identifying the source and the destination of the document; this information is not used for conversion to "Xupd" format. Three attributes qualify this element: one attribute determining the namespace, one attribute giving the DTD version to which the document conforms and one attribute that can be used to identify the document in the production system.

```
<!ELEMENT mtt:meta-from (mtt:meta+)>
<!ELEMENT mtt:meta (#PCDATA)>
<!ATTLIST mtt:meta name NMTOKEN #REQUIRED>
The element "mtt:meta-from" is used to indicate the origin of the document using a dictionary
of terms, as in the following example:
<mtt:meta-from>
<mtt:meta name="appl">StructureTool</mtt:meta>
<mtt:meta name="version">1.2</mtt:meta>
<mtt:meta name="desc">Test</mtt:meta>
</mtt:meta-from>
<!ELEMENT mtt:meta-dest (mtt:meta+)>
```

Similarly, the element "mtt:meta-dest" is used to indicate the destination of the document.

```
<!ELEMENT mtt:transaction (mtt:desc?, (mtt:insert|mtt:update|mtt:delete|mtt:merge) )>
<!ELEMENT mtt:desc (#PCDATA)>
<!ATTLIST mtt:transaction id ID #REQUIRED status (pending | planned | closed | canceled)
"pending">
```

The element "mtt:transaction" describes a single transaction. A transaction begins with an optional text description followed by an element representing one of the three types of transaction.

```
<!ELEMENT mtt:insert (mtt:element)>
<!ATTLIST mtt:insert select CDATA #REQUIRED type (before|after|appendchild) "after">
```

The insertion operation is represented by the element "mtt:insert", the attributes of which qualify the way the insertion is to be performed: the attribute "select" is an XPath and is used to determine, in the document to be modified, the element used to anchor the insertion; the attribute "type" determines whether the insertion is to be made before the anchor, after the anchor, or whether a new child element should be added to the selected element. The element "mtt:element" defines the fragment to be inserted and is described later.

```
<!ELEMENT mtt:delete (mtt:element?)>
<!ATTLIST mtt:delete select CDATA #REQUIRED>
```

The deletion operation is represented by the element "mtt:delete", of which the attribute "select" is an XPath and is used to determine which element to delete in the document to be modified. The element "mtt:element" can be used for information, to document the fragment for deletion. It is not used by the MetaTrans to Xupd conversion process.

```
<!ELEMENT mtt:update (mtt:element)>
<!ATTLIST mtt:update select CDATA #REQUIRED>
```

The update operation is represented by the element "mtt:update", of which the attribute "select" is an XPath and is used to determine which element to update in the document to be modified. The element "mtt:element" defines the fragment to be used for the update and is described later.

Important note regarding updates:

If an attribute is updated, only this attribute is updated and other attributes are left unchanged;

If the content of an element is updated, all its contents are replaced, including attributes. So, if we need to update the heading and remarks of an 'nmc-title' element that contains several 'nmc-chapter', we need to individually address the title and the remarks in order to avoid loosing nested chapters.

```
<!ELEMENT mtt:merge (mtt:element)>
<!ATTLIST mtt:merge select CDATA #REQUIRED>
```

The merge operation is represented by the element "`mtt:merge`", of which the attribute "`select`" is an XPath and is used to determine in the document to be modified, the element into which the other budget items are to be merged. The element "`mtt:element`" defines the fragment to be used for the update and is described later. It should be noted that the items copied to the destination item are not deleted and, if necessary, must be deleted deliberately using one or more deletion transactions.

```
<!ELEMENT mtt:element (mtt:attribute* , mtt:contents?)>
<!ATTLIST mtt:element name %qname; #REQUIRED namespace CDATA #IMPLIED>
```

The element "`mtt-element`" determines the content of a transaction. The attribute "`name`" gives the name of the element in the target DTD and the optional attribute "`namespace`" indicates the namespace of this target DTD. The content of the element "`mtt-element`" can be a series of attribute values ("`mtt:attribute`") that may be followed by a document fragment ("`mtt:contents`").

```
<!ELEMENT mtt:attribute (#PCDATA)>
<!ATTLIST mtt:attribute name NMTOKEN #REQUIRED namespace CDATA #IMPLIED>
```

The element "`mtt:attribute`" is used to define a value for an attribute. The attribute "`name`" indicates the name of the attribute in the target DTD and, for the element specified by the attribute "`name`", the surrounding element "`mtt:element`"; the attribute "`namespace`" gives the corresponding namespace.

```
<!ELEMENT mtt:contents(mtt:content+ | mtt:copy-from | mtt:merge-from+)>
```

The element "`mtt-contents`" is used to specify a document fragment for the target publication. This document fragment can consist of an XML fragment (one per language) tagged according to the target DTD, or an "`mtt:copy-from`" instruction used to reference another budget item from which to copy the content, or several "`mtt-merge-from`" instructions used to specify the items to be merged.

```
<!ELEMENT mtt:content ANY>
<!ATTLIST mtt:content lang (da|de|el|en|es|fi|fr|it|nl|pt|sv) #REQUIRED>
```

The element "`mtt-content`" contains an ANY template in the DTD, but in practice this is XML tagging in another namespace referring to the target publication. The attribute "`lang`" indicates the language of the fragment.

```
<!ELEMENT mtt:copy-from (#PCDATA)>
<!ATTLIST mtt:copy-from select CDATA #REQUIRED>
```

The element "`mtt-copy-from`" is used to reference another budget item from which to copy the content. The attribute "`select`" is an XPath that should be able to be resolved in a budget item in the target document. This construction makes it possible in particular to produce transactions that are independent of the target language at the moment.

```
<!ELEMENT mtt:merge-from (#PCDATA)>
<!ATTLIST mtt:merge-from select CDATA #REQUIRED>
```

The element "`mtt-merge-from`" is a synonym of "`mtt-copy-from`" used only in merge operations; it is used to reference another budget item from which to copy the content. The attribute "`select`" is an XPath that should be able to be resolved in a budget item in the target document.

7.4. Examples of transactions

7.4.1. Deletion of an item with the unique identifier AAAKI.

```
<mtt:transaction id="T939"> <mtt:desc>Delete entry id AAAKI</mtt:desc> <mtt:delete
select="*[@id='AAAKI']"/> </mtt:transaction>
```

7.4.2. Insertion of a new Title item in Danish and German.

```
<mtt:transaction id="T940" status="pending">
<mtt:desc>Insert a new entry in the budgetary structure</mtt:desc>
<mtt:insert type="after" select="*[@id='AAAKI']">
<mtt:element name="nmc-title">
<mtt:attribute name="alias">04</mtt:attribute>
<mtt:contents>
<mtt:content lang="da"><bud-heading><p>TJENSTG&#00D8;RENDE PERSONALE</p></bud-
heading></mtt:content>
<mtt:content lang="de"><bud-heading><p>PERSONAL IM AKTIVEN DIENST</p></bud-
heading></mtt:content>
</mtt:contents>
</mtt:element>
</mtt:insert>
</mtt:transaction>
```

7.4.3. Update of a Title in Danish and German.

```
<mtt:transaction id="T937" status="pending">
<mtt:desc>Update alias and heading for title id AAAKI</mtt:desc>
<mtt:update select="*[@id='AAAKI']/bud-heading">
<mtt:element name="bud-heading">
<mtt:contents>
<mtt:content lang="da"><p>TJENSTG&#00D8;RENDE PERSONALE</p></mtt:content>
<mtt:content lang="de"><p>PERSONAL IM AKTIVEN DIENST</p></mtt:content>
</mtt:contents>
</mtt:element>
</mtt:update>
</mtt:transaction>
```

7.4.4. Merging of four Titles.

```
<mtt:transaction id="T942" status="pending">
<mtt:desc>Merge four budgetary entries into an existing one</mtt:desc>
<mtt:merge select="*[@id='AAAKI']">
<mtt:element name="nmc-title">
<mtt:attribute name="alias">04</mtt:attribute>
<mtt:contents>
<mtt:merge-from select="*[@id='AAAKJ']" />
<mtt:merge-from select="*[@id='AAAKL']" />
<mtt:merge-from select="*[@id='AAAKM']" />
</mtt:contents>
</mtt:element>
</mtt:merge>
</mtt:transaction>
```

7.4.5. Insertion of a new item by copying an existing item.

```
<mtt:transaction id="T941" status="pending">
<mtt:desc>Insert a new entry by copy of another one</mtt:desc>
<mtt:insert type="before" select="*[@id='AAAKI']">
<mtt:element name="nmc-title">
<mtt:attribute name="alias">04</mtt:attribute>
<mtt:contents>
<mtt:copy-from select="*[@id='AAAKI']"/> </mtt:contents>
</mtt:element>
</mtt:insert>
</mtt:transaction>
```

7.5. MetaTrans to XUpdate conversion

"High-level" transactions specified in MetaTrans format must be translated on the SEI-BUD server into Xupd format before they can be applied to the XML repository. This conversion is performed by the class "`Meta2xupd`" from the package "`com.softwareag.belgium.seibud.abb`"; this class derives from the class "`XSLTFilter`", which is part of the Xef framework.

This filter requires a mandatory parameter: the document language.

This filter consists of a single style sheet: "meta2xupd.xslt". Technically, it is split into three groups of rule according to three modes: the default mode processes the MetaTrans elements; the two other modes process the elements of the target DTD.

This filter has certain knowledge of the target DTD, in particular for translating merge operations. So when merging two items, "`bud-data`" and "`bud-data-extra`" should not be merged, but the other sub-elements of these items should be merged one by one: the content of "`bud-remarks`", "`bud-legal`" and "`bud-reference`" should be merged, remembering that all these sub-elements may be empty both in the source and the destination.

Transactions expressed by the structure item, for example, always relate to the version currently being edited; it is therefore possible to modify a budget item after having created it, or to make two successive updates to that item; Xupd transactions generated by the conversion therefore also relate to the version currently being edited, which means that either the XPaths need to be resolved dynamically, or the data used by the XPath processor need to be re-indexed.

8. INTERFACES WITH OTHER SYSTEMS

8.1. POETRY exchanges

8.1.1. Principles of Poetry exchanges

Translators at the Commission's Translation Service do not use the SEI-BUD control tool: they have their own system of access to documents for translation (Poetry).

When the Commission's Authors close the editing of a fragment (generally a Title), the supervisor responsible for the preparation of that fragment informs the Translation Service and transfers the files needed for translation to the Poetry system. To do this, the supervisor uses the "send translation request" function available in the SEI-BUD control tool.

This translation request must be qualified by a number of parameters explained below.

The translation request launched from the control tool arrives at the SEI-BUD server and is processed by the corresponding business object (`AbbDocument`). The server then validates the request and carries out a series of preparation operations (extraction of the fragment for translation, differentials, etc.). All these prepared documents are then sent to the Translation Service for translation.

The Translation Service sends back the fragments by electronic mail, one by one, as soon as each fragment is finished.

The SEI-BUD system regularly checks its dedicated mailbox and each time a translation is returned it launches an update request (as if that request came from a control tool).

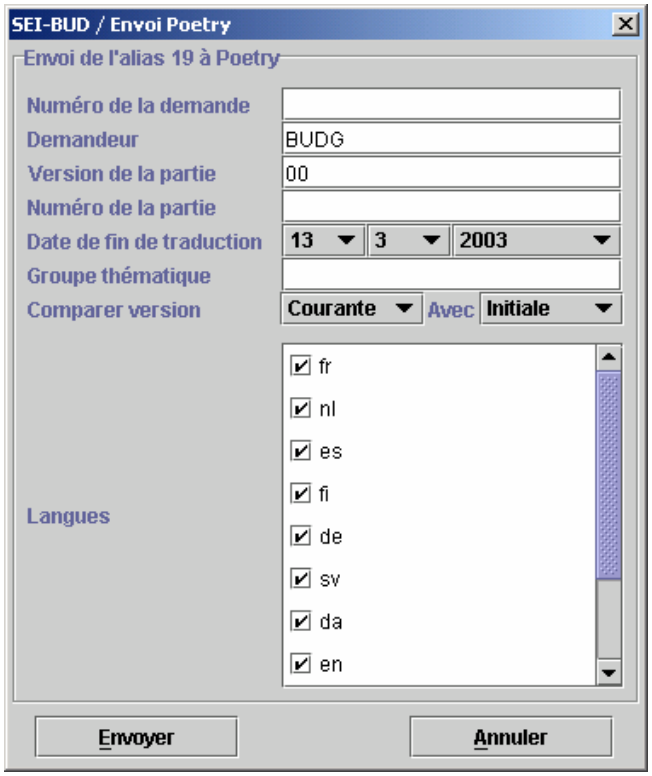
If any update requests fail, they are sent to the verifiers.

8.1.1.1. Parameters of a Poetry translation request

Several special requests are associated with Poetry dispatches:

the request for preparation and dispatch of a translation request to the Poetry system (this request is normally launched by the control tool): "`public send_poetry(SendPoetryRqst"`).

The parameters entered by the supervisor into the control tool are shown in the screen shot below:



These parameters are as follows:

an order number; this number is allocated to the supervisor by the Translation Service and covers an entire publication;

the requester; by default, the keyword "BUDG" is used to indicate that this is a budget document issued by the SEI-BUD system;

a version number for the section; this version number is incremented every time the same fragment is sent for the same publication; this can happen where author changes are made after the section is sent for translation;

a part number, which is used to distinguish between the different fragments that make up the publication to be translated; for budget publications, this number is generally based on the alias; for ABB publication, where the fragments sent for translation are titles, it is, for instance, the title number that is used;

the translation end date gives an indication of when the translation should be finished;

the subject group is used to indicate the semantic area to which the fragment for translation relates;

the supervisor can select the target languages for translation;

it is also possible to specify the labels of the two versions to be used to produce the differential (a label is normally created at a change of phase so that, for an entire section of a publication, there is a

mnemonic link between a name and a version number); for the fragment that needs to be translated, the different labels are obtained by the control tool using the request "`getVersionLabels`";

These parameters entered by the supervisor are completed by the control tool to make the parameter "`SendPoetryRqst`" required by the method "`send_poetry`". The request "`SendPoetryRqst`" is described in detail in the section on the data handled by business objects.

8.1.1.2. Preparation of a translation request

Translation requests are asynchronous requests (see the processing of asynchronous requests in the section on "the principle governing communication between the server and the clients"). This request is asynchronous because it requires a number of preparation operations to be carried out within the server:

change of user to perform the different operations involved in processing a translation request; the system uses a dedicated user to send Poetry requests; this allows processing reports for returned translations to be separated: these processing reports are not addressed to the supervisor who issued the translation request, but to the verifier who may need to act if there is an update failure.

then, for all languages (including the "author" language, which does not need to be translated)

insertion in the table "RMG_PoetryIdentifier" of an entry corresponding to the fragment for translation; this entry will make it possible to find the management information for this fragment when the translation is returned;

extraction of the current version of the fragment for translation using the method "`checkout_translation`" of the business object`AbbDocument`.

extraction of the original version in the author's language (normally French) to serve as input data for generating the differential;

calculation of differences with production of the result in HTML format (see section on "Other client functions": "printing out differences")

use of the method "`postMail`" from the class "`PoetryMailDispatcher`" (package "`com.softwareag.belgium.seibud.rmg.tools`"); this method packs as e-mail attachments the various documents obtained so far, limiting the number of attachments per message; this method uses the basic functions of the standard Java package "`javax.mail`" to actually send the e-mails assembled in this way. It should be noted that the message subject contains within a precise syntax the date, the Poetry number, the version number and the part number.

8.1.1.3. Reception of returned translations: the "mailreader" daemon

The "`mailreader`" daemon is a programme executed at regular intervals of time, to scan the mailbox that receives returned translations. It is a Java client programme, without a user interface, which runs on the same machine as the SEI-BUD server (but could, in practice, run on a different machine). It scans the mail inbox and processes messages by querying the SEI-BUD server.

The "`mailreader`" client is implemented by two classes:`MailReader`and`SubjectValidator`from the package "`com.softwareag.belgium.seibud.mailreader`"

Inspection of the mailbox

The virtual user "Poetry" has an account on the mail server.

Initialisation of the mailreader client consists of reading configuration data from the XML file "`mail-reader-config.xml`", connecting to the mail server, and executing a request to "`login`" to the SEI-BUD server via an XMLDispatcher in RMI mode rather than HTTP (see the mechanism for communication with the SEI-BUD server).

At regular intervals it scans the contents of the inbox and processes messages by looking at the message subject and possibly also the format of the attachment(s).

Besides the inbox, this account also has several storage folders located within an "`archive`" folder:

the folder "`Acknowledgements`" is used to hold acknowledgement messages for translation requests;

the folder "`Rejected`", on the other hand, is used to hold rejection messages for translation requests;

the folder "`Invalid`" is used to hold returned translations for which the programme has detected non-valid content, for example no attachment when the return e-mail is supposed to have a document attached;

the folder "`No Reservation`" contains returns for which a Poetry reservation cannot be found; this can happen either because the message is corrupted, or because the supervisor has cancelled the translation request;

the folder "`Processed`" contains translation returns that have been correctly processed by the SEI-BUD server (but not necessarily successfully processed: these are documents for which the SEI-BUD server has created a processing report; this processing report indicates whether or not the update was successful);

the folder "`Unexpected Error`" contains translation returns that need to be examined by the verifier because they could not be processed by the SEI-BUD server (there will be no processing report for these documents).

Update of a translated document

Updating a translation return involves unpacking the document attached to the message and wrapping it in a "`checkin_poetry`" request (class method from the object`AbbDocument`).

The class method "`checkin_poetry`" is in fact a wrapper around the synchronous object method "`checkin`". Before activating the method "`checkin`", the method "`checkin_poetry`" executes a login request with the username reserved for Poetry returns; it retrieves the identifier of the Poetry document from the table "`RMG_PoetryIdentifier`", and uses this to create the parameters needed for the "`checkin`" request. It should be noted that a new transaction is created before this "`checkin`" request is executed; in fact, if an update failure occurs during this method, the operations performed by the "`checkin`" method can be cancelled, but not those performed by the method "`checkin_poetry`".

Whether or not the update is successful, a processing report (ProcessingReport) is created by a control tool. A verifier or supervisor can examine the report by means of the control tool, using the username especially for this purpose.

Configuration of the "mailreader" daemon

The XML file "mail-reader-config.xml" contains the parameters needed to configure the client "mailreader". This file responds to the following DTD:

```
<!ELEMENT mail-reader (poetry, rmi-mode-properties, mail-session-properties, mail-retrieval-
properties)>
<!ELEMENT poetry (userid, password, askingdg)>
<!ELEMENT userid (#PCDATA)>
<!ELEMENT password (#PCDATA)>
<!ELEMENT askingdg (#PCDATA)>
```

The element "mail-reader" is the root element of the configuration file. It firstly contains certain parameters relating to the Poetry system: the Poetry user name and password, and the name of the translation requester.

```
<!ELEMENT rmi-mode-properties (property+)>
<!ELEMENT property (#PCDATA)>
<!ATTLIST property name CDATA #REQUIRED>
```

The element "rmi-mode-properties" contains the properties needed to instantiate an XMLDispatcher used to issue requests to the SEI-BUD server.

```
<!ELEMENT mail-session-properties (property+)>
```

The elements "mail-session-properties" and "mail-retrieval-properties" contain the information needed to connect to the mail server and scan the inbox.

The following is an example of a configuration file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mail-reader>
<poetry>
<userid>poetry</userid>
<password>poetry</password>
<askingdg>BUDG</askingdg>
</poetry>
<rmi-mode-properties>
<property name="app_name">seibud/RMGXMLDispatcher</property>
<property name="initial_context_factory">org.jnp.interfaces.NamingContextFactory</property>
<property name="provider_url">localhost:1099</property>
<property name="url_pkg_prefixes">org.jboss.naming:org.jnp.interfaces</property>
</rmi-mode-properties>
<mail-session-properties>
<property name="mail.store.protocol">imap</property>
<property name="mail.host">ALCESTE</property>
<property name="mail.user">lft</property>
<property name="mail.debug">false</property>
</mail-session-properties>
<mail-retrieval-properties>
<property name="password">lft</property>
<property name="mbox">INBOX</property>
<property name="wait">600000</property>
</mail-retrieval-properties>
</mail-reader>
```

8.2. Interfaces with the Parliament

Translators at the Parliament's Translation Service do not use the SEI-BUD control tool: they have their own system of access to documents for translation.

The author of the amendment (or another appropriate user) uses the Web Interface to request the translation of a set of amendments. The user selects the desired translation languages. A traduction delay and a request number are also specified (by default, there is a 7 days delay).

Upon user request, the systems prepares a set of compressed RTF documents (Translation Text or TT format) and sends all those documents to the Parliament to an FTP server.

Once translated, documents are either directly uploaded via HTTP (as are normal author documents) or transferred to a dedicated directory (the transfer being managed by EUR-OP).

This directory is periodically polled by a "ftpreader" daemon process. (In this case, the name "ftpreader" is somewhat misleading since the FTP protocol is not used but the same program can be used either to poll a local directory or an FTP server). Translated documents are made of two files: one mandatory compressed XML file and an optional compressed RTF file. Compressed RTF files are simply copied to an archive directory. XML files are uncompressed and submitted for checkin to the Repository Manager. If any update requests fails, the faulty document is moved to a dedicated directory for later manual diagnostic.

8.2.1. File naming conventions

The naming conventions of document files sent to the translation service of the Parliament are expressed with the following pattern:

`NNNN_L*_NT_YYYYMMDDHHMISS_{P}SL2TL`

where

NNNN: amendment tabling number (4 digit left padded with zeros) if it exists; otherwiseit is the amendment label (characters 6 to 9 if the length is greater than 9, otherwise the last 4 characters).

L*: amendment label

NT: number of requested translations

YYYYMMDDHHMISS: start date of the translation

P: (optional) present when it is a multilingual document

SL: source language

TL: target language

Remark: for the amendment label, the following characters " ", "/", "\" are replaced by underscores

8.2.2. Configuration of the FTP transfer

The configuration information relative to the FTP transfer of documents for translation is specified in the`rmg-config.xml`file in the`FtpPut`section:

element`<remote_hostname>`specified the FTP serveur name where to be translated documents must be put (an IP address can also be specified);

elements`<username>`and`<password>`specify user login information;

element`<target_directory_root>`specifies the root directory to be used on the FTP server;

element`<suffix_dir>`can be used to specify a sub-directory inside the directory specified by`<target_directory_root>`;

element<disabled>is used to debugging purpose, to issue translation requests and perform translation document preparation without actually sending the documents. If the value istrue, no transfer is performed.

8.2.3. Configuration of the "ftpreader" daemon

Configuration information used by the FTP reader is specified in theftp-reader-config.xmlfile.

The<rmi-properties>element specifies a set of properties defined by the Java Naming and Directory Interface (JNDI) API to initialise a JNDI context (address of JNDI service, prefix, ...).

The<mail-session-properties>element specifies a set of properties defined by the Java Mail API to configure a session with a mail server (protocol, server address, user identifier). This information is currently not used by the "ftpreader" daemon but is intended to be used for sending messages to the system administrator when something goes wrong.

The<FtpGet>section of the specified parameters used to access the FTP server:

element<remote_hostname>indicates the server name (it can be an IP address);

elements<username>and<password>specify the login information;

The<rmg-user>section specifies the Repository user information to be used by the FTP reader to issue checkin requests:<userid>and<password>elements;

8.2.4. Configuration of the “codictreader” daemon

The CODICTreader configuration goes through the following steps:

Configure the user’s environment variables.

Create the following configuration files:

codict-reader-config.xml: configuration of the directories and of the FTP transfers

codict.datadump.db.properties: configuration of the database access.

codict.datadump.load.properties: configuration of the tables to load.

log4j-codict-reader.properties: configuration of LOG4J.

8.2.4.1. Configure the environment variables

The owner user of CODICTreader must set the two variables:

PATH: modify this variable to get the good java RunTime.

NLS_LANG: this variable is used by the Data dump module. Normally, set it to:

NLS_LANG=AMERICAN_AMERICA.UTF8

On Unix Sun Solaris side, these two variables should be initialized and exported in the‘ .profile ,file.

On Windows XP side, these two variables should be initialized using:

Start Control Panel System

System Properties Advanced Environments Variables

8.2.4.2. Create the configuration files

The configuration files must be put inside the general configuration file of the SEI-BUD server (~seibud4/run/conf)

Then, the four configuration templates must be copied to the configuration directory from the directory:

seibud_codictreader/conf/

Then, the four copied configuration files must be edited.

Edition of‘ codict.datadump.db.properties ,

In this file‘ codict_db_export.properties ,, modify the following lines:

dbUrl = jdbc:oracle:thin:@hostaddress:1521:databasenameIn this line, modify the tow parts‘ hostaddress ,and‘ databasename ,

dbUser = codictIn this line, replace‘ codict ,by the name of the Oracle user.

dbPassword = codictIn this line, replace‘ codict ,by the password of the Oracle user.

dbSchema = CODICTIn this line, replace‘ CODICT,by the name of the schema having the tables and views.

Edition of‘ codict.datadump.load.properties ,

The file « codict_tables_export.properties » does not need to be modified.

Edition of‘ codict-reader-config.xml ,

In this file, the strings between <!-- … --> must be replaced. If FTP is not used then the strings 1-4 are not used.

However, null strings are not allowed. So, the strings 1-4 must be replaced by a space or by any dummy value.

<!-- FTP server hostname -->: to be replaced by the FTP hostname.

<!-- FTP user id -->: to be replaced by the FTP user name.

<!-- FTP password -->: to be replaced by the FTP user password.

<!-- absolute pathname of Remote root directory where to look into -->: to be replaced by the absolute pathname of the remote directory where the database dump files must be searched.

<!-- absolute pathname of local directory where to load files -->: to be replaced by the absolute pathname of the local directory where the database dump files must be searched for.In the OPOCE production environment, this directory must be the same as the one where the European Parliament translation files are transferred.

<!-- absolute pathname of local directory where to copy loaded files -->: to be replaced with the absolute pathname of the local directory where to copy the loaded files.In the OPOCE production environment, this directory may be the same as the directory where the European Parliament translation files are copied after their handling. However, it is advisable not to mix both directories

<!-- absolute pathname of local directory where to copy bad files -->: to be replaced by the absolute pathname of the local directory where to copy files that caused an error.In the OPOCE production environment, this directory may be the same as the directory where the European Parliament translation files are copied when their handling causes an error. However, it is advisable not to mix both directories.

<!-- use FTP: false or true -->: to be replaced by true if FTP must be used or false otherwise.

<!-- absolute pathname of temporary directory where to extract files from compressed files -->: to be replaced by the absolute pathname of a temporary directory where to extract the content from the compressed files.In the OPOCE production environment, it may be the same as the common SEI-BUD directory where all the temporary files are put.

Edition of‘ log4j-codict-reader.properties ,

The file log4j-codict-reader.properties » does not need to be modified.

8.3. Interfaces with the Council

Translators at the Council Translation Service do not use the SEI-BUD control tool: they have their own system of access to documents for translation.

The author of the amendment (or another appropriate user) uses the Web Interface to request the translation of a set of amendments. The user selects the desired translation languages. A traduction delay and a request number are also specified (by default, there is a 7 days delay).

Upon user request, the systems prepares a set of compressed RTF documents (Translation Text or TT format) and sends all those documents to the Council to an HTTP server.

Once translated, documents are either directly uploaded via HTTP (as are normal author documents).

8.3.1. File naming conventions

The naming conventions of document files sent to the translation service of the Council are expressed with the following pattern:

```
NNNN{"N"|"P"}SLTL
```

where

NNNN: amendment tabling number (4 digit left padded with zeros) if it exists; otherwiseit is the amendment label (characters 6 to 9 if the length is greater than 9, otherwise the last 4 characters).

N: for normal documents,**P**: for multilingual documents,

SL: source language

TL: target language

8.3.2. Configuration of the HTTP transfer

The configuration information relative to the HTTP transfer of documents for translation is specified in the`rmg-config.xml`file in the`HttpSendFile`section:

element`<url>`specifies the URL used to post document files;

elements`<username>`and`<password>`specify user login information;

element`<disabled>`is used to debugging purpose, to issue translation requests and perform translation document preparation without actually sending the documents. If the value is`true`, no transfer is performed.

8.4. Interface with the CODICT system

8.4.1. Introduction

SEI-BUD+Amendments system needs a daily update of data coming from CODICT database.

This update is executed in three steps:

The data extraction from the useful CODICT views into a compressed file.

The FTP transfer of this compressed file from the CODICT system to the SEI-BUD+Amendments system.

The loading of the compressed file into the SEI-BUD+Amendments database.

The third step is executed by a load module named‘ CODICTreader ,.

The load module is running as a daemon.

It looks periodically for compressed files in a dedicated directory and loads them into the SEI-BUD+Amendments database.

The directory can either be located in a local file system or in a remote one.

If needed, CodictReader can retrieve the compressed files from a remote directory using the FTP protocol.

The scripts of the CODICTreader tool can execute on Windows XP or on Unix Sun Solaris.

8.4.2. Loaded tables

The dump should contain the following tables:

V_BODY_DIFF

V_ROLM_DIFF

V_MEMB_CURRENT

8.4.3. Created tables and views

The following tables are created:

V_BODY_DIFF

V_ROLM_DIFF

V_MEMB_CURRENT

The following tables “T_*” are needed to store the old data of the same tables “V_*” before the loading:

T_BODY_DIFF

T_ROLM_DIFF

T_MEMB_CURRENT

9. ANNEX I: ABB DTD

9.1. Root declarations: abb.dtd

```
<!--
Author : Software AG Belgium - 2001
Comments : abb dtd
The budgetary dtd for an abb publication.
-->
<!ENTITY % rep "INCLUDE">
<!ENTITY % prs "IGNORE">
<!ENTITY % wrx "IGNORE">
<!ENTITY % bud-data.figures "INCLUDE">
<!ENTITY % bud-data.tables "IGNORE">
<!--
The basic text...
-->
<!ENTITY % abb-btx.module "INCLUDE">
<![%abb-btx.module;[
<!ENTITY % abb-btx SYSTEM
"abb-btx.mod">
%abb-btx;
] ] >
<!--
The budgetary figures...
-->
<!ENTITY % abb-fig.module "INCLUDE">
<![%abb-fig.module;[
<!ENTITY % abb-fig SYSTEM
"abb-fig.mod">
%abb-fig;
] ] >
<!--
The budgetary nomenclature elements...
-->
<!ENTITY % abb-nmc.module "INCLUDE">
<![%abb-nmc.module;[
<!ENTITY % abb-nmc SYSTEM
```

```
"abb-nmc.mod">
%abb-nmc;
] ] >
<!--
The tables...
-->
<!ENTITY % abb-tbl.module "INCLUDE">
<![%abb-tbl.module;[
<!ENTITY % abb-tbl SYSTEM
"abb-tbl.mod">
%abb-tbl;
] ]>
```

9.2. Nomenclature element declarations: abb-nmc.mod

```
<!--
Author : Software AG Belgium - 2001-2004
Comments : abb-nmc dtd
The budgetary nomenclature elements for an abb publication.
History:
11feb2004 add reuses at the level of nomenclature elements
add nomenclature elements type attribute for activities including/without budget line
add metadata
-->
<!ENTITY % nomenc.class "nmc-section | nmc-sectpart | nmc-revenue | nmc-expenditure | nmc-
title | nmc-chapter | nmc-article | nmc-item">
<![%wrx;[
<!ENTITY % appendix.class "nmc-annex | sect1">
] ] >
<!ENTITY % appendix.class "nmc-annex | nmc-grseq">
<!ENTITY % ext.nomenc.unit "%nomenc.class; | %appendix.class;">
<!ENTITY % lockable.unit "%ext.nomenc.unit; | bud-heading | bud-intro | bud-data | bud-remarks
| bud-legal | bud-reference | bud-text">
<![%rep;[
<!ENTITY % ext.nomenc.unit.attrib "
alias CDATA #IMPLIED
id ID #REQUIRED
state (hidden) #IMPLIED
type (awbl|aibl) #IMPLIED"
>
] ] >
<![%prs;[
<!ENTITY % ext.nomenc.unit.attrib "
alias CDATA #IMPLIED
id ID #IMPLIED
state (hidden) #IMPLIED
type (awbl|aibl) #IMPLIED"
>
] ] >
<![%wrx;[
<!ENTITY % ext.nomenc.unit.attrib "
alias CDATA #IMPLIED"
>
] ] >
<!ENTITY % unit.content.attrib "
translate (translate) #IMPLIED">
<!-- abb, abbinfo ..................................................... -->
<!ENTITY % abb.module "INCLUDE">
<![%abb.module;[
<![%wrx;[
<!ENTITY % abb.attrib "
<!ATTLIST abb
xml:lang CDATA #IMPLIED
>"
>
] ] >
<!ENTITY % abb.attrib "">
<![%wrx;[
```

```
<!ENTITY % nmc.abb.mdl "metadata?, (%ext.nomenc.unit;)">
] ] >
<!ENTITY % nmc.abb.mdl "metadata?, (%lockable.unit;)">
<!ENTITY % abb.element "INCLUDE">
<![%abb.element;[
<!ELEMENT abb (%nmc.abb.mdl;)>
%abb.attrib;
] ] >
] ] >
<!-- ================================================================== -->
<!-- metadata -->
<!-- ================================================================== -->
<!ELEMENT metadata (m.entry)+>
<!ELEMENT m.entry (m.occ+ | m.val)>
<!ATTLIST m.entry
name CDATA #REQUIRED
>
<!ELEMENT m.occ (m.val+)>
<!ELEMENT m.val (m.occ*)>
<!ATTLIST m.val
name CDATA #REQUIRED
value CDATA #IMPLIED
>
<!-- ================================================================== -->
<!-- budgetary nomenclature elements (nomenc.class) -->
<!-- ================================================================== -->
<![%rep;[
<!ENTITY % bud-data.opt "bud-data">
] ] >
<![%prs;[
<!ENTITY % bud-data.opt "bud-data">
] ] >
<![%wrx;[
<!ENTITY % bud-data.opt "bud-data">
] ] >
<!ELEMENT nmc-section (bud-heading,
(
(nmc-section|reuse-link|(%appendix.class;))+
| (bud-intro?, bud-remarks?, bud-legal?, bud-reference?,
(nmc-sectpart+ | (nmc-revenue?, nmc-expenditure?)) )
))>
<!ATTLIST nmc-section
%ext.nomenc.unit.attrib;
>
<!ELEMENT nmc-sectpart (bud-heading, (reuse-link | (bud-intro?, bud-remarks?, bud-legal?, bud-
reference?, nmc-revenue?, nmc-expenditure?, (%appendix.class;)*)))>
<!ATTLIST nmc-sectpart
%ext.nomenc.unit.attrib;
>
<!ELEMENT nmc-revenue (bud-heading, (reuse-link | (bud-intro?, %bud-data.opt;, bud-remarks?,
bud-legal?, bud-reference?, nmc-title*, (%appendix.class;)*)))>
<!ATTLIST nmc-revenue
%ext.nomenc.unit.attrib;
>
<!ELEMENT nmc-expenditure (bud-heading, (reuse-link | (bud-intro?, %bud-data.opt;, bud-
remarks?, bud-legal?, bud-reference?, nmc-title*, (%appendix.class;)*)))>
<!ATTLIST nmc-expenditure
%ext.nomenc.unit.attrib;
>
<!ELEMENT nmc-title (bud-heading, (reuse-link | (bud-intro?, %bud-data.opt;, bud-remarks?,
bud-legal?, bud-reference?, nmc-chapter*)))>
<!ATTLIST nmc-title
%ext.nomenc.unit.attrib;
>
<!ELEMENT nmc-chapter (bud-heading, (reuse-link | (bud-intro?, %bud-data.opt;, bud-remarks?,
bud-legal?, bud-reference?, nmc-article*)))>
<!ATTLIST nmc-chapter
%ext.nomenc.unit.attrib;
```

```
>
<!ELEMENT nmc-article (bud-heading, (reuse-link | (bud-intro?, %bud-data.opt;, bud-remarks?,
bud-legal?, bud-reference?, nmc-item*)))>
<!ATTLIST nmc-article
%ext.nomenc.unit.attrib;
>
<!ELEMENT nmc-item (bud-heading, (reuse-link | (bud-intro?, %bud-data.opt;, bud-remarks?, bud-
legal?, bud-reference?, nmc-subitem*)))>
<!ATTLIST nmc-item
%ext.nomenc.unit.attrib;
>
<!-- subitems have been introduced to support nature classification within horizontal
affectations -->
<!ELEMENT nmc-subitem (bud-heading, (reuse-link | (bud-intro?, %bud-data.opt;)))>
<!ATTLIST nmc-subitem
%ext.nomenc.unit.attrib;
>
<!-- appendix.class -->
<!ELEMENT nmc-annex (bud-heading, (reuse-link | (bud-intro?, bud-remarks?, bud-legal?, bud-
reference?, (nmc-revenue?, nmc-expenditure?))))>
<!ATTLIST nmc-annex
%ext.nomenc.unit.attrib;
>
<![%rep;[
<!ELEMENT nmc-grseq (bud-heading,(reuse-link| (bud-text?, nmc-grseq*)))>
<!ATTLIST nmc-grseq
%ext.nomenc.unit.attrib;
>
] ] >
<![%prs;[
<!ELEMENT nmc-grseq (bud-heading,(reuse-link| (bud-text?, nmc-grseq*)))>
<!ATTLIST nmc-grseq
%ext.nomenc.unit.attrib;
>
] ] >
<![%wrx;[
<!ELEMENT sect1 (bud-heading,(reuse-link| (bud-text, sect2*)))>
<!ATTLIST sect1
%ext.nomenc.unit.attrib;
>
<!ELEMENT sect2 (bud-heading,(reuse-link| (bud-text, sect3*)))>
<!ATTLIST sect2
%ext.nomenc.unit.attrib;
>
<!ELEMENT sect3 (bud-heading,(reuse-link| (bud-text, sect4*)))>
<!ATTLIST sect3
%ext.nomenc.unit.attrib;
>
<!ELEMENT sect4 (bud-heading,(reuse-link| (bud-text, sect5*)))>
<!ATTLIST sect4
%ext.nomenc.unit.attrib;
>
<!ELEMENT sect5 (bud-heading,(reuse-link|bud-text)>
<!ATTLIST sect5
%ext.nomenc.unit.attrib;
>
] ] >
<!-- ================================================================== -->
<!-- bud-heading, bud-data, bud-remarks, bud-legal, bud-reference, bud-text -->
<!-- ================================================================== -->
<!ELEMENT bud-heading (%para.class;)*>
<!ATTLIST bud-heading
%unit.content.attrib;
>
<!ELEMENT bud-intro (%para.class; | %list.class; | %formal.class;)*>
<!ATTLIST bud-intro
%unit.content.attrib;
>
```

```
<!ELEMENT bud-remarks (%para.class; | %list.class; | %formal.class;)*>
<!ATTLIST bud-remarks
%unit.content.attrib;
>
<!ELEMENT bud-legal (%para.class; | %list.class; | %formal.class;)*>
<!ATTLIST bud-legal
%unit.content.attrib;
>
<!ELEMENT bud-reference (%para.class; | %list.class; | %formal.class;)*>
<!ATTLIST bud-reference
%unit.content.attrib;
>
<!ELEMENT bud-text (%para.class; | %list.class; | %formal.class;)*>
<!ATTLIST bud-text
%unit.content.attrib;
>
```

9.3. Figure element declarations: abb-fig.mod

```
<!--
Author : Software AG Belgium - 2001
Comments : abb-fig dtd
The budgetary figures for an abb publication.
History:
11feb2004 remove bud-data-extra
add more type attribute values to human resources ('not distributed' and 'not decentralised')
change horizontal=true to type=horizontal or reserve
-->
<!-- ================================================================== -->
<!-- budgetary figures -->
<!-- ================================================================== -->
<![%bud-data.figures;[
<!ELEMENT bud-data (amounts?, hresources?, schedules?, relations?)>
<!ATTLIST bud-data
exprev (exp|rev) #IMPLIED
tot (tot) #IMPLIED
type (horizontal|reserve) #IMPLIED
>
<!ELEMENT amounts (amount)* >
<!ELEMENT amount (figure+, reserve?)? >
<!-- amount attributes
aele: ? (true)
catpol: category.policy or financial perspective classification
comp: compulsory (true)
delegation: for policies which have two budgets (institution and delegation) (true)
diff: differentiated appropriation (true)
peco: ? (true)
year: current (n), minus 1 (nm1), minus 2 (nm2), brs/lr amount (delta), new amount (new)
-->
<!ATTLIST amount
aele (true) #IMPLIED
catpol CDATA #IMPLIED
comp (true) #IMPLIED
delegation (true) #IMPLIED
diff (true) #IMPLIED
peco (true) #IMPLIED
year (n | nm1 | nm2 | delta | new) #IMPLIED
>
<!ELEMENT hresources (hresource)* >
<!ELEMENT hresource (figure, relations?)? >
<!-- hresource attributes
type:
Establishment plan staff : operational (operation)
Establishment plan staff : research (research)
Support staff ex-A-7 (support)
Support staff other (othersupport)
Linguistic services (linguistic)
total (tot)
```

```
not distributed (ndis)
not decentralised (ndec)
year: current (n), minus 1 (nm1), minus 2 (nm2), brs/lr amount (delta), new amount (new)
-->
<!ATTLIST hresource
type (operation|research|support|othersupport|linguistic|tot|epstat|policy|pending
|ndis|ndec) 'operation'
year (n | nm1 | nm2 | delta | new) 'n'
>
<!ELEMENT schedules (schedule)* >
<!-- the model of schedule element should be (figure, figurenote*)? but we kept the note
element for backward compatibility -->
<!ELEMENT schedule (figure, (note | figurenote)*)? >
<!-- schedule attributes
type: comm. still outstanding (cso), comm. made available (cma), year minus 1 (nm1), current
year (n), total (tot). Indicates the row in the table.
year: minus 1 (nm1), current (n), plus 1 (np1), plus 2 (np2), subsequent (npx)
-->
<!ATTLIST schedule
type (cso | cma | nm1 | n | tot) #IMPLIED
year (nm1 | n | np1 |np2 | npx) #IMPLIED
>
<!ELEMENT reserve (figure+, alias) >
<!ELEMENT figure (#PCDATA) >
<!-- figure attribute
commpay: commitment (comm), payment (pay)
-->
<!ATTLIST figure
commpay (comm | pay) #IMPLIED
>
<!ELEMENT alias (#PCDATA) >
<!ELEMENT figurenote (%para.class;)*>
<!ELEMENT relations (relation)* >
<!ELEMENT relation EMPTY >
<!-- relation attributes
type: the referenced alias of the pcdata is from 'XX 01' inside the document (admin) or from
the traditional nomenclature (traditional)
pp ('pour partie'):
type traditional: for 'abb' elements which correspond to only a part of the 'traditional'
element (true)
type admin: Gives the percentage of the amount which appears in XX 01 ('..%')
value: alias
-->
<!ATTLIST relation
pp CDATA #IMPLIED
type (admin | traditional | policy) 'admin'
value CDATA #IMPLIED
>
<!-- relation values :
type traditional:
alias of the traditional nomenclature
type admin:
XX 01 01 01 staff-institution (remuneration, transfers)
XX 01 01 02 staff-delegation (remuneration, transfers)
XX 01 02 01 external-institution (auxiliary, agency, temporary)
XX 01 02 02 external-delegation (local, training, other)
XX 01 02 11 other-institution (interpretation, translation, mission, conference, meetings,
studies, training)
XX 01 02 12 other-delegation (mission, training)
XX 01 03 01 buildings-institution (acquisition, other, equipment, services)
XX 01 03 02 buildings-delegation (acquisition, equipment)
XX 01 05 01 staff-research
XX 01 05 02 external-research
XX 01 05 03 other-research
-->
] ] >
<!--
.....................................................................
```

```
For presentation purposes, figures are translated into tables.
.....................................................................
-->
<![%bud-data.tables;[
<!ELEMENT bud-data (table)*>
<!ATTLIST bud-data
exprev (exp|rev)        #IMPLIED
tot (tot)        #IMPLIED
type     (horizontal|reserve) #IMPLIED
>
] ] >
```

9.4. Basic text components declarations: abb-btx.mod

```
<!--
Author : Software AG Belgium - 2001
Comments : abb-btx dtd
Basic text elements.
History:
11feb2004 add a mark with a language attribute to enclose reported authoring text
-->
<!-- ============================================================================ -->
<!-- Object-level classes -->
<!-- ============================================================================ -->
<!ENTITY % para.class "p | reuse-link">
<!ENTITY % formal.class "table">
<!ENTITY % list.class "list">
<!-- ============================================================================ -->
<!-- Inline-level classes -->
<!-- ============================================================================ -->
<![%rep;[
<!ENTITY % char.class "note | ft | qt">
] ] >
<![%prs;[
<!ENTITY % char.class "note | ft | qt | prefix">
] ] >
<![%wrx;[
<!ENTITY % char.class "note | emphasis | glossterm | subscript | superscript | uppercase">
] ] >
<!ENTITY % btx "(#PCDATA | %char.class;)*">
<!-- ============================================================================ -->
<!-- Objects -->
<!-- ============================================================================ -->
<!ELEMENT p %btx;>
<!ATTLIST p
lang CDATA #IMPLIED
>
<![%rep;[
<!ELEMENT list (int.li?, item+, close.li?)>
<!ATTLIST list
type (dash | bullet | none | arab | uc | lc | romuc | romlc) "dash"
>
<!ELEMENT item (%para.class; | %list.class;)*>
<!ELEMENT int.li (%para.class;)?>
<!ELEMENT close.li (%para.class;)?>
] ] >
<![%prs;[
<!ELEMENT list (int.li?, item+, close.li?)>
<!ATTLIST list
type (dash | bullet | none | arab | uc | lc | romuc | romlc) "dash"
>
<!ELEMENT item (%para.class; | %list.class;)*>
<!ELEMENT int.li (%para.class;)?>
<!ELEMENT close.li (%para.class;)?>
] ] >
<![%wrx;[
<!ELEMENT list (listitem+)>
<!ATTLIST list
```

```
type (dash | bullet | none | arab | uc | lc | romuc | romlc) "dash"
spacing (normal | compact) "compact"
inheritnum (inherit | ignore) "ignore"
continuation (continues | restarts) "restarts"
>
<!ELEMENT listitem (%para.class; | %list.class;)*>
<!ATTLIST listitem
override CDATA #IMPLIED
>
] ] >
<!ENTITY % btx.title.mdl "(%para.class;)*">
<!ENTITY % btx.subtitle.mdl "(%para.class;)">
<!ELEMENT title %btx.title.mdl;>
<!ATTLIST title
align (left|right|center|justify) #IMPLIED
>
<!ELEMENT subtitle %btx.subtitle.mdl;>
<!ATTLIST subtitle
align (left|right|center|justify) #IMPLIED
>
<!-- ================================================================== -->
<!-- Inline elements -->
<!-- ================================================================== -->
<!ELEMENT note (%para.class; | %list.class;)*>
<![%rep;[
<!ELEMENT ft %btx;>
<!ATTLIST ft
type (ht | sup | sub | uc | glossary) #IMPLIED
>
<!ELEMENT qt %btx;>
<!ELEMENT lang %btx;>
<!ATTLIST lang
lg CDATA #REQUIRED
>
] ] >
<![%prs;[
<!ELEMENT ft %btx;>
<!ATTLIST ft
type (ht | sup | sub | uc | glossary) #IMPLIED
>
<!ELEMENT qt %btx;>
<!ELEMENT prefix %btx;>
<!ELEMENT lang %btx;>
<!ATTLIST lang
lg CDATA #REQUIRED
>
] ] >
<![%wrx;[
<!ELEMENT emphasis %btx;>
<!ELEMENT glossterm %btx;>
<!ELEMENT subscript %btx;>
<!ELEMENT superscript %btx;>
<!ELEMENT uppercase %btx;>
] ] >
<!-- ================================================================== -->
<!-- Inclusion or reuse-link -->
<!-- ================================================================== -->
<!ELEMENT reuse-link (reuse-param|reuse-param-list)+>
<!ATTLIST reuse-link
reuse CDATA #FIXED "reuse-link"
>
<!ELEMENT reuse-param EMPTY>
<!ATTLIST reuse-param
name CDATA #REQUIRED
value CDATA #REQUIRED
>
<!ELEMENT reuse-param-list (reuse-param)+>
<!ATTLIST reuse-param-list
```

```
name CDATA #REQUIRED
>
```

9.5. Customization of CALS table components: abb-tbl.mod

```
<!--
Author : Software AG Belgium - 2001
Comments : abb-tbl dtd
The table model (CALS) for an abb publication.
-->
<!-- Content model for Table. -->
<!ENTITY % tbl.table.mdl "(title?, subtitle*, tgroup)">
<![%wrx;[
<!ENTITY % tbl.tgroup.mdl "thead?,tbody">
] ] >
<!ENTITY % tbl.tgroup.mdl "colspec*, thead?, tbody">
<!ENTITY % tbl.hdft.mdl "(row+)">
<!ENTITY % tbl.row.mdl "(entry)+">
<!ENTITY % tbl.entry.mdl "(%para.class;|%list.class;)">
<![%wrx;[
<!ATTLIST entry
hpos CDATA #IMPLIED
leftedge CDATA #IMPLIED
>
] ] >
<!-- Reference CALS Table Model -->
<!--
<!ENTITY % tablemodel PUBLIC
"-//OASIS//DTD DocBook XML CALS Table Model V4.1.2//EN"
"calstblx.dtd">
%tablemodel;
-->
<!ENTITY % tablemodel SYSTEM
"calstblx.dtd">
%tablemodel;
```

10. ANNEX II: VL0 DTD

10.1. Root declarations: vl0.dtd

```
<!--
Author : Software AG Belgium - 2001
Comments : vl0 dtd
The budgetary dtd for volume 0 publication.
-->
<!ENTITY % rep "INCLUDE">
<!ENTITY % wrx "IGNORE">
<!--
The basic text...
-->
<!ENTITY % vl0-btx.module "INCLUDE">
<![%vl0-btx.module;[
<!ENTITY % vl0-btx SYSTEM
"vl0-btx.mod">
%vl0-btx;
] ] >
<!--
The document hierarchy...
-->
<!ENTITY % vl0-hier.module "INCLUDE">
<![%vl0-hier.module;[
<!ENTITY % vl0-hier SYSTEM
"vl0-hier.mod">
%vl0-hier;
] ] >
<!--
The tables...
```

```
-->
<!ENTITY % vl0-tbl.module "INCLUDE">
<![%vl0-tbl.module;[
<!ENTITY % vl0-tbl SYSTEM
"vl0-tbl.mod">
%vl0-tbl;
] ] >
```

10.2. Nomenclature element declarations: vl0-hier.mod

```
<!ENTITY % heading.class "heading">
<!ENTITY % text.class "%para.class; | %list.class; | %formal.class;">
<!ENTITY % section.class "sect1 | sect2 | sect3 | sect4 | sect5 | sect6 | sect7 | sect8 |
sect9">
<!ENTITY % ext.hier.unit "volume | preamble | part | annex | %section.class;">
<![%wrx;[
<!ENTITY % vl0.attrib "
<!ATTLIST vl0
xml:lang CDATA #IMPLIED
>"
>
] ] >
<!ENTITY % vl0.attrib "">
<!ELEMENT vl0 (headerfooter?, (%ext.hier.unit;))>
%vl0.attrib;
<!ELEMENT headerfooter (((pageheader , pagefooter) |
(leftpageheader, rightpageheader, leftpagefooter, rightpagefooter)) ,
firstpageheader?, firstpagefooter?)>
<!ENTITY % ext.hier.unit.attrib "
alias CDATA #IMPLIED"
>
<!ELEMENT volume (%heading.class;, coverpage?, preamble?, part+)>
<!ATTLIST volume
%ext.hier.unit.attrib;
>
<!ELEMENT part (%heading.class;, coverpage?, toc_heading?, (%text.class;)*, (sect1)*, annex*)>
<!ATTLIST part
%ext.hier.unit.attrib;
>
<!ELEMENT sect1 (%heading.class;, (%text.class;)*, sect2*)>
<!ATTLIST sect1
%ext.hier.unit.attrib;
>
<!ELEMENT toc_heading (heading, (%text.class;)*)>
<!ELEMENT annex (%heading.class;, (%text.class;)* , sect1*)>
<!ATTLIST annex
%ext.hier.unit.attrib;
>
<!ELEMENT sect2 (%heading.class;, (%text.class;)*, sect3*)>
<!ATTLIST sect2
%ext.hier.unit.attrib;
>
<!ELEMENT sect3 (%heading.class;, (%text.class;)*, sect4*)>
<!ATTLIST sect3
%ext.hier.unit.attrib;
>
<!ELEMENT sect4 (%heading.class;, (%text.class;)*, sect5*)>
<!ATTLIST sect4
%ext.hier.unit.attrib;
>
<!ELEMENT sect5 (%heading.class;, (%text.class;)*, sect6*)>
<!ATTLIST sect5
%ext.hier.unit.attrib;
>
<!ELEMENT sect6 (%heading.class;, (%text.class;)*, sect7*)>
<!ATTLIST sect6
%ext.hier.unit.attrib;
>
```

```
<!ELEMENT sect7 (%heading.class;, (%text.class;)*, sect8*)>
<!ATTLIST sect7
%ext.hier.unit.attrib;
>
<!ELEMENT sect8 (%heading.class;, (%text.class;)*, sect9*)>
<!ATTLIST sect8
%ext.hier.unit.attrib;
>
<!ELEMENT sect9 (%heading.class;, (%text.class;)*)>
<!ATTLIST sect9
%ext.hier.unit.attrib;
>
<!ELEMENT preamble (heading, (%text.class;)*)>
<!ATTLIST preamble
%ext.hier.unit.attrib;
>
<!ELEMENT heading (%para.class;)*>
<!ELEMENT coverpage (language, institution?, emission?, reference?, refinter?,
confidentiality?, ptype, ptitle, statut?)>
<!ELEMENT language (%para.class;)>
<!ELEMENT institution (%formal.class;)>
<!ELEMENT emission (%para.class;)>
<!ELEMENT reference (%para.class;)>
<!ELEMENT refinter (%para.class;)>
<!ELEMENT confidentiality (%para.class;)>
<!ELEMENT ptype (%para.class;)>
<!ELEMENT ptitle (%para.class;)>
<!ELEMENT statut (%para.class;)>
<!ELEMENT pageheader (table?)>
<!ATTLIST pageheader
differentfirstpage CDATA #IMPLIED
>
<!ELEMENT pagefooter (table?)>
<!ATTLIST pagefooter
differentfirstpage CDATA #IMPLIED
>
<!ELEMENT leftpageheader (table?)>
<!ELEMENT rightpageheader (table?)>
<!ELEMENT firstpageheader (table?)>
<!ELEMENT leftpagefooter (table?)>
<!ELEMENT rightpagefooter (table?)>
<!ELEMENT firstpagefooter (table?)>
```

10.3. Basic text components declarations: vl0-btx.mod

```
<!--
Author : Software AG Belgium - 2001
Comments : vl0-btx dtd
Basic text elements.
History:
07sep2004 addition of literallayout (paragraph level, not inline) and literal
-->
<!-- ================================================================== -->
<!-- Object-level classes -->
<!-- ================================================================== -->
<!ENTITY % para.class "p|reuse-link|literallayout">
<!ENTITY % formal.class "table">
<!ENTITY % informal.class "object | graphic | caption">
<!ENTITY % list.class "list">
<!-- ================================================================== -->
<!-- Inline-level classes -->
<!-- ================================================================== -->
<![%rep;[
<!ENTITY % char.class "note | ft | qt | literal | fontsize | placeholder">
] ] >
<![%wrx;[
<!ENTITY % char.class "note | emphasis | glossterm | subscript | superscript | uppercase |
literal | fontsize | placeholder">
```

```
] ] >
<!ENTITY % btx "(#PCDATA | %char.class; | %informal.class;)*">
<!-- ============================================================== -->
<!-- Objects -->
<!-- ============================================================== -->
<!ELEMENT p %btx;>
<![%rep;[
<!ATTLIST p
lang CDATA #IMPLIED
>
] ] >
<!-- note: qt is included here but shouldn't: due to a wrong choice
in the implementation of this stupid and silly character to element
conversion, it was not possible to avoid the conversion for such
specific elements assumed to contain things like xml examples
-->
<![%rep;[
<!ELEMENT literallayout (#PCDATA|qt)*>
] ] >
<![%wrx;[
<!ELEMENT literallayout (#PCDATA)>
] ] >
<![%rep;[
<!ELEMENT list (item+)>
<!ATTLIST list
type (dash | bullet | none | arab | uc | lc | romuc | romlc) "dash"
>
<!ELEMENT item (%para.class; | %list.class;)*>
] ] >
<![%wrx;[
<!ELEMENT list (listitem+)>
<!ATTLIST list
type (dash | bullet | none | arab | uc | lc | romuc | romlc) "dash"
spacing (normal | compact) "compact"
inheritnum (inherit | ignore) "ignore"
continuation (continues | restarts) "restarts"
>
<!ELEMENT listitem (%para.class; | %list.class;)*>
<!ATTLIST listitem
override CDATA #IMPLIED
>
] ] >
<!ENTITY % notation.class
"WMF | GIF | PNG | JPG | JPEG | EMF">
<!ELEMENT graphic (pict_data, alt_pict?) >
<!ATTLIST graphic
format (%notation.class;) #REQUIRED
width CDATA #IMPLIED
contentwidth CDATA #IMPLIED
depth CDATA #IMPLIED
contentdepth CDATA #IMPLIED
align (left|right|center|justify) #IMPLIED
valign (top|middle|bottom) #IMPLIED
scale CDATA #IMPLIED
scalefit (true|false) #IMPLIED
format2 (%notation.class;) #IMPLIED
>
<!ELEMENT pict_data (#PCDATA)>
<!ELEMENT object (obj_data, alt_pict, metafile) >
<!ATTLIST object
fileref CDATA #IMPLIED
class CDATA #REQUIRED
format2 (%notation.class;) #REQUIRED
contentdepth CDATA #IMPLIED
contentwidth CDATA #IMPLIED
depth CDATA #IMPLIED
scale CDATA #IMPLIED
scalefit (true|false) #IMPLIED
```

```
width CDATA #IMPLIED
>
<!-- format2: format of the alternate picture -->
<!ELEMENT obj_data (#PCDATA)><!-- serialized object -->
<!ELEMENT alt_pict (#PCDATA)><!-- alternate picture in a format that can be published -->
<!ELEMENT metafile (#PCDATA)><!-- metafile (needed to rebuild a RTF copy) -->
<!ELEMENT caption (%para.class;)>
<!ENTITY % btx.title.mdl "(%para.class;)*">
<!ENTITY % btx.subtitle.mdl "(%para.class;)">
<!ELEMENT title %btx.title.mdl;>
<!ATTLIST title
align (left|right|center|justify) #IMPLIED
>
<!ELEMENT subtitle %btx.subtitle.mdl;>
<!ATTLIST subtitle
align (left|right|center|justify) #IMPLIED
>
<!-- ================================================================== -->
<!-- Inline elements -->
<!-- ================================================================== -->
<!ELEMENT note (%para.class; | %list.class;)*>
<!ELEMENT literal (#PCDATA)>
<!ELEMENT fontsize %btx;>
<!ATTLIST fontsize
size CDATA #IMPLIED
>
<![%rep;[
<!ELEMENT ft %btx;>
<!ATTLIST ft
type (ht | sup | sub | uc | glossary) #IMPLIED
>
<!ELEMENT qt %btx;>
] ] >
<![%wrx;[
<!ELEMENT emphasis %btx;>
<!ELEMENT glossterm %btx;>
<!ELEMENT subscript %btx;>
<!ELEMENT superscript %btx;>
<!ELEMENT uppercase %btx;>
] ] >
<!ELEMENT placeholder (#PCDATA)>
<!-- ================================================================== -->
<!-- Inclusion or reuse-link -->
<!-- ================================================================== -->
<!ELEMENT reuse-link (reuse-param|reuse-param-list)+>
<!ATTLIST reuse-link
reuse CDATA #FIXED "reuse-link"
>
<!ELEMENT reuse-param EMPTY>
<!ATTLIST reuse-param
name CDATA #REQUIRED
value CDATA #REQUIRED
>
<!ELEMENT reuse-param-list (reuse-param)+>
<!ATTLIST reuse-param-list
name CDATA #REQUIRED
>
```

10.4. Customization of CALS table components: vl0-tbl.mod

```
<!--
Author : Software AG Belgium - 2001
Comments : abb-tbl dtd
The table model (CALS) for an abb publication.
-->
<!-- Content model for Table. -->
<!ENTITY % tbl.table.mdl "(title?, subtitle*, caption?, tgroup)">
<![%wrx;[
```

```
<!ENTITY % tbl.tgroup.mdl "thead?, tbody">
] ] >
<!ENTITY % tbl.tgroup.mdl "colspec*, thead?, tbody">
<!ENTITY % tbl.hdft.mdl "(row+)">
<!ENTITY % tbl.row.mdl "(entry)+">
<!ENTITY % tbl.entry.mdl "(%para.class;|%list.class;)">
<![%wrx;[
<!ATTLIST entry
hpos CDATA #IMPLIED
leftedge CDATA #IMPLIED
>
] ] >
<!-- Reference CALS Table Model -->
<!--
<!ENTITY % tablemodel PUBLIC
"-//OASIS//DTD DocBook XML CALS Table Model V4.1.2//EN"
"calstblx.dtd">
%tablemodel;
-->
<!ENTITY % tablemodel SYSTEM
"calstblx.dtd">
%tablemodel;
```

11. ANNEX III: AMENDMENT DTD

11.1. Root declarations: amd.dtd

```
<!--
Author : Software AG Belgium - 2001
Comments : amd dtd
The budgetary dtd for an amd publication.
History:
01jun2004 add lb_amendment_compromise to show where the text comes from
-->
<!ENTITY % rep "INCLUDE">
<!ENTITY % prs "IGNORE">
<!ENTITY % wrx "IGNORE">
<!--
The basic text...
-->
<!ENTITY % amd-btx.module "INCLUDE">
<![%amd-btx.module;[
<!ELEMENT lb_amendment_compromise (#PCDATA) >
<!ENTITY % para.class "lb_amendment_compromise | p | reuse-link">
<!ENTITY % amd-btx SYSTEM
"abb-btx.mod">
%amd-btx;
] ] >
<!--
The tables...
-->
<!ENTITY % amd-tbl.module "INCLUDE">
<![%amd-tbl.module;[
<!ENTITY % amd-tbl SYSTEM
"abb-tbl.mod">
%amd-tbl;
] ] >
<!--
The document hierarchy...
-->
<!ENTITY % amd-hier.module "INCLUDE">
<![%amd-hier.module;[
<!ENTITY % amd-hier SYSTEM
"amd-hier.mod">
%amd-hier;
] ] >
```

11.2. Basic text components declarations: amd-btx.mod

```
<!--
Author : Software AG Belgium - 2001
Comments : abb-btx dtd
Basic text elements.
History:
11feb2004 add a mark with a language attribute to enclose reported authoring text
-->
<!-- ================================================================== -->
<!-- Object-level classes -->
<!-- ================================================================== -->
<!ENTITY % para.class "p | reuse-link">
<!ENTITY % formal.class "table">
<!ENTITY % list.class "list">
<!-- ================================================================== -->
<!-- Inline-level classes -->
<!-- ================================================================== -->
<![%rep;[
<!ENTITY % char.class "note | ft | qt">
] ] >
<![%prs;[
<!ENTITY % char.class "note | ft | qt | prefix">
] ] >
<![%wrx;[
<!ENTITY % char.class "note | emphasis | glossterm | subscript | superscript | uppercase">
] ] >
<!ENTITY % btx "(#PCDATA | %char.class;)*">
<!-- ================================================================== -->
<!-- Objects -->
<!-- ================================================================== -->
<!ELEMENT p %btx;>
<!ATTLIST p
lang CDATA #IMPLIED
>
<![%rep;[
<!ELEMENT list (int.li?, item+, close.li?)>
<!ATTLIST list
type (dash | bullet | none | arab | uc | lc | romuc | romlc) "dash"
>
<!ELEMENT item (%para.class; | %list.class;)*>
<!ELEMENT int.li (%para.class;)?>
<!ELEMENT close.li (%para.class;)?>
] ] >
<![%prs;[
<!ELEMENT list (int.li?, item+, close.li?)>
<!ATTLIST list
type (dash | bullet | none | arab | uc | lc | romuc | romlc) "dash"
>
<!ELEMENT item (%para.class; | %list.class;)*>
<!ELEMENT int.li (%para.class;)?>
<!ELEMENT close.li (%para.class;)?>
] ] >
<![%wrx;[
<!ELEMENT list (listitem+)>
<!ATTLIST list
type (dash | bullet | none | arab | uc | lc | romuc | romlc) "dash"
spacing (normal | compact) "compact"
inheritnum (inherit | ignore) "ignore"
continuation (continues | restarts) "restarts"
>
<!ELEMENT listitem (%para.class; | %list.class;)*>
<!ATTLIST listitem
override CDATA #IMPLIED
>
] ] >
<!ENTITY % btx.title.mdl "(%para.class;)*">
<!ENTITY % btx.subtitle.mdl "(%para.class;)">
```

```
<!ELEMENT title %btx.title.mdl;>
<!ATTLIST title
align (left|right|center|justify) #IMPLIED
>
<!ELEMENT subtitle %btx.subtitle.mdl;>
<!ATTLIST subtitle
align (left|right|center|justify) #IMPLIED
>
<!-- ================================================================== -->
<!-- Inline elements -->
<!-- ================================================================== -->
<!ELEMENT note (%para.class; | %list.class;)*>
<![%rep;[
<!ELEMENT ft %btx;>
<!ATTLIST ft
type (ht | sup | sub | uc | glossary) #IMPLIED
>
<!ELEMENT qt %btx;>
<!ELEMENT lang %btx;>
<!ATTLIST lang
lg CDATA #REQUIRED
>
] ] >
<![%prs;[
<!ELEMENT ft %btx;>
<!ATTLIST ft
type (ht | sup | sub | uc | glossary) #IMPLIED
>
<!ELEMENT qt %btx;>
<!ELEMENT prefix %btx;>
<!ELEMENT lang %btx;>
<!ATTLIST lang
lg CDATA #REQUIRED
>
] ] >
<![%wrx;[
<!ELEMENT emphasis %btx;>
<!ELEMENT glossterm %btx;>
<!ELEMENT subscript %btx;>
<!ELEMENT superscript %btx;>
<!ELEMENT uppercase %btx;>
] ] >
<!-- ================================================================== -->
<!-- Inclusion or reuse-link -->
<!-- ================================================================== -->
<!ELEMENT reuse-link (reuse-param|reuse-param-list)+>
<!ATTLIST reuse-link
reuse CDATA #FIXED "reuse-link"
>
<!ELEMENT reuse-param EMPTY>
<!ATTLIST reuse-param
name CDATA #REQUIRED
value CDATA #REQUIRED
>
<!ELEMENT reuse-param-list (reuse-param)+>
<!ATTLIST reuse-param-list
name CDATA #REQUIRED
>
```

11.3. Customization of CALS table components: amd-tbl.mod

```
<!--
Author : Software AG Belgium - 2001
Comments : abb-tbl dtd
The table model (CALS) for an abb publication.
-->
<!-- Content model for Table. -->
<!ENTITY % tbl.table.mdl "(title?, subtitle*, tgroup)">
```

```
<![%wrx;[
<!ENTITY % tbl.tgroup.mdl "thead?,tbody">
] ] >
<!ENTITY % tbl.tgroup.mdl "colspec*, thead?, tbody">
<!ENTITY % tbl.hdft.mdl "(row+)">
<!ENTITY % tbl.row.mdl "(entry)+">
<!ENTITY % tbl.entry.mdl "(%para.class;|%list.class;)">
<![%wrx;[
<!ATTLIST entry
hpos CDATA #IMPLIED
leftedge CDATA #IMPLIED
>
] ] >
<!-- Reference CALS Table Model -->
<!--
<!ENTITY % tablemodel PUBLIC
"-//OASIS//DTD DocBook XML CALS Table Model V4.1.2//EN"
"calstblx.dtd">
%tablemodel;
-->
<!ENTITY % tablemodel SYSTEM
"calstblx.dtd">
%tablemodel;
```